Detecting Duplicate Questions on Quora with Deep Learning

Abhishek Jindal [1] **Oscar Chang** [1] **Siddharth Varia** [1]

Abstract

Detection of semantic similarity in text is an important and well researched problem in natural language processing. Recent advances in deep learning have highlighted promising new approaches for this task including the Siamese LSTM architecture (Mueller & Thyagarajan, 2016). We explore several variants of these approaches and propose a new approach to this problem based on entailment. For evaluation, we apply our models to the task of identifying questions with similar intent in the Quora Duplicate Questions dataset. We are able to match Quora's benchmark of 87% accuracy for this dataset and discover interesting insights from our results. We observe, perhaps a bit counter−intuitively that a simple deep LSTM architecture with concatenated hidden representations of the question pairs achieves the best validation accuracy with the shortest training time. We hypothesize that this is because the amount of data is insufficient to achieve the capacity of the more complicated models, and show theoretical reasons for this based on Gersgorin's Circle Theorem (Julian Georg Zilly, 2017)

1. Introduction

Websites like Quora, StackOverflow and Reddit etc. provide a platform for their users to ask a wide variety of questions and get answers from other members of the community. Some of these websites are more user−moderated compared to others. For example, on StackOverflow, if one posts a duplicate question, other users with a high reputation can flag that question and can delete it. It will be very useful for these websites if they can automatically identify these duplicate questions and retain only on them. Clearly, this has huge advantages like better search experience for the users and a higher quality knowledge base. Despite these advantages, none of the Q&A sites have this feature completely in place because capturing semantic similarity is a non-trivial task. In the recent years, with the advent of deep learning techniques in the field of NLP, there is a renewed hope that this problem can be solved. Quora recently launched a competition on Kaggle where given a set of question pairs, the task is to predict the probability of the questions being duplicates of each other. In this paper, we describe our methods for tackling this problem and compare and discuss the results achieved. We share the insight we gained by our analysis and hope that it will be useful to other researchers in this field. The paper is organized as follows: in section 2 we cover related work, in section 4 we describe our approaches in detail and this is followed by the results and concluding remarks in sections 5 and 6 respectively. Some of the relevant figures are included in section 8.

[1] Computer Science Department, Columbia University, New York, USA.

2. Related Work

Most of the traditional approaches for detecting text similarity and, more specifically for detecting sentence similarity were adapted from the techniques used in information retrieval to compute document similarities. These methods fall in to 3 broad categories: word co−occurrences, statistical information of words in huge corpses (latent semantic analysis) and human engineered features. However, all these approaches worked well for long bodies of text and failed for short text like individual sentences.

Li et al. proposed a method to compute semantic sentence similarity based on the use of a lexical database like WordNet. In their approach, they compute the semantic similarity between sentences by decomposing it into semantic similarity between words, which they calculate by exploiting the hierarchical structure of WordNet.

Most of the recent efforts in capturing semantic similarity have used variants of LSTMs. Mueller & Thyagarajan proposed Manhattan LSTM model where the pair of sentences are encoded using the same LSTM. These encodings are then used to compute similarity $g(h_{T_a}^{(a)}, h_{T_b}^{(b)}) = \exp(-||h_{T_a}^{(a)} - h_{T_b}^{(b)}||_1) \in [0, 1]$. In other words, they define Manhattan distance between the LSTM encodings of the sentences in a pair as a similarity measure. The trained MaLSTM captures semantic similarity by simply aggregating differences in various characteristics. The Quora 's engineering team proposed two different variants of LSTM networks namely “LSTM with concatenation” and “LSTM with distance and angle” and reported an accuracy of around 87 % using these networks.

(Kim, 2014) applied convolutional neural networks to the task of text classification. Since our task is a binary classification task, we also used CNNs along side LSTMs. Unlike Images, when used for text classification, the convolution is performed in one direction by moving a sliding filter along the length of the sentence. In his work, CNNs produced state of the art results for many of the benchmark datasets.

3. Dataset

The Quora Duplicate Questions Dataset [Shankar Iyer & Csernai] consists of 404288 pairs of questions, of which 149263 (37%) are similar/duplicate pairs and 255025 (63%) are dissimilar/non-duplicate pairs. The pairs are labeled with either a ’0’ indicating dissimilarity or a ’1’ indicating similarity. Of course, there are several ambiguous examples where it’s hard to say if two questions are truly duplicate or not. For instance, the pair ”How do I become a better programmer?” and ”What skills should a programmer learn to get better?” is ambiguous because the latter question may solicit answers for improving a programmer’s non-technical skills. The Quora team that released the dataset acknowledged that the labels might be noisy, and we can’t trust the ground truth entirely. Also, it’s uncertain how the labels have been sourced, but we speculate that the labeling process is machine-aided and not solely human driven. Nonetheless, our task is a binary classification problem. Given the noise inherent in the data, we should not expect to have an incredibly high validation accuracy ($> 90\%$).

Quora also noted that their original sampling method results in an imbalanced dataset with a lot more similar than dissimilar pairs. They thus augmented the dataset with several pairs of questions they knew to be topically related, but are non-duplicate question pairs. They also emphasize that the distribution of questions in this dataset is not an unbiased sample of the distribution of questions asked on Quora, because of the many data engineering methods they used, for example removing questions above a certain length. We also had to further sanitize the given dataset, by removing a couple of entries that had ‘NaN’ as a given question and doing some traditional text preprocessing like stopwords removal, stemming etc.

4. Experimental Setup

4.1. Word versus Sentence Embeddings

We embedded the input text into a matrix of numbers that the neural network architecture can access. A one-hot encoding is sparse and not very useful. We decided to compare pretrained word and sentence embeddings, and see which is more suitable for the task. Word2vec correlates words with other words and are thus suitable for word embeddings, whereas doc2vec correlates words with document labels and are thus suitable for sentence embeddings. We use word embeddings pretrained from the Google News corpus which has 3 billion wrds, and sentence embeddings pretrained from the full collection of English Wikipedia articles.

We create two simple deep LSTM architectures, one using word embeddings (Fig. 7) and the other using sentence embeddings (Fig. 6). We find that the word embeddings model quickly achieves 83 % validation accuracy (Fig. 4) using about half as many parameters as the sentence embeddings model (Fig. 5), which only achieves 69 % validation accuracy.

We suspect that the significantly worse performance of the sentence embeddings model is because it was pretrained on Wikipedia articles, which probably has very distinct distributional semantics from Quora questions. In Fig. 1, we show a PCA plot of ten question pairs (a - j), where only f and h are dissimilar question pairs. Despite f being dissimilar, we notice that the f pair is very close to each other. The b pair is similar, yet is very far apart in the plot. We demonstrate PCA plots in Fig. 2 and Fig. 3, with new set of ten all similar and all dissimilar pairs respectively. This simple visualization informs us about the susceptibility of the pretrained sentence embeddings to the inherent noise in the dataset, particularly creating a lot more false positives than false negatives, i.e. there are more instances of dissimilar question pairs being much closer together than similar question pairs being far apart. This validates our intuition that word embeddings are a better choice of this task, since there would be more variety in the embeddings that wouldn't overfit the noise in the dataset.

4.2. Methods

We experimented with various LSTM networks including the siamese LSTMs discussed earlier in related work. We tried siamese LSTMs with and without weight sharing. In one case, we use a single LSTM and encode both the questions using the same LSTM whereas in other case, we use a separate LSTM to encode each question. In the most basic version of siamese LSTMs, we concatenated the two sentence representations output from the LSTMs and fed the concatenated vector into a dense layer to produce the final classification result. Apart from naive concatenation, we also experimented with other similarity metrics to measure the semantic similarity between the sentence representations output by the LSTMs. To be specific, we tried Euclidean distance, cosine distance, Hadamard product, Manhattan distance etc. of the representations of the two questions instead of a simple concatenation. For all the experiments with siamese LSTMs, we used pre-trained Word2Vec vectors as initial embeddings.

4.3. Entailment based architecture

We utilize the concept of entailment to identify duplicate text. Entailment (Rocktäschel et al., 2015) basically determines whether two sentences contradict each other, are not related or if the first sentence entails the second sentence. In entailment, first sentence is usually referred to as the premise and the second sentence is referred to as the hypothesis.

In our approach, we model the task of detecting duplicate questions such that the first question for each instance is treated as the premise and the second question is treated as the hypothesis. And then we process the hypothesis conditioned on the premise. And then in the second stage of our algorithm, we treat the second question as the premise and the first question as the hypothesis. In both these cases, we determine entailment by passing the premise and the hypothesis through a neural network based on LSTM units. The premise is read by the first LSTM and the second LSTM with different weights reads the hypothesis but the memory of the latter is initialized with the final cell state of the first LSTM to condition it on the final hidden representation of premise. We utilized Glove embeddings for the embedding vectors of the questions, for out of vocabulary words, we set them to a random vector representation. We trained our model using just pretrained embeddings and also by training the embeddings. For the final prediction, we use a softmax over the final output layer and predict the label '1' for questions detected as duplicate (both-sided entailment detected) and label '0' for non-duplicate pair of questions, and we minimized the binary cross entropy loss with the Adam and SGD optimizers.

We use attention to remove the constraint of capturing the entire sentence 's information in the LSTM 's single cell. Instead we attend over all the output vectors produced by the first LSTM when reading the words of the hypothesis such that output of latter can soft-align with most important words from the premise.
To improve our model, we also experimented with word by word attention where instead of attending over the first LSTM 's output vectors in a static manner, we apply attention while inputting each word of the hypothesis such that we get attention weight vectors over all the output vectors of the premise for every hypothesis word. Word by word attention model should help us achieve better soft$-$alignment between the words and phrases in the premise and the hypothesis.

$$\begin{aligned} M_t &= \tanh(W^y * Y + (W^h * h_t + W^r * r_{t-1}) \otimes e_L) \\ \alpha_t &= softmax(w^T * M_t) \\ r_t &= Y * \alpha_t^T + \tanh(W^t * r_{t-1}) \\ h^* &= \tanh(W^p * r_N + W^x * h_N) \end{aligned} \tag{1}$$

Equation 1 describes the word by word attention process mathematically. Here h_N is the final hidden state of the LSTM after processing the premise. h_t and r_t are hidden state representation and attention representation corresponding to t_{th} word of the hypothesis respectively. α_t is the probability distribution of soft alignment of words of the premise with t_{th} word of hypothesis.

4.4. Tricks used during training

1. Using cross validation, we identified the ballpark ranges for important network parameters like out the dropout rate, the recurrent drop out rate and the number of hidden units in the LSTM units. Now we generated an ensemble model such that we created 20 similar models and the only difference between these models was they had randomly initialized hyperparameters (values close to the best value we had identified through cross validation). Then, we created our final predictions by combining the predictions of these models. We divided the probability range from 0 to 1 into 10 equal sized intervals such that for each interval, all the probabilities having values higher than that interval add votes for that interval. In the end, we just return the average probability of the interval having the maximum number of votes. This technique results in an improvement of nearly 2% validation accuracy for all our models.

2. While analyzing the performance of our models, we plotted calibration curves for the predicted probabilities against the true probabilities in order to analyze failures. We then performed isotonic regression on the output probabilities in order to calibrate them with the true distribution of our data. However, the model didn't show any significant improvement when using isotonic regression.

3. Our training dataset and test dataset had different skew i.e. test data had a larger proportion of duplicate questions compared to the training data, and since our

test data is 5 times bigger than our training data, we changed our loss function according to the skew in the test data.

4. The test data had 2.5 million instances, and it was cumbersome to test our model 's performance. Therefore, we used a batch size of 8192 during the testing phase.

5. We fed the final hidden state representation of the questions to a XGBoost classifier as an experiment and this approach gave results for the small subset of the data. We couldn't pursue this option further because of time constraints.

6. We lengthened the data by randomly replacing some of the words in questions with some synonyms in each minibatch during training. The synonyms were selected using a combination of word2vec and WordNet similarity. The data augmentation resulted in only a very minor gain accuracy.

5. Results and Evaluation

We found that concatenating question pairs and then training simple deep LSTMs on them prove to deliver the highest validation accuracy, while at the same time minimizing training time. This largely mirrors the findings by Quora's internal data science team: Dandekar reports that adding an attention mechanism to the LSTM actually decreases rather increase validation accuracy.

A recent paper by Julian Georg Zilly provides a new theoretical analysis of LSTMs based on the Gersgorin's Circle Theorem, which states that the eigenvalues of a matrix are always located within the union of the complex circles centered around its diagonal values whose radius is equal to the sum of the absolute values of the off-diagonal entries in each row. They showed that the susceptibility of an LSTM architecture to the exploding gradient problem is extremely sensitive to the initialization conditions of the transition matrices. This is because the exploding gradient problem occurs when the off-diagonal entries of the transition matrices are too large. The success of an attention mechanism in an LSTM can thus in part be attributed to the fact that it causes smaller gradients to be propagated.

In this case, it is important to note that the lengths of the questions are all relatively short, since the Quora team that collected the data had already removed long questions from the dataset. As a result, the LSTM architecture does not really suffer from the exploding or vanishing gradient problem in this case. This is corroborated by the fact that across the variety of LSTM architectures that we tested, the validation accuracy plateaus very quickly (in less than five epochs). In cases where overfitting happens rather quickly, we should aim for a lower capacity model rather than a higher capacity one. The more complicated LSTM architectures like the attention LSTMs and the Siamese LSTMs are higher capacity models than the simple deep concatenated LSTM and hence, are more prone to overfitting the data. This explains why they do not perform as well. If our analysis is correct, then the more complicated LSTMs should perform better when the size of the dataset is an order of magnitude larger.

Table 1. Accuracy of various models in %

Model	Accuracy on Validation Set
Siamese LSTM	83
Siamese LSTM + Convolution	84
Siamese LSTM + Convolution + Data Augmentation	85
Bi-LSTM + Cosine Similarity + Euclidean Distance	84
Entailment with with 2-way Attention	84
Entailment with 2-way word by word attention	80
20 Ensemble models of 4th model in this table	87

$$spec(A) \subset \bigcup_{i \in \{1 \cdots n\}} \left\{ \lambda \in C \;\middle|\; \parallel \lambda - a_{ii} \parallel_C \leq \sum_{j=1, j \neq i}^{n} |a_{i,j}| \right\} \quad (2)$$

5.1. Postmortem Analysis

On a close analysis of the data, it seems the dataset is extremely noisy and has been generated synthetically. Considering semantic similarity is a slightly abstract criteria, some question pair instances are very hard for even a linguist to classify as a duplicate or not as a duplicate. Some questions are just gibberish and make no grammatical or semantic sense.

Q1. Is it good for health 0 drink 20-30 ml of whisky / rum after dinner daily?
Q2. Is humanity, love only answers exist to share in social media?

Model Prediction: 0.09 probability of being duplicate. Model does well when there are disjoint entities and words with different distributional similarity in the two questions.

Q1. How does gravity dilate time?
Q2. I figured out why speed dilates time, but how does gravity dilate time?

Model Prediction: 0.82 probability of being duplicate. Model does well here too, because of our entailment model, this case is perfectly captured.

Q1.How do TIME Travel really possible? What should I know before doing it?
Q2.Will backwards time travel ever clothes possible?

Model Prediction: probability 0.75 of being duplicate. Model does well here too, despite some noisy words being present “clothes“ in Q2. In many other instances, we observe that our model is noise tolerant to some extent and its predictions dont change if only a few noisy words are present in the input.

Q1. What are best places to visit in UK?
Q2. What are the best places to go in the USA?

Model Prediction : 0.9 probability of being duplicate.Because in pretrained embeddings there is high degree of similarity between UK and USA. This implies, that pretrained embeddings here will not be sufficient to detect questions having different named entities. There are many such cases in the dataset.

Our remedy : We can extract named entities using create a feature vector capturing the “realworld“ similarity of these named entities. Then we can append this feature vector to the embedding vector to provide this information to our model.

Q1. I to be a climbing bum?
Q2. Do scorpions climb walls?

Model Prediction : 0.45 probability of being duplicate. The first sentence makes no sense to us, even as humans, we are supposed to be better at language understanding. In this case, we dont know what the gold label should be so it makes sense that the model is confused too.

Q1.Does self really exist?
Q2.Does ghost exist? If yes, then justify their 7?

Model Prediction: 0.28 of being duplicate. Our model is tolerant against variation in non-named entities.“self“ and “ghost“ perhaps dont have enough distributional similarity as named entities.

Q1. Is there an doesn’t life?
Q2. What is exactly books after life?

Model Prediction : 0.63 probability of being duplicate. This is one of the many seemingly synthetic questions present in the dataset. We still dont know how to handle these cases.

Overall, this analysis helped us infer that because of the extremely noisy data, it will be best to correct the grammatical and other errors by evaluating the perplexity of each of the questions under the English language model. Then, we can consider each of the high perplexity questions and modify the question by replacing one of the words in the question such that the overall perplexity of the question is minimized.

6. Conclusion

In this paper, we show the performance of the various state of the art neural networks in identifying text having semantic similarity on the Quora duplicate questions dataset. We designed a new architecture using twoway wordbyword attention with entailment. We proposed several performance tweaks and tricks that can result in a significant gain in the accuracy. Our results demonstrate that an ensemble model of simple LSTM networks outperforms sophisticated networks with larger number of parameters by a significant margin. Not only are our results at par with the Quora Engineering Team, we also find that they also came to the same conclusion about simple LSTM networks outperforming more complicated networks. We provide a possible theoretical explanation for this seemingly unintuitive observation based on the Gersgorin's Circle Theorem and the small size of our training data. We do a thorough postmortem analysis of some of the interesting observations based on the predictions of our models on the test data. Overall, we feel that we made significant theoretical and practical contributions which can give other researchers insight and direction for solving the task of detecting semantic similarity in text. As far as the Kaggle competition goes, we were as high as top 60 when we had last submitted (2 weeks ago) and from the contest discussion's section, it seems that the traditional feature engineering is triumphing so far.

6.1. Future Work

Our future work will focus on doing some more preprocessing on the data, handling nonsensical and synthetically generated questions. Another avenue of exploration is concatenating named entities based features for improving the accuracy of our model. Additionally, it will be worthwhile to explore more datasets for this task so that we can test our conjecture about the relatively poor results of our entailment network. On the basis of our results and analysis, we infer that wide and deep networks which utilize the strength of deep learning combined with the power of traditional natural language processing techniques for feature engineering as the way to go.

7. Acknowledgement

We will like to thank the course instructors for the course, Dr. LiangLiang Cao, Dr. Xiaodong Cui for their constant guidance and support and a special thanks to Dr. Kapil Thadani for providing us a lot of insight about applying deep learning to natural language processing and helping us survey a large part of this field. We will also like to thank the teaching assistant Chad DeChant for his kind help regarding technical issues and Yizhou for providing us Google Cloud Credits.

References

Dandekar, Nikhil. Semantic question matching with deep learning. *Engineering at Quora*, 13 Feb 2017. URL https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Lea

Julian Georg Zilly, Rupesh Kumar Srivastava, Jan Koutnik Jurgen Schmidhuber. Recurrent highway networks. In *arXiv preprint arXiv:1607.03474*, 2017. URL https://arxiv.org/pdf/1607.03474.pdf.

Kim, Yoon. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL http://arxiv.org/abs/1408.5882.

Li, Yuhua, McLean, David, Bandar, Zuhair A., O'Shea, James D., and Crockett, Keeley. Sentence similarity based on semantic nets and corpus statistics. *IEEE Trans. on Knowl. and Data Eng.*, 18(8):1138–1150, August 2006. ISSN 1041-4347. doi: 10.1109/TKDE.2006.130. URL http://dx.doi.org/10.1109/TKDE.2006.130.

Mueller, Jonas and Thyagarajan, Aditya. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pp. 2786–2792. AAAI Press, 2016. URL http://dl.acm.org/citation.cfm?id=3016100.3016291.

Rocktäschel, Tim, Grefenstette, Edward, Hermann, Karl Moritz, Kočiský, Tomáš, and Blunsom, Phil. Reasoning about entailment with neural attention. In *arXiv preprint arXiv:1509.06664*, 2015. URL http://arxiv.org/abs/1509.06664.

Shankar Iyer, Nikhil Dandekar and Csernai, Kornl. First quora dataset release: Question pairs. *Data@Quora*, 24 Jan 2017. URL https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs.

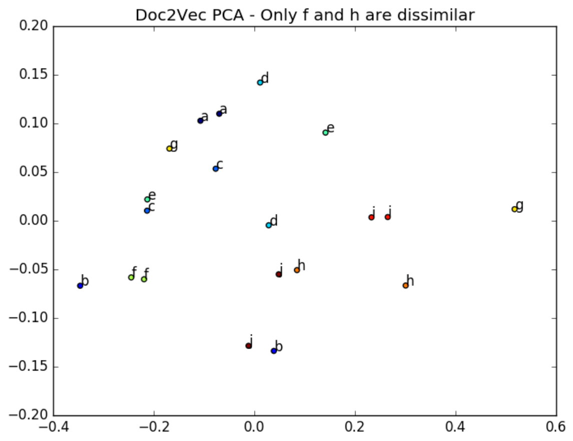

Figure 1. Sentence Embeddings: only f and h are dissimilar

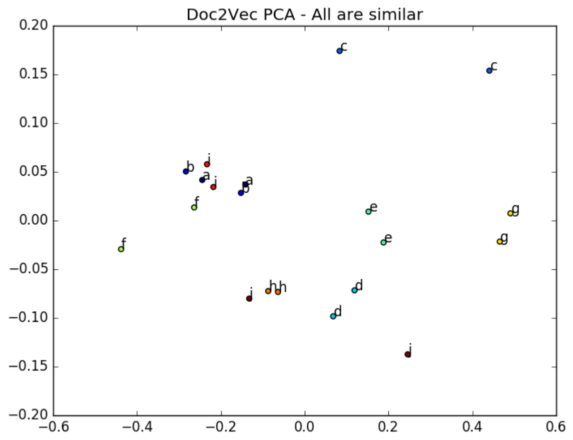

Figure 2. Sentence Embeddings: All are similar

8. Appendix

8.1. TSNE Plot for Attention

Question 1:
Q1: Should I have a hair transplant at age 24? How much would it cost?
Q2: How much cost does hair transplant require?
Result : Duplicate

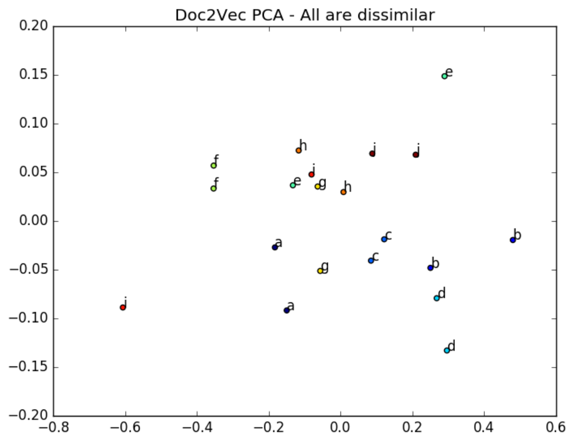

Figure 3. Sentence Embeddings: All are dissimilar

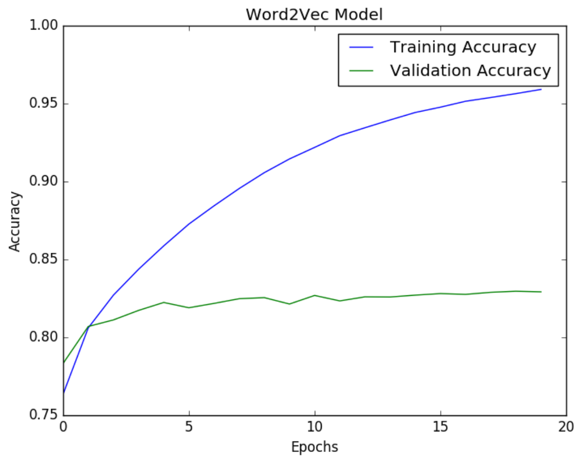

Figure 4. Word2vec plot

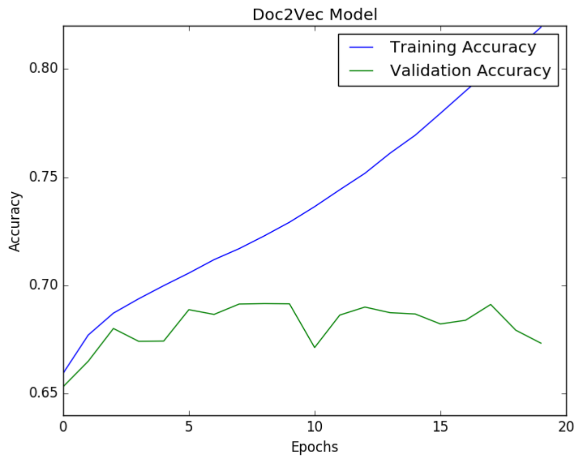

Figure 5. Doc2vec plot

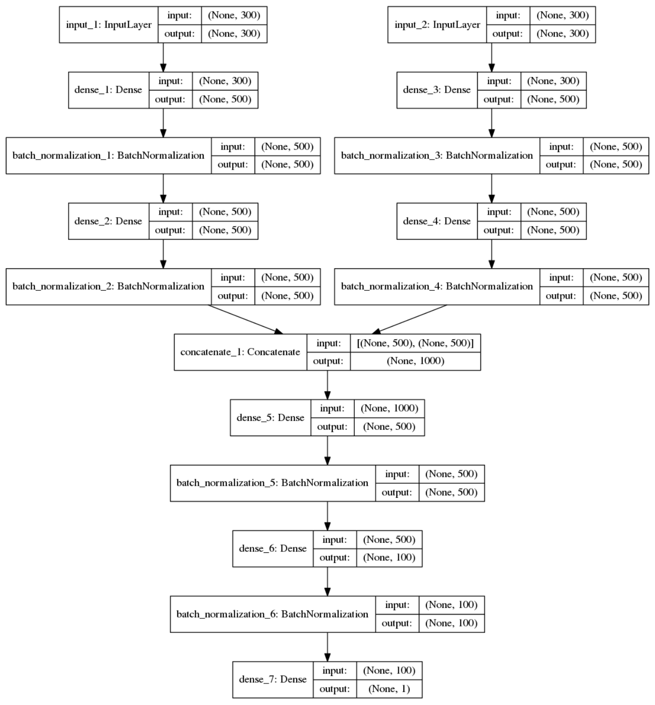

Figure 6. LSTM with Sentence Embeddings (1363101 parameters)

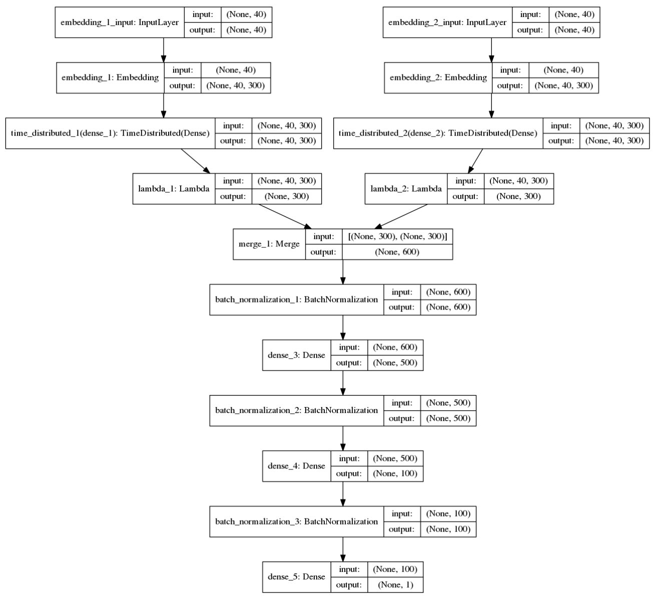

Figure 7. LSTM with Word Embeddings (536101 parameters)

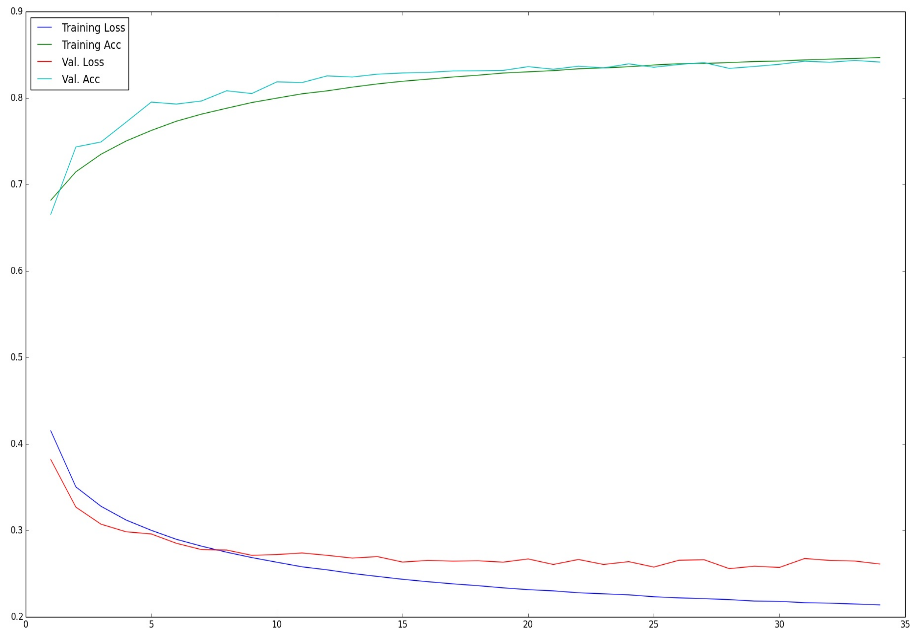

Figure 8. Plot of losses and accuracies v/s epochs of our best model

Question 2:
Q1: How do I become a data scientist in Malaysia?How can I become a data scientist?
Q2: How can I become a data scientist?
Result : Duplicate

Question 3:
Q1: How can I speak English fluently and banking?
Q2: How can I speak english naturally and?
Result : Duplicate

Question 4:
Q1: I wanna tell a girl how much I all love her?
Q2: I fell in love with a unknown girl. place can I tell her that I love her?
Result : Duplicate

Question 5:
Q1: How is CS taught at BITS?
Q2: How is programming taught used in BITS Pilani?
Result : Duplicate

Question 6:
Q1: Is it poisonous to eat a betta best?
Q2: Can I have learn after eating fish?
Result : Not Duplicate

Question 7:
Q1: Vit vs nirma?
Q2: Why do you create a blog?
Result : Not Duplicate

Conclusion: As we can see from the t−SNE plot 11 duplicate question pairs have much lesser intra−pair distance compared to non−duplicate question pairs.

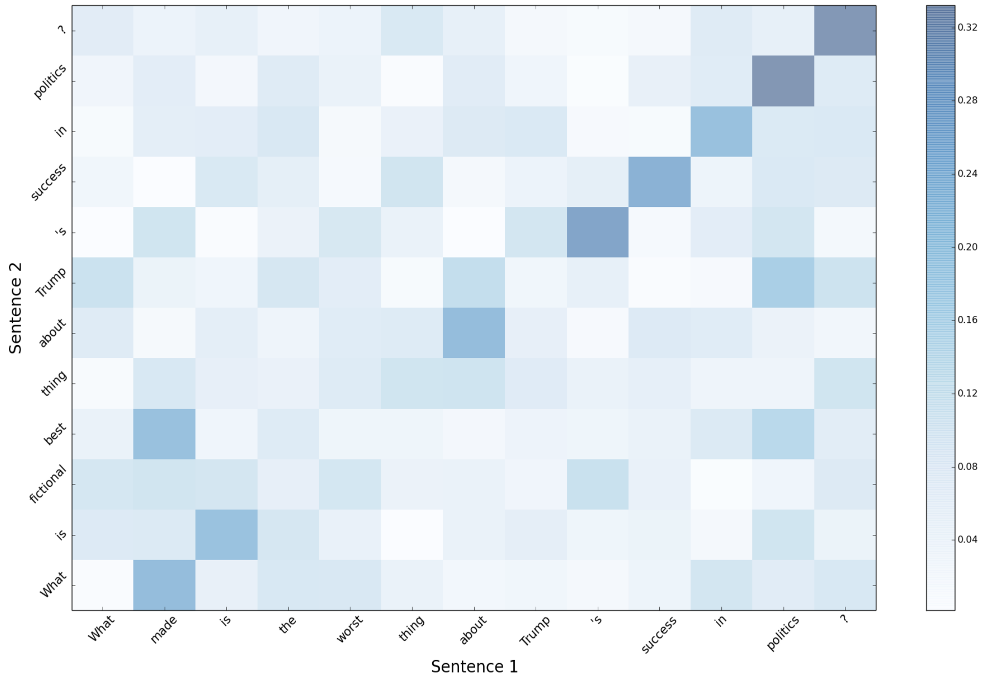

Figure 9. Attention Visualization (Word by Word)

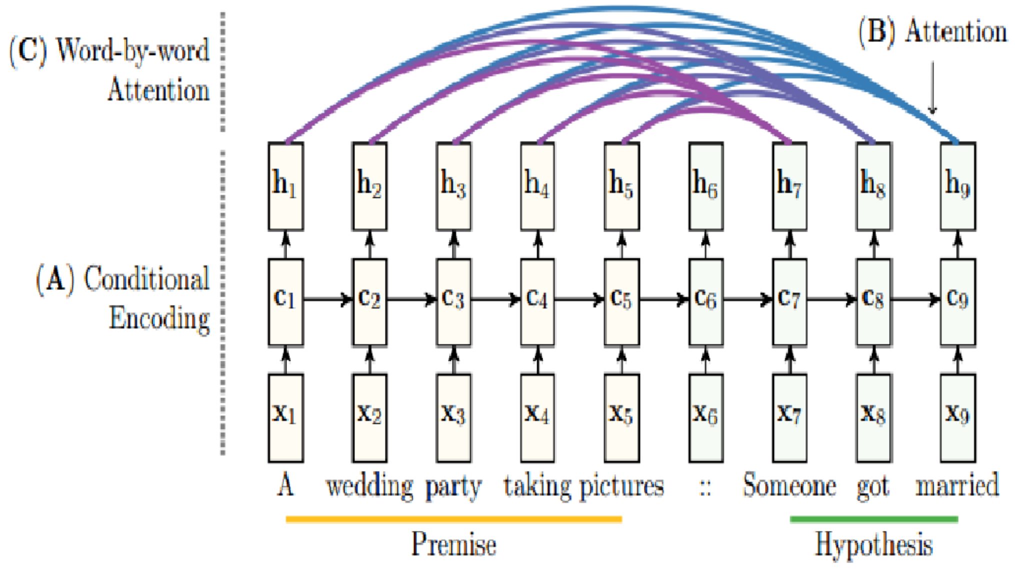
(C) Word-by-word Attention
(B) Attention
(A) Conditional Encoding
h1 h2 h3 h4 h5 h6 h7 h8 h9
c1 c2 c3 c4 c5 c6 c7 c8 c9
x1 x2 x3 x4 x5 x6 x7 x8 x9
A wedding party taking pictures :: Someone got married
Premise
Hypothesis

Figure 10. The Entailment Attention Architecture

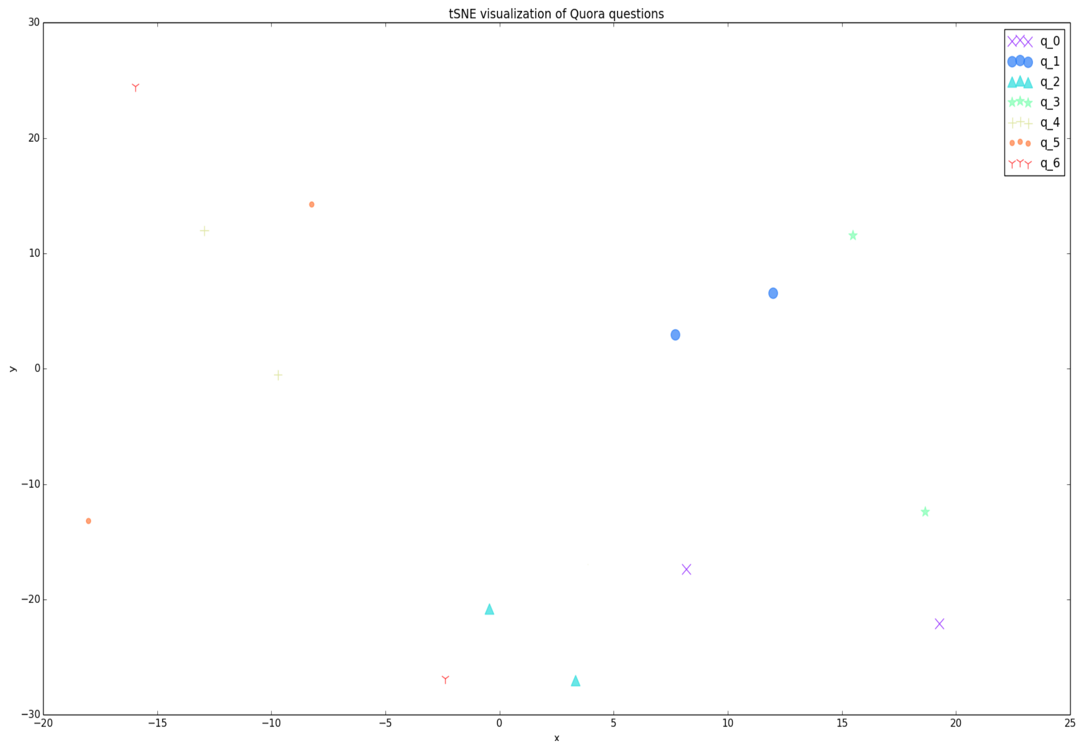

Figure 11. t-SNE plot achieved using Attention