INFORMATION RETRIEVAL PROJECT

Answering E-commerce Product Questions using existing QA Collections and Reviews

January 2020 - May 2020

SUBMITTED BY

Anurag Malakar (19BM6JP06)

Mimansi Agarwal (19BM6JP15)

Gaurav (19BM6JP40)

Bhushan Jayant Chaudhary (19BM6JP59)

UNDER THE GUIDANCE OF

Prof. Pawan Goyal
Prof. Saptarshi Ghosh

Mentor: Kalyani Roy

INDEX

Problem Statement

Create a Deep Learning Model that can automatically respond to new product-specific question by leveraging the existing Question-Answers and Reviews.

We have created a framework where the user can input a Question corresponding to a specific product. Our model can return relevant reviews corresponding to the entered question. It will also compare the new question with the existing Question-Answer Pairs to check for relevance and return the best Question-Answer Pair against a threshold similarity value.

Dataset Description

We chose the dataset created by McAuley and Yang in 2016. They collected reviews, questions, and answers by scraping the product pages of Amazon.com, for the period of May 1996 to July 2014, spanning 17 categories of products including Electronics, Baby Products, Video Games, Home and Kitchen, etc.

The total number of Questions in the dataset is 923k and corresponding to them are 3.6 Million Answers across 156k products. The total number of reviews across all the products are 14 Million. We have chosen the **Baby Products** Category for our analysis.

There was a total of 21996 Questions and 82034 Answers across 3752 Products, and there are 915446 Reviews. We have trained and tested our model on a subset of the dataset.

Attribute	Total	Training Set	Test Set
Products	126	120	7
Questions	737	586	151
Answers	2731	2201	---
Reviews	27573	25700	2574

Although the Dataset was provided in a *.JSON* file, the formatting was inconsistent, and we couldn't extract the data using the features of the file. We created the Training and Test Excel files by extracting the data which we loaded in a *.TXT* format. For consistency, we mapped every Question and Review to a unique Product ID, and each Answer was mapped to a unique Question.

Dataset Preparation

One major challenge with our Dataset was the absence of True labels, i.e**.,** a list of relevant reviews for each question. This is our first task as mentioned in the problem statement, and we applied **Cosine Similarity, BM25, and BERT** to generate the same. Around 30 questions were sampled randomly for manual quality assessment, and BERT was found to return the most relevant reviews.

Therefore, we created the ground labels for our **Training Set** by using the state of the art bert-base sentence transformer. BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained Transformer Encoder stack. bert-base has 12 encoder layers along with 768 feedforward-networks and 12 attention heads. BERT takes a sequence of words as input which keep flowing up the stack. Each layer applies self-attention and passes its results through a feed-forward network, and then delivers it to the next encoder.

For each question, the dataset had a list of answers that had been submitted by users. First, we found out the most appropriate answer for each question using bert, and then we concatenated the question and the answer to ensure that the correct context is picked up while assessing the relevance of reviews with the question. We chose the most appropriate answer to get a unique list of relevant reviews to every question.

Thereafter, all the reviews belonging to the product for which the question had been asked were compared with the question-answer pair to find out the similarity using spatial cosine distance.

Each review received a similarity score, and the 5 most relevant reviews were incorporated into the training dataset as being relevant. Thus, for every question we had 5 good reviews, and as a part of the data preparation process, 5 bad reviews were also chosen by randomly sampling from the remaining review database.

Thus, each training instance became a question, one good review, and a corresponding bad review. This was fed into our models as one training instance. We have trained the model on **586** questions across **120** Products, all from the Baby product category.

Post creation of the Training Dataset, the **Validation Dataset** was created by manually labelling **151** Questions, by assessing the relevance of each review belonging to the product for which the question has been asked. These questions were used to assess the model performance.

For each question, we took the top 5 manually labelled reviews corresponding to its Product ID. The remining reviews for that Product ID were labelled as the bad reviews for that question.

Creation of Input Files for the Model

1) **Vocabulary File -** This file will contain **all the words** which are a part of the questions and reviews which are present in the training and testing set of our model.

2) **Embedding File -** We have prepared the model to generate the embedding of words in the corpus. We have used the **Word2vec** model for the same purpose. Its working is explained in the subsequent section. The Embedding file will contain embeddings of all words in the identical order as present in the vocabulary file. This file is the mapping of embedding to words in the vocabulary file that have been generated by the Word2vec model.

3) **train.pkl File -** This file is the **list of dictionaries** where each element is the dictionary which contains the label encoding of the question with respect to the vocabulary file and the index of one or more most relevant reviews present in the ***answers_amazon.pkl*** file.

4) ***answers_amazon*.pkl File -** This file contains a dictionary where the key of the dictionary is the review number and its value is the review label-encoded as per the indexes of individual words present in the ***Vocabulary file***. This file is mapped to the **train*.pkl*** to find the encodings of the relevant reviews, post which the embeddings of each individual word is picked up from the embedding file and provided as input to the model.

5) **dev.pkl File -** This file is used to validate the model performance. It is a list of dictionaries, where each dictionary has **3 key-value** pairs. The first key is the question ID and the corresponding value is its label encoding with respect to the ***Vocabulary file***. The second key is the good answer which is the most relevant review to that question and its value is the index of that review present in the ***answers_amazon.pkl*** file. The third key contains bad answers, i.e., the remaining reviews present for the Product ID corresponding to that question.

Embedding Model

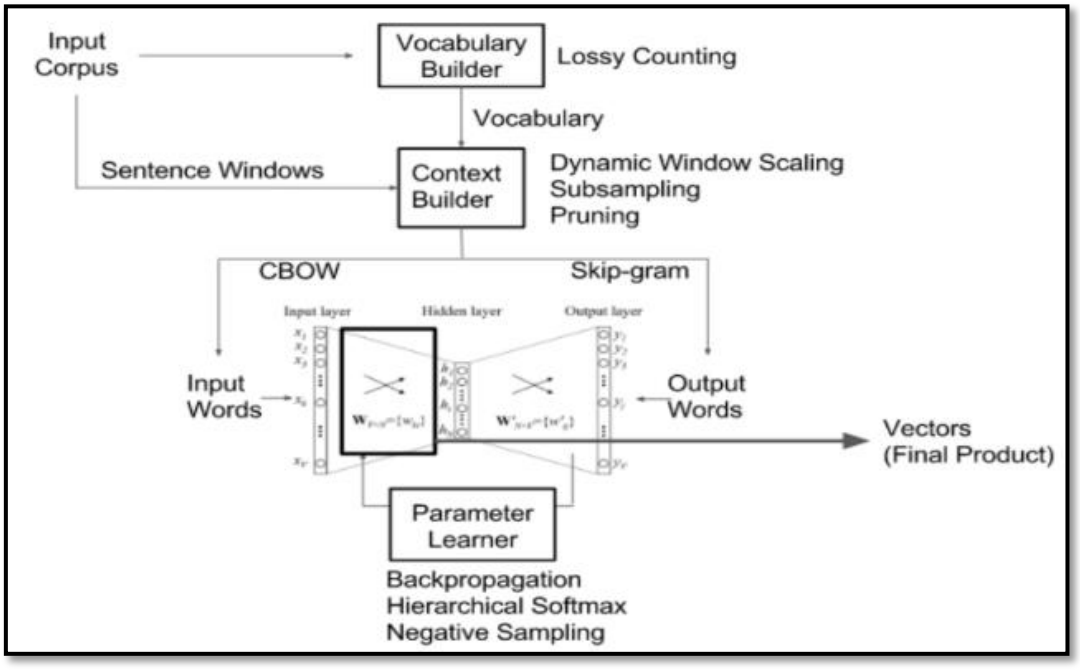

Figure 1. Word2vec embedding model

For our model we have taken our input corpus as all the questions and reviews which are present in the training data. We have used the ***genism library*** to build this model. **The *Vocabulary file*** which is mentioned above is the byproduct of the above model as we can see in the figure first step of this model is to build vocabulary.

There are two main training algorithms for word2vec, one is the continuous bag of words (CBOW), another is called skip-gram. The major difference between these two methods is that CBOW is using context to predict a target word while skip-gram is using a word to predict a target context. Generally, the skip-gram method can have a better performance compared with CBOW method, for it can capture two semantics for a single word. For instance, it will have two vector representations for Apple, one for the company and another for the fruit. Hence, we had used skip-gram method to calculate embedding for our task.

Modelling

Various frameworks were tried to achieve the given task.

The basic framework is to create the embeddings of questions and answers based on bidirectional long short-term memory (Bi-LSTM) models and measure their closeness by cosine similarity. We further extend this basic model in two directions. One direction is to define a more composite representation for questions and answers by combining convolutional neural network with the basic framework. The other direction is to utilize a simple but efficient attention mechanism in order to generate the answer representation according to the question context.

Model Input Structure -

For training –

- Question
- Good Answer (Most relevant review to question being asked)
- Bad Answer (Randomly chosen from corpus apart from the relevant reviews)

For running the model –

- Question
- Product Id (ID is used to fetch all reviews corresponding to predict)

Output of Model - Most relevant review along with similar question-answer pair if available.

Loss function –

Each training item consists of a question, the correct answer and a distractor (an incorrect answer). A prominent choice is using a shifted hinge loss, designating that the correct answer must have a higher score than the distractor by at least a certain margin M, where the score is based on the similarity to the question.

Loss Function = MAX{ 0 , M – good_answer_similarity + bad_answer_similarity)

The above expression has a zero loss if the correct answer has a score higher than the distractor by at least a margin M, and the loss linearly increases in the score difference between the correct answer and the distractor. Any reasonable neural network design can be used to build a working answer-selection systems using the above approach; however, the network design can have a big impact on the system's accuracy.

Various tried models are detailed explained below:

1) Base Model – BiLSTM -

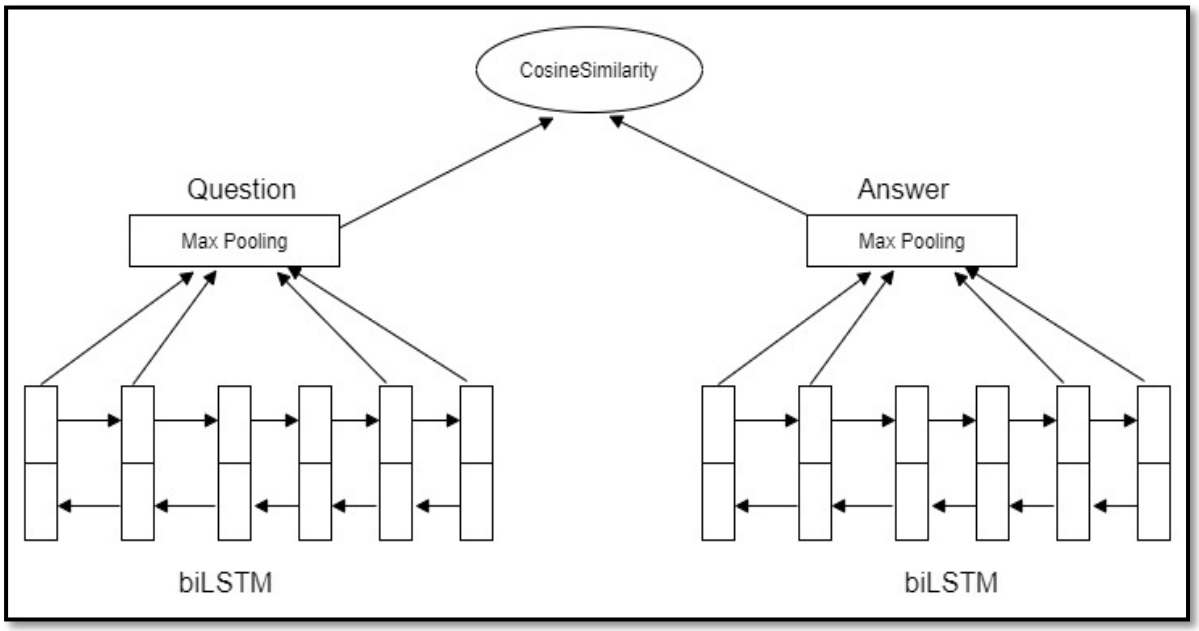

Figure 2. BiLSTM model architecture

Single direction LSTMs suffer a weakness of not utilizing the contextual information from the future tokens. Bidirectional LSTM utilizes both the previous and future context by processing the sequence on two directions, and generate two independent sequences of LSTM output vectors. For the same reason we had used the BiLSTM.

There are three simple ways to generate representations for questions and answers based on the word-level Bi-LSTM outputs: (1) Average pooling; (2) max pooling; (3) the concatenation of the last vectors on both directions. After experimenting it has been observed that mean max pooling gives better results hence we have used the same.

Finally, from preliminary experiments, we observe that the architectures, in which both question and answer sides share the same network parameters, is significantly better than the one that the question and answer sides own their own parameters separately, and converges much faster. For the same reason we have used the same BiLSTM layer on both question and answer side.

2) BiLSTM with Attention -

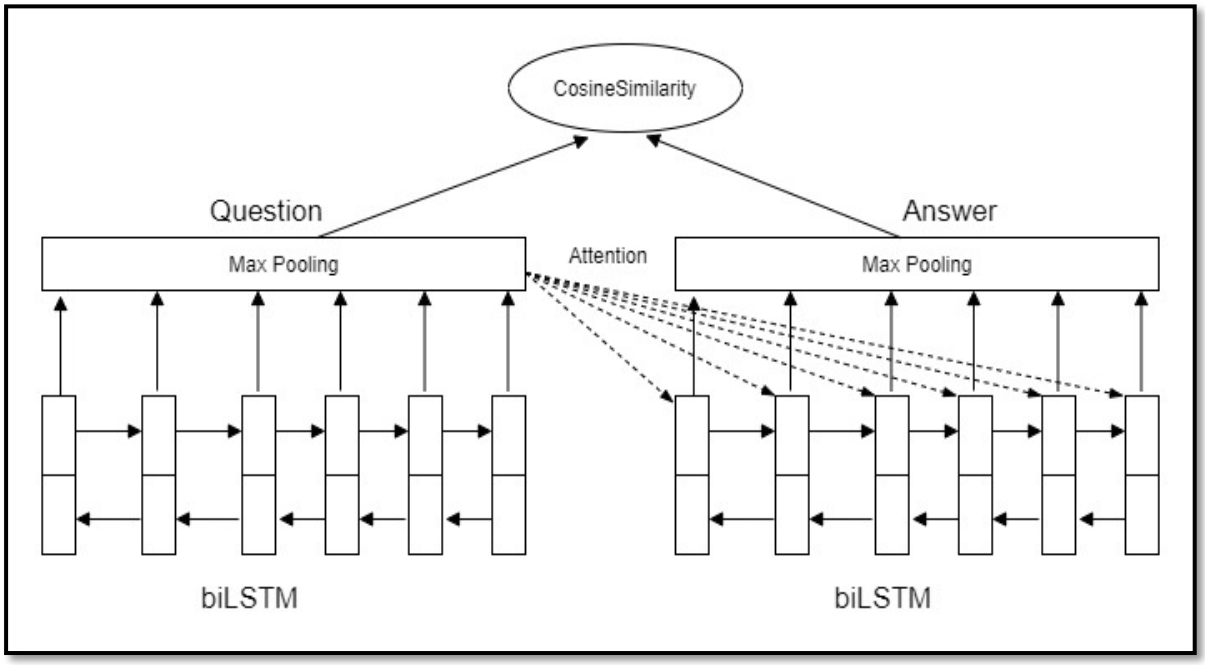

Figure 3. BiLSTM with Attention model architecture

In the previous subsection, we described one extension from the basic model, which targets at providing more composite embeddings for questions and answers respectively. In this subsection, we investigate an extension from another perspective. Instead of generating QA representation independently, we leverage a simple attention model for the answer vector generation based on questions.

The fixed width of hidden vectors becomes a bottleneck, when the bidirectional LSTM models must propagate dependencies over long distances over the questions and answers. An attention mechanism are used to alleviate this weakness by dynamically aligning the more informative parts of answers to the questions. This strategy has been used in many other natural language processing tasks, such as machine translation. Hence we tried to use this technique in this model which generates good result than baseline model.

Specifically, given the output vector of BiLSTM on the answer side at time **step t**, ha(t), and the
question embedding, **Oq**, the updated vector **ha(t)** for each answer token are formulated below.

$$M_{a,q(t)} = \tanh(W1 * h_{a(t)} + W2 * O_q)$$
$$S_{a,q(t)} = \exp(W3 * M_{a,q(t)})$$
$$h_{a(t)} = h_{a(t)} * S_{a,q(t)}$$

Where W1, W2, W3 are attention parameter.

3) BiLSTM plus CNN -

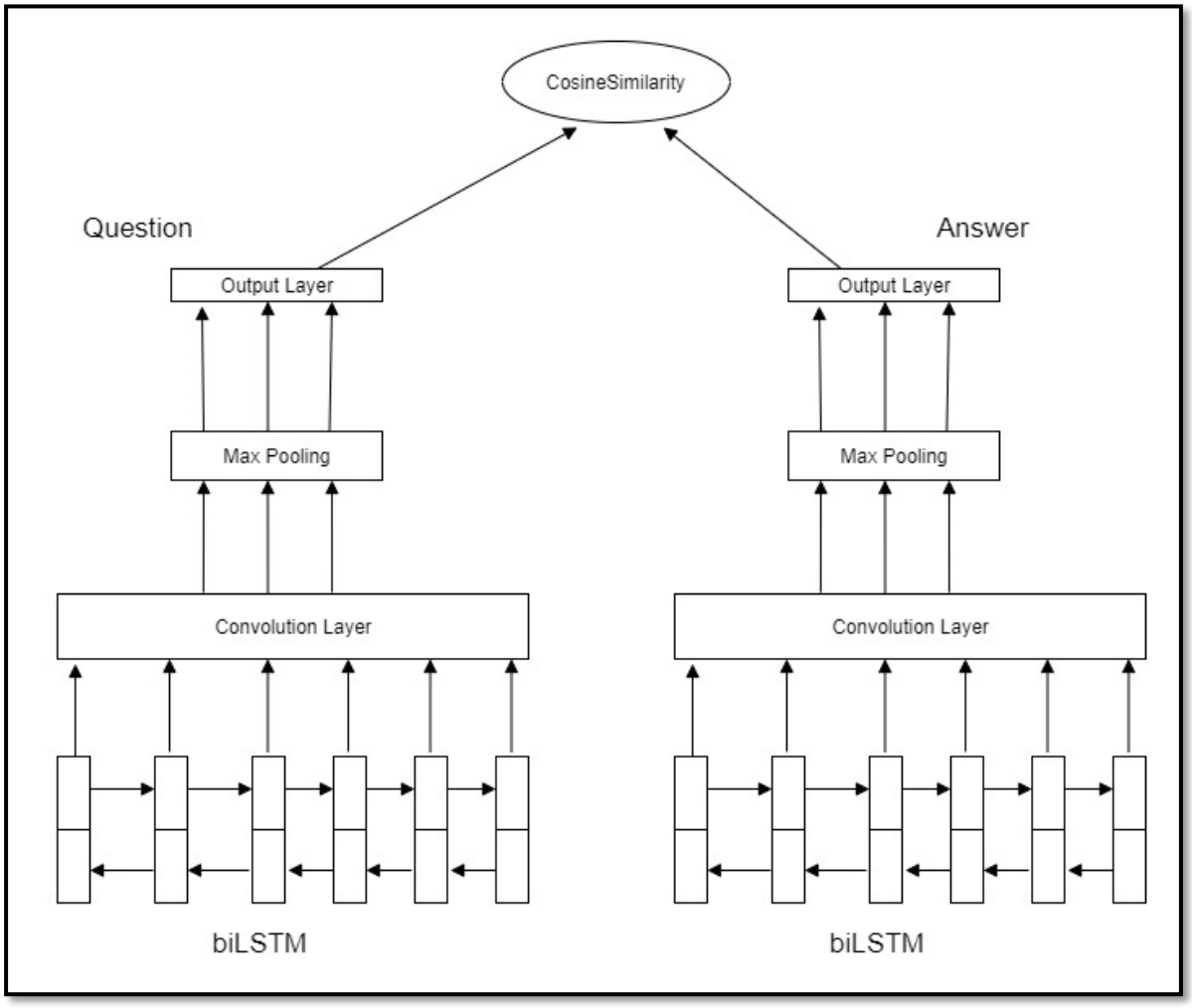

Figure 4. BiLSTM plus CNN model architecture

In the previous models, we generate the question and answer representations only by simple operations, such as max or mean pooling. In this model, we resort to a CNN structure built on the outputs of BiLSTM, in order to give a more composite representation of questions and answers.

The intuition of this model is, instead of evenly considering the lexical information of each token as the previous models, we emphasize on certain parts of the answer, such that QA-LSTM/CNN can more effectively differentiate the ground truths and incorrect answers.

4) BiLSTM plus CNN with Attention -

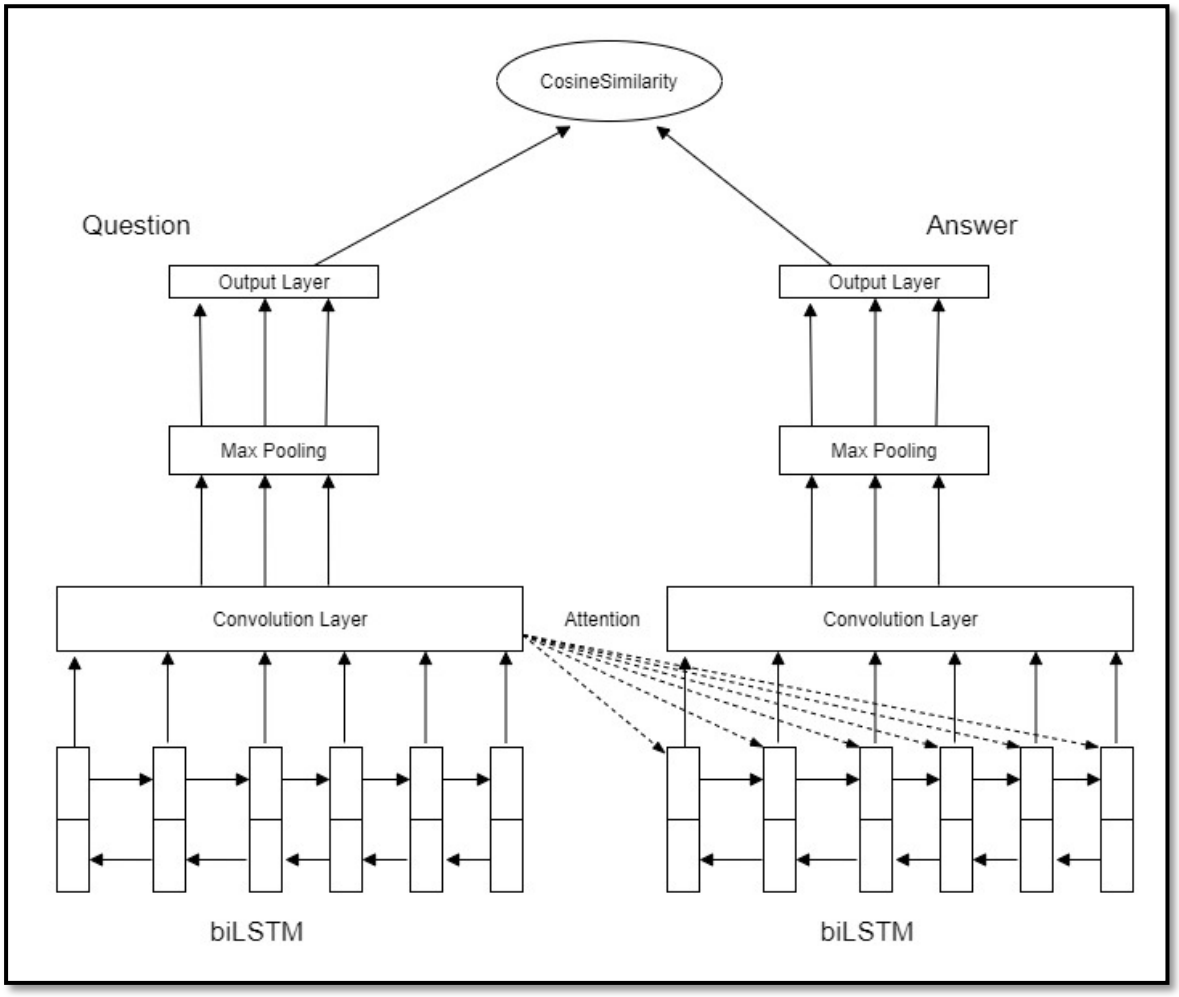

Figure 5. BiLSTM plus CNN with Attention model architecture

The models introduced previously are combined in a simple manner. First, the BiLSTM hidden vectors of answers **ha(t)** are multiplied by **Sa,q(t)**, which is computed from the question average pooling vectors **Oq**, and updated to vector **ha(t),** as shown in model 2 which QA LSTM with attention model . Then, the original question and updated answer hidden vectors serve as inputs of CNN structure respectively, such that the question context can be used to evaluate the softmax weights of the input of CNN. From the experiments, we observe that the QA-LSTM/CNN with attention can outperform the baselines on our given dataset.

Results

To validate the model performance, **Precision** and **Mean Reciprocal Rank** (MRR) was computed for the 150 questions present in the Validation set. For each question, all reviews particular to product in consideration were provided as input, of which there was good reviews (**as determined by manual labelling**) and other are bad reviews chosen from the rest of the review database.

Model	Precision	MRR
BiLSTM	0.45	0.52
BiLSTM with Attention	0.52	0.58
BiLSTM plus CNN with Attention	0.58	0.63
BiLSTM plus CNN	0.63	0.68

From the above results it is evident that models with CNN layers are performing better than model without CNN which is quite intuitive but for this dataset we had observed that model with attention mechanism performing somewhat poorly than the models without it which is the grey area for us. The potential reason can be less amount of data used in training of models.

Screenshot of Model Outputs:

Figure 6. Model Outputs

In the above screenshot we can see that the model had returned the relevant review in response to question askes while there was no similar question asked previously hence there was no similar question in our database hence it haven't give similar question and response however in the screenshot below we can see that as the similar question was previously asked and was present in our database hence models had returned that question too along with relevant review in response to question asked.

Conclusion

In this project, we employed a BiLSTM based deep learning framework for answering new questions using existing question-answer pairs and reviews. Convolutional neural networks and attention layers were added to our basic framework. Our results show that **BiLSTM plus CNN** gave us the **highest precision** on the validation dataset.

In the future, the performance of these models can be significantly improved if they are trained on a much larger dataset. The model can be extended to pick up snippets of text from the reviews that best answers the question asked by the user.