Detecting Semantic Similarity in Questions

Arpan Mukherjee
Dept. of CSE, IIIT-Delhi
arpan17007@iiitd.ac.in

Prabhat Kumar
Dept. of CSE, IIIT-Delhi
prabhat17036@iiitd.ac.in

I. Problem Statement

Search is defined by queries, and more often than not, these are in the form of questions. There are many such *Community Question Answering(CQA)* platforms, such as Quora [1], Stackoverflow [3], Yahoo Answers [4] etc. , where users can ask questions and other fellow users provide with answers.

However, there are many instances when questions are rephrased and asked again. Ideally, such questions which are semantically same should be merged. Merging of such questions or avoiding creating additional threads for the duplicate question can have many advantages. Firstly, merging would result in significant time saving on users part, where one can get most related information in one go, and wont have to navigate to further rephrased questions or ask another question and wait for the answer. Secondly, it will lead to a collective knowledge base which is useful from the perspective of both users and researchers planning to understand to study the correlation between questions and answers. Also, combined knowledge bases can help in creating better and effective domain-specific knowledge repository.

How one defines two questions as duplicate? We say two questions are duplicate if the users asking the questions have expressed similar *intent*, i.e., users expect the same kind of answers to the questions asked.

Through this project, we aimed to model this *intent*, i.e., given two questions classify if the purpose specified by the two questions are similar or not.

II. Literature review

Finding similar content on CQA forum has a long history on academic research and industry as well. [8] conduct a comparative study on traditional machine learning algorithms such as Support Vector Machines(SVMs) by using handpicking features and prepossessed data, which performed well on SemEval-2015 dataset. They made an argument that deep learning models are extensively dependent on data, it has a huge limitation on small or noisy dataset.

Nevertheless, we can't argue with the recent success of deep learning in spite of having a heavy dependency on the quality of the dataset. Most of the initial Natural Language Sentence Matching(NLSM) relies on the Siamese neural network architecture [6]. In this architecture, the same neural network model is being used to encode two individual sentences which are given as input independently. Both the input sentences are now encoded into sentence vector in the same embedding space, as shown in Figure 1. Then by using some distant metric decision will be made solely based on this result [13]. Later [5] showed that by using this architecture paired convolutional neural network(CNN) with cosine similarity distance measure provided a much better result than traditional Jaccards similarity or SVMs in identifying duplicate questions in StackExchange dataset. Shared parameters make the model smaller and easier to train. However, the disadvantage is that theres no interaction between two inputs while encoding, which may cause some loss of important lexical information.

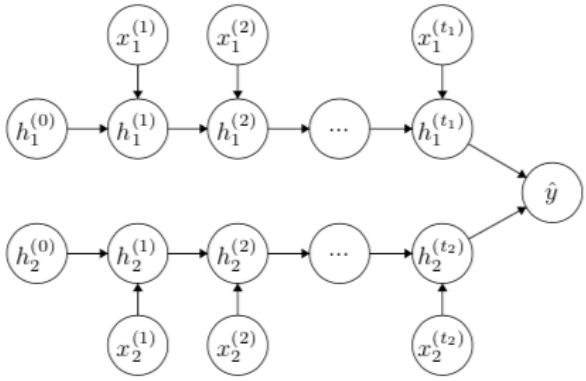

Fig. 1. General Architecture of Siamese Model

Another architecture matching aggregation has been proposed by [14]. This framework considers and matches semantic similarity at different levels of granularity (word-level, phrase-level and syntax-level), and then matched results are aggregated (by CNN or LSTM) into a vector in order to make the final decision. It captures more smaller units which provides improvement, however it has some limitations as well, matching is only performed at a single direction

[15] proposed a bilateral multi-perspective matching(BiMPM) model. Given two sentences P and Q BiMPM LSTM encodes both of them and then they match encoded sentences into two directions $P \rightarrow Q$ and $Q \rightarrow P$, each time step of Q is matched against all time-steps of P from multiple perspectives. Another BiLSTM layer is utilized to aggregate the matching results to into a fixed-length vector and based on the matching vector, a decision is made through a fully connected layer.

III. Dataset

A. Introduction

The dataset to be used for the study has been provided by Quora [1], a Question - Answer platform not specific to any domain.

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

Fig. 2. Sample of the Quora Question Pairs dataset [7]

The dataset contains 404,352 question pairs as per the format given in Figure 2. id represents the unique question pair identifier. $qid1$ and $qid2$ represent the question identifier for first and the second question in the pair respectively. $question1$ and $question2$ are the full text of Question 1 and Question 2 respectively.

IV. Proposed Algorithm

A. Preprocessing and Analysis

From the dataset of the 400K question pairs, roughly 40 percent are labeled as duplicate and the rest 60 percent as unique or non-duplicate.

Fig. 3. Wordcloud of top 2000 most frequently occurring words

Generic NLP preprocessing techniques were applied such as punctuation removal followed by stop-words removal.

As shown from the Figure 7, wh-questions dominate the dataset as expected from a CQA platform dataset. But it gives an idea about the subject of interest of a large number of questions in the dataset such as *India*, therefore eliminating the use of vocabulary based word removal or spell check as these proper nouns may play a key role in defining the similarity of context between the question pairs. Also as the Quora officially released the dataset, so our primary assumption was each question would be syntactically correct. On platforms like Quora - there are moderators assigned for these kinds of tasks as well, so we can safely assume grammatically and spelling wise our dataset is correct. But later we realized our mistake and done the extensive preprocessing cleaning on the dataset using regex expression. E.g., I'll, I've was converted to "I will" and "I have" respectively [2].

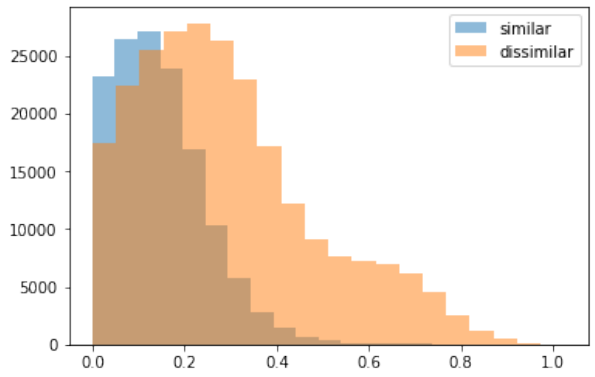

Fig. 4. Cosine distance between similar and dissimilar embedding vectors

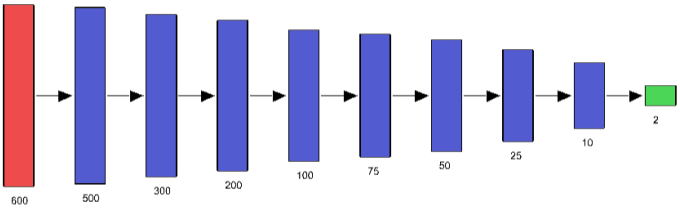

Fig. 5. 8-Layer NN Architecture

B. Models

1) Support Vector Machines: SVMs also known as Support Vector Machine, are a discriminative classifier which works on the concept of increasing the margin between the classes in a hyperspace. SVM with Linear kernel was applied and used as a baseline for the classification task. Results from the model helped in better understanding of the data.

2) Dense Network: Several Dense layer-based architectures were tested, by varying number of layers, number of neurons in each of the layers. Both cone based and inverse bottleneck architecture was evaluated. Finally, an 8-layered cone based architecture as shown in Figure 5 with ReLU activation was used.

3) Autoencoders: Question similarity can also be seen as a generative task of generating questions that are similar to the input. Therefore, these tasks can be modeled using Autoencoders where the task is to learn a representation after the encoder such that the autoencoder minimizes the loss between the representation of two similar questions. For this task, the model was trained on only similar question pairs, and no comment can be made about the representation of non-similar question pairs. The following loss function L was minimized

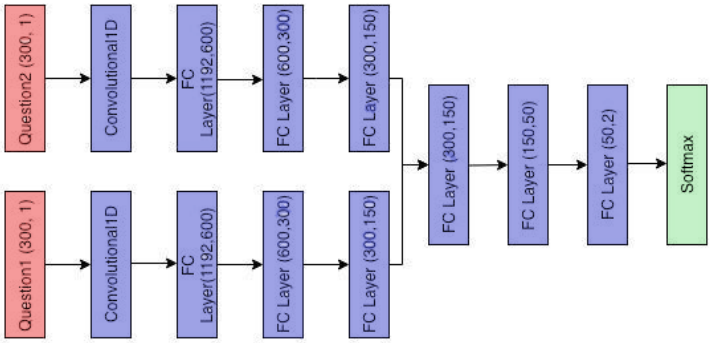

Fig. 6. Basic Siamese Architecture

$$L = (M(X_1) - X_2)^2$$

where X_1 and X_2 represent the two similar question in a pair and $M()$ is the autoencoder's output. Later a classification layer was added to the concatenated output of two question after the encoder layer through this network can also be seen as a siamese network where the representations are learned by an autoencoder based upon the similarity only.

4) Convolutional Siamese Network: Extensive work has been performed regarding the use of Siamese network for similarity deduction. On similar grounds, a convolutional siamese network as shown in Figure 6 were used for experimentation. The network had a 1-dimension convolutional layer followed by three fully connected dense layer which reduced the dimension of data from a 300-dimension vector to a 150-dimension vector. The weights of these four layers were shared, i.e., where same for the two streams. The representation received were further concatenated to form a 300-dimension vector which was followed by a classification layer.

5) LSTM Siamese Network: Although neural network based approaches have been quite useful, in this model, we tested the hypothesis better representation can be learned by incorporating the knowledge of "long-term dependencies," which simple NN was not able to learn properly. In theory, RNNs are capable of handling such long-term dependencies. A human could carefully pick parameters for them to solve toy problems of this form. Sadly, in practice, RNNs dont seem to be able to learn them, yet they perform better than simple NN or CNN.

Long Short-Term Memory(LSTM) networks -are a special kind of RNN, capable of learning long-term dependencies [9]. In this model, we tried to use long-range dependencies between the words of a sentence, which plays an essential role in the syntactical meaning of the content.

Initially, we created our vocabulary based on the dataset, and later word level embedding matrix was used as the feature vector. For the embedding of a single word, we used standard pre-trained word2vec [12] model and leave the unrecognized ones (less than 0.5%) random. Padding was done on sentence vector for symmetricity. As shown in Figure 8, Siamese LSTM layer's outputs were merged and passed through Manhattan similarity function to predict the output.

V. Results

Method	Accuracy	Embedding Type
Support Vector Machine	59.23	DOC2VEC
NN Model	79.28	DOC2VEC
Convolutional Siamese Model	64.33	DOC2VEC
Autoencoder Siamese Model	62.85	DOC2VEC
LSTM Siamese Model	80.32	WORD2VEC

TABLE I
ACCURACY ON TEST SET

The Neural Network Architecture was trained for 100 epoch with Adam Optimizer at a learning rate of 0.001. The same configuration was used for training of Autoencoder based Siamese Architecture and was trained of 75 epoch, and later the classification layer was trained for 30 epochs.

The LSTM Siamese Architecture was trained with learning rate 0.001 and Mean Square Error as the loss function with AdaDelta optimizer.

For classification layer, Cross Entropy loss has been used.

We also tried different types of word level embedding and(e.g. - GloVe, tf-idf), which yields huge dimensional vector. Due to resource constraint we couldn't test our model on those embedding. For tf-idf single word dimension was 84334, and it was highly sparse matrix.

VI. Analysis

As we used SVM with linear kernel, it was Initially; representations were learned using doc2vec [10] model, which yields a 300-dimensional embedded vector for any sentence. But the conversion of each sentence into a fixed dimensional vector doesn't seem very logical. It tries to encode the whole content in a fixed length, so data loss is meant to happen in this case. Also, sentence level embedding doesn't consider or prioritize inter-dependencies between words, which plays a significant role in the syntactical meaning of any textual data.

A simple NN model can perform well compared to the CNN model. We tried out different architecture where we varied the number of convolutional(conv) operations and also the number of FC layers in the architecture. Ultimately we ended up with a single conv operation(*kernel size*=1). We have tried out

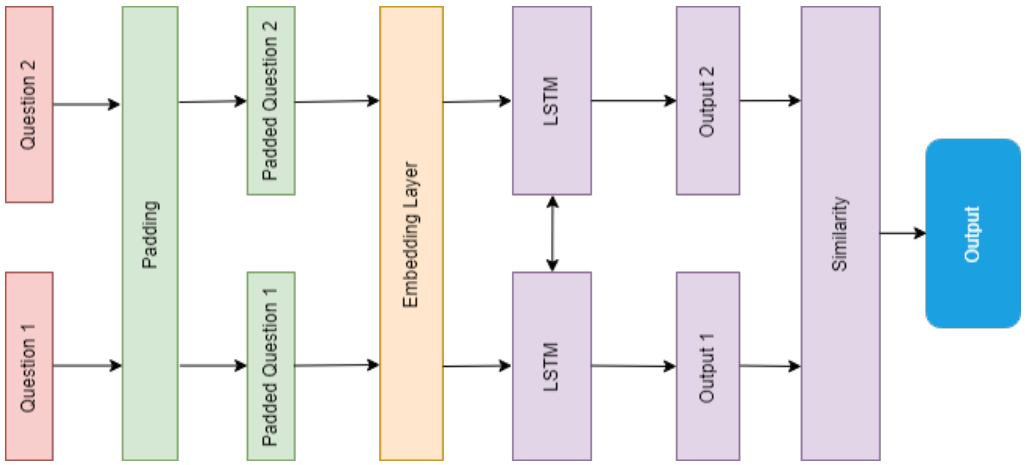

Fig. 7. Siamese LSTM Architecture

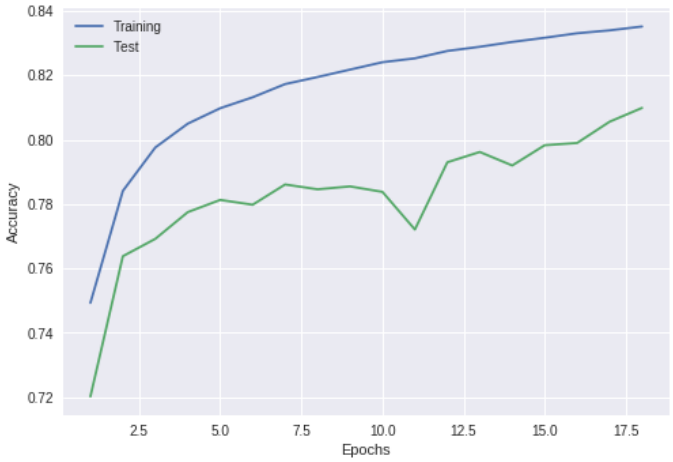

Fig. 8. Variation of accuracy with epoch for balanced dataset

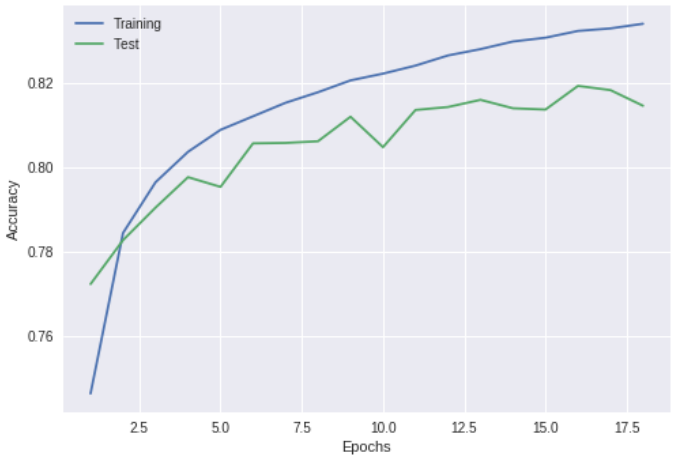

Fig. 9. Variation of accuracy with epoch for unbalanced dataset

multiple architectures considering concatenated embedding of both the sentences.

But none of those mentioned above architectures were able to capture *long-term dependencies*, which are one of the critical factors for NLP classification domain. The LSTM model was able to remember previous contextual information in a much better way compared to simple NN or CNN, which helped it to achieve better accuracy.

VII. The Dont's

- Creation of balanced dataset, in case of an unbalanced test set the results may lead to wrong inferences.
- For NLP based task, prepossessing plays an important role in defining the final performance of the model, a wrong or an incomplete reprocessing step can lead to either addition of noise or deletion of significant features that may come in hand for task.
- Sentence level embedding often don't work given the sentence have high variation in length and subject covered by these sentences.
- Applying Batch Normalization is not always gives a boost in performance, specially if applying after dense layers.
- Semantic level dependency is hard to model without the usage of time-series architecture.

VIII. Learning

- Effect of prepossessing on the performance of model
- Use of Distance function such as Cosine has classification criteria
- Getting familiar with time-series based network models
- Applying autoencoders to a classification task partially.

IX. Conclusion

The similarity between Natural Language such as English is heavily based upon the sequence in which the words are used, whether it be a question, statement or an expression. These dependencies are hard to be captured by models which look for dependence in static input. Architectures using RNN, LSTM which are often used to model time-series dependency can be seen to capture these dependencies and show robust

performance as compared to models such as Neural Network and SVMs. But a more thorough study through experimentation of a wide variety of dataset, not just questions will help model architectures that are more generic. Also, useful word embedding trained on extensive data, with robust similarity metric can also be explored. Also for the future work, we can use *Attention* [11] architecture instead of simple LSTM. In this architecture instead of given uniform weight to every word, weighted factor has been learned, i.e., which word should be given more *attention* to capture the much underlying dependencies.

Embedding of the words which we couldn't find in the pre-trained word2vec model was replaced by normal distribution values. Even though number of unknown words were really less, still in order not to miss out the information we can train our own word2vec model, which will learn the data distribution of the given corpus.

REFERENCES

[1] Quora.

[2] Quora question pairs — kaggle discussion.

[3] Stackoverflow.

[4] Yahoo answers.

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

[6] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[7] Kornl Csernai. First quora dataset release: Question pairs, 2017.

[8] Kuntal Dey, Ritvik Shrivastava, and Saroj Kaushik. A paraphrase and semantic similarity detection system for user generated short-text content on microblogs. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2880–2890, 2016.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[10] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

[11] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[13] Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*, 2015.

[14] Shuohang Wang and Jing Jiang. A compare-aggregate model for matching text sequences. *arXiv preprint arXiv:1611.01747*, 2016.

[15] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*, 2017.