Question Answer Summarization

Nidhi Kadkol
University of Texas at Austin
nidhik@cs.utexas.edu

Shivangi Mahto
University of Texas at Austin
shivangi@cs.utexas.edu

Abstract

In this work, we present an automatic question-answering system for online question-answering websites like Quora, Stack Overflow and Yahoo. A lot of information on these sites tend to be repeated, as different users ask duplicate questions. Our goal is to answer a new question by summarizing the answers to the previously-asked similar questions. For fetching similar questions from the database, we implemented a Siamese-Recurrent Architecture which is trained to identify semantically similar question pairs. We extended the idea for question recognition by implementing a classifier. It helped us to achieve a best accuracy of 80% on the development dataset. For the task of answer summarization, we explored attention-based encoder-decoder model with coverage mechanism. We also present a novel question-centric summarization where question embeddings are taken as input to the decoder to provide contextual support.

1 Introduction

There a number of online question-answering sites such as Quora, Stack Overflow, and Yahoo where a community of people increase each other's knowledge by providing information on a variety of topics. A lot of information on these sites tends to be repeated, as different users ask similar questions without knowing that one or more people have already provided answers to previously asked duplicate questions before. This leads to a number of redundant questions, each with an independent answer and all the answers having mostly similar content. Our goal is to create a system which summarizes all answers to duplicate questions and provide that as the answer to a new duplicate question, thus giving the user the most information possible without any redundancy.

We break down this task into three stages: identifying if a pair of questions are duplicates of each other, grouping similar questions into classes, and summarizing answers to all questions within a class. Identifying if a pair of questions is similar requires capturing their semantic meaning in the form of a vector. We use an LSTM to encode a question into a fixed dimensional vector and make use of the siamese recurrent architecture [1] to predict the similarity between 2 questions. For the next stage, we use the trained encoder from the previous step to encode questions from the dataset. We then pass this through a neural network to classify it into one out of 1180 classes. For the final step, we summarize all the answers of the questions belonging to a class using a sequence to sequence attention based summarizer. We also implemented the coverage [2] mechanism to avoid repeating the same text. We also present a question-centric summarization which uses semantic information present in the question to provide contextual support to the decoder.

The report is organized as follows: Section 2 provides an overview of the related work in the field of semantic similarity among texts and article summarization. Section 3 presents our models on question recognition using semantic similarity. In section 4, we presents answer summarization models and the ideas we explored during experimentation. In section 5, experimental setup and results have been discussed. Section 6 concludes the work presented in this project.

2 Related Work

Capturing semantic information from natural language has been a subject of much research. Early semantic parsers focused on database-query tasks [3] which converted the input query to a logical form that can be parsed by a machine to return results. The work of Zettlemoyer and Collins [4] focuses on mapping natural language sentences to lambda-calculus encodings. These methods require the data to be annotated with the logical or lambda-calculus form for every example, which is infeasible for training large datasets. Hierarchical feature-based translation [5] has been explored to convert more complex user commands into formal commands. Berant *et al*. [6] train a semantic parser that learns from question-answer pairs. These methods also only focus on converting sentences that are in the form of fact-based queries to logical forms. Conditional Random Fields [7] focus on segmenting and labelling sequence data, but do nothing to capture or understand the meanings of these segments.

State of the art models use deep neural networks for semantic encoding of sentences. Yoon Kim [8] demonstated the effectiveness of convolutional neural networks in sentence classification. Collobert and Weston [9] used a deep learning architecture to make semantic predictions such as similar meaning words as well as sentence structure predictions such as part-of-speech tags. Chiu and Nichols [10] combined bidirectional LSTMs with CNNs to achieve named entity recognition. Using such architectures eliminates the need for feature engineering.

LSTMs with CRFs have produced state of the art NER systems [11], but more importantly, their application in sequence to sequence prediction of sentences [12], [13] is extremely useful for semantic encoding. Encoder-decoder models have also been used in generating regular expressions from natural language [14].

Summarization tasks in natural language processing also make extensive use of LSTMs. Abstractive summarization can be effectively carried out using pointer-generator networks [15] which rely on using a copying mechanism [16] in sequence to sequence models. Using methods such as attention [17], [18], [19] and coverage [2] make the summarizer more robust and close to the output of a human summarizer. In our work, we make use of LSTMs for their memory-retaining ability and use them for identifying semantically similar questions. Training an LSTM network gives us a pre-trained encoder which we use to train our pointer-generator network for answer summarization.

3 Question Recognition Using Semantic Similarity

To build a question-answering system, the first goal was to train a model to recognize semantically similar questions so that their corresponding answers can be grouped and summarized. We proceeded in 2 stages, first detecting semantic similarity between question pairs and then using a feed-forward network to group all duplicate questions into one class.

3.1 Question Pair Similarity

In this section, we describe the process of detecting pairs of duplicate questions. We use the public Quora Question Pairs dataset [20]. This dataset contains 0.4 million pairs of questions, labelled 1 if the questions are duplicates of each other and 0 otherwise. We discarded statistical methods of predicting semantic similarity as semantic relatedness between 2 sentences goes beyond quantitative measures such as number of words in common between the sentences. Our model tries to learn the semantic representation of a sentence and compare it with that of another. We adopt a siamese recurrent architecture [1] for this purpose.

The dataset is imbalanced, containing 63% question pairs that are not duplicates of each other. Since the total number of examples itself is so large, we only took the first 149k pairs from each set of positive and negative examples. Our training was as follows: we encode each question from an example pair into a 100-dimensional vector using an LSTM, then compute the distance between these two representations using a Manhattan based distance metric as shown in figure 1. The output is a real-valued integer between 0 and 1, with higher values indicating more similarity.

We used Binary Cross Entropy loss with the Adam optimizer and an initial learning rate of 1e-3. We took 15k examples of the positive and negative label each to get a development set with 30k examples and a training set with 0.2M examples. In each training iteration, we passed in a positive

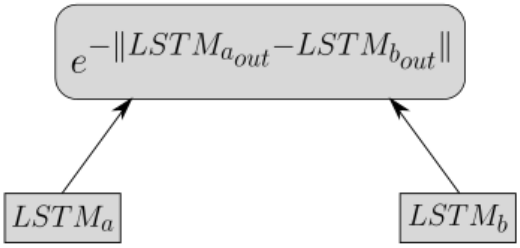

Figure 1: Siamese architecture for calculating similarity between questions.

example followed by a negative example. We initialized our word embeddings with pre-trained 300-dimensional GloVe vector embeddings [21]. Some samples in the dataset did not have one of the questions, so we skipped those examples. We trained our model for 5 epochs and got a best development accuracy of **77.9%**. Due to the large amount of training data, it was not necessary to train the model for further epochs. We initialized the weights of the encoder uniformly at first, and then with Xavier initialization. The results with these 2 initialization methods did not differ much. Table 1 shows the mean loss and development accuracy for each epoch.

Epoch	Uniform weight initialization		Xavier weight initialization	
	Avg Loss	Dev Acc	Avg Loss	Dev Acc
1	0.725	61.9	0.844	62.2
2	0.657	67.7	0.678	64.5
3	0.629	68.5	0.633	67.8
4	0.624	73.2	0.631	**72.6**
5	0.613	**77.9**	0.619	70.7

Table 1: Average loss and development set accuracy (%) in question pair similarity task

3.2 Categorization of Duplicate Questions

In the previous section, we focused on finding if a pair of questions are duplicates. We now extend this to categorizing all duplicate questions into a single group. We treat this as a multi-class classification problem, where each class represents a set of semantically identical questions.

We use the Text REtrieval Conference (TREC) 2017 dataset [22], which contains a set of Yahoo questions and answers collected by NIST. The data is structured as follows - each example comprises an original Yahoo question along with its question id, and a list of paraphrased questions for the original question. In addition, there are multiple answers to the original question. The paraphrased questions have been created by human assessors, who interpret and paraphrase the original question as well as rate the answers to the original question based on relevance and meaningfulness. The answers and their ratings are not used here, but will be used in the next stage as described in section 4. We assign each set of (original question) + (all its paraphrased questions) a class id, and we use this to train our model to classify questions. In other words, we train our model to identify duplicate questions by teaching it to classify paraphrased questions into the same class.

To build a system for this, we already have a pre-trained Siamese Recurrent Architecture (SRA) encoder from the previous stage which we can use for encoding questions to compare their interpretations. The TREC dataset has 1180 original questions, each with its own list of paraphrased questions. However, the number of paraphrased questions per class is far too small - ranging from 1 to 4 paraphrases per original question. To overcome this issue, we perform data augmentation by adding permutations of the words in the question as extra examples for a class. We reason that this should work favourably, as we are also teaching the network that the order of words in a sentence does not matter as much as their actual meaning when it comes to detecting duplicates. We also

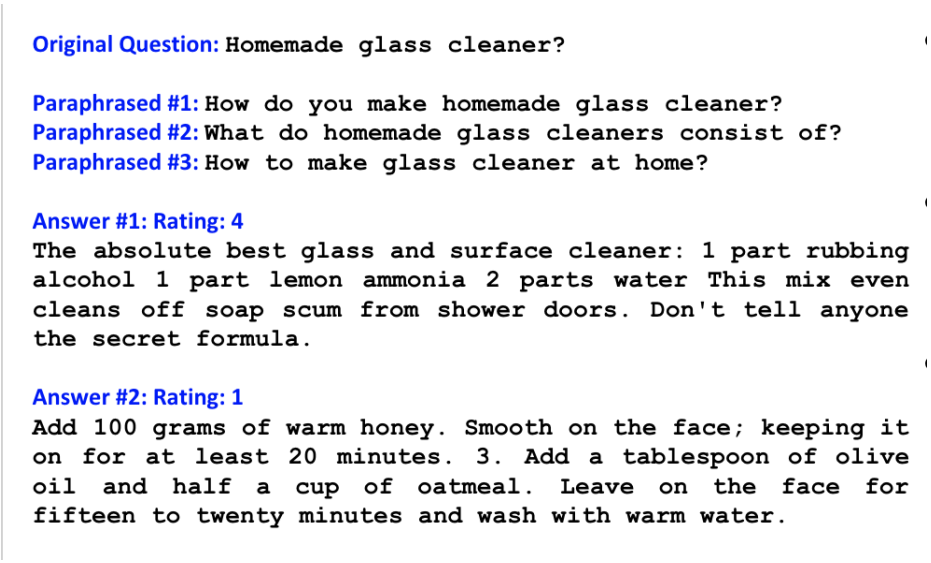

Figure 2: Question-answer dataset with original questions, corresponding paraphrased questions and multiple answers from users.

discard examples in which there is only one other paraphrased question. Some examples have no original question, so we discard those examples as well.

For training, we add an integer class label to each question based on which set it belongs to. For each class, we select at random one of the questions (original or paraphrased) to go into the development set, so the size of the development set is equal to the number of classes. We then add permutations of each question to the training dataset. We used the pre-trained SRA encoder to get the vector representations of the words and fine-tuned it over the course of this training. We pass the encoded question into a 3 layer neural network, with all the layers being fully connected and having 400 neurons each. We initialized the parameters of this network using Xavier initialization, and added a ReLU non-linearity after each layer. We used the Adam optimizer with an initial learning rate of 1e-3, and used cross entropy loss. The experimental results are shown in section 5.1.

4 Answer Summarization

For each original question in the database, we have a source document containing multiple independent answers as shown in figure 2. To provide an answer to a newly question which is a duplicate of previously asked questions, we prepare a summary of all the independent answers to these previously-asked similar questions. We implemented an attention-based sequence-to-sequence model for the summarization task. It receives a document of all the previous answers as input and outputs the summary.

4.1 Baseline Sequence-to-sequence Model

Our baseline model is an attention-based sequence-to-sequence encoder-decoder model as illustrated in figure 3. The model includes an encoder which takes in an input document and a decoder which predicts a summary of the document by taking the encoder output as context information.

The tokens of the document t_i are fed one-by-one into the encoder (a single-layer unidirectional LSTM), producing a sequence of encoder hidden states h_i. At each step t, the decoder (a single-layer unidirectional LSTM) receives the word embedding of the previous word w_t (while training, this is the previous word of the reference summary; at test time it is the previous word emitted by the decoder), and has decoder state d_t. The attention distribution is calculated as in Luong *et al.* [19]:

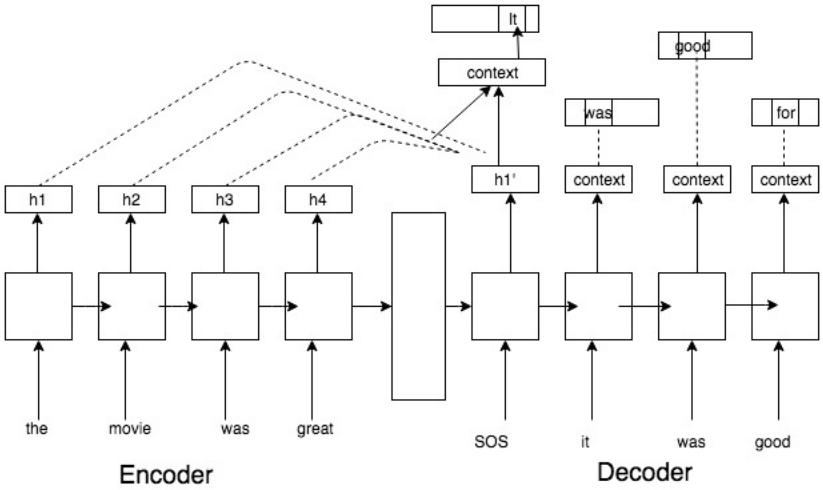

Figure 3: Attention-based encoder-decoder summarizer.

$$output, s_t = lstm(w_t, s_{t-1}) \tag{1}$$

$$e_i^t = h_i^T W_a d_t \tag{2}$$

$$a_i^t = softmax(e_i^t) \tag{3}$$

where W_a is learnable parameter. The attention is a kind of probability distribution over words in the document. It allows the decoder to pay attention to the words which might help to predict the next word. The attention distribution is used to produce a weighted sum over the encoder states, known as context vector, as follows:

$$h_t^* = \sum_i a_i^t h_i \tag{4}$$

The context vector is concatenated with the decoder state s_t and fed through a $tanh$ layer followed by a softmax layer to get a produce the vocabulary distribution:

$$P_{vocab}(w) = softmax(V_1(tanh(V_2([d_t, h_t^*]) + b_2)) + b_1) \tag{5}$$

The encoder-decoder model is trained end-to-end to minimize negative log likelihood over the vocabulary distribution for each document:

$$loss_t = -\log(P_{vocab}(w_t^*)) \tag{6}$$

$$Loss = -\sum_{t=0}^{T} loss_t \tag{7}$$

where $\log(P_{vocab}(w_t^*)$ is the log probability of the predicted token to be the gold token w^* at time instant t and T is the length of document. In our case, a document includes all the answers present in the database for a particular question set.

4.2 Coverage Mechanism

Our baseline model could learn basic sentence structure and was able to generate a summary. However, it would often tend to repeat phrases again and again as shown in figure 11. To avoid such repetition,

we adopted the coverage mechanism [2] from the Pointer-Generator model as presented by Abigail *et al*. [15]. A coverage vector c^t was maintained over attention weights for each time stamp t as:

$$c^t = \sum_{i=0}^{t-1} a^i \tag{8}$$

To incorporate its effect on attention mechanism, we recalculated attention energy e_i in equation (2) as follows:

$$e_i^t = h_i^T W_a d_t + W_c c_i^t \tag{9}$$

$$a_i^t = softmax(e_i^t) \tag{10}$$

where W_c is a learnable parameter vector. To penalize repeated attention to same location, we added a coverage loss $covloss_t$ as:

$$covloss_t = \sum_i \min c_i^t, a_i^t \tag{11}$$

The final loss for a time stamp t during training the encoder-decoder model with coverage mechanism is as follows:

$$loss_t = -\log(P_{vocab}(w_t^*)) + \sum_i \min c_i^t, a_i^t \tag{12}$$

4.3 Question-centric Summarization

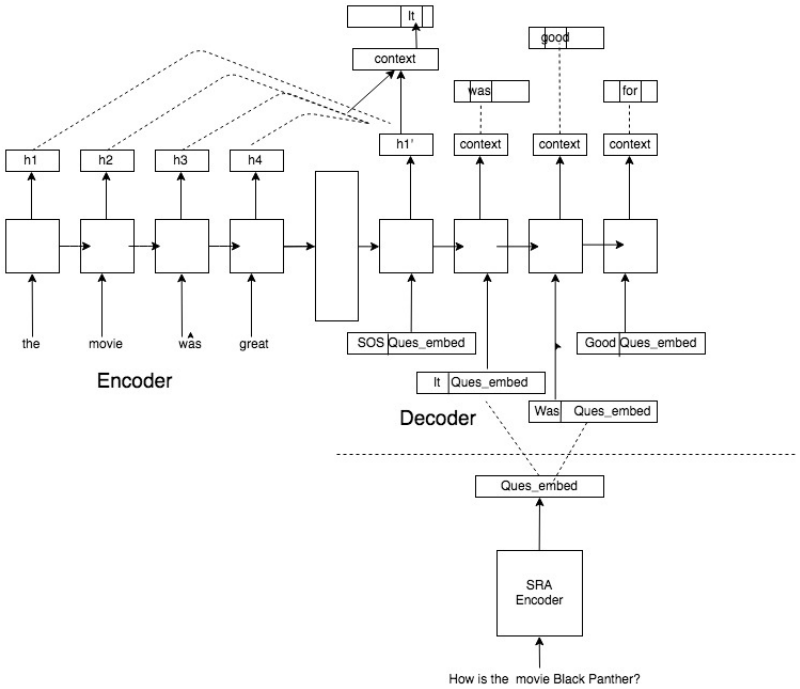

Figure 4: Question-centric attention-based encoder-decoder summarizer.

Unlike usual summarization tasks, our goal for summarization is to provide an answer to a specific question. The extracted summary should be directly related to the content of the question. However, the source document in our experiments has answers from real people which usually has a lot of irrelevant information for the model. This motivated us to provide a support vector to the decoder model, which can help the model connect with the gold summary contextually.

To incorporate contextual information present in the question, we concatenate encoded question vectors with the decoder input in each time stamp along with the word of the reference summary, as shown in figure 4. While training, the semantic information present in the question embedding will help the decoder to connect the gold summary with the input question. During test phase, this contextual information can guide the decoder to pick only relevant phrases from the source document.

Our encoder-decoder model is the same as our baseline model with the coverage mechanism as described above. For each question in the database, we extracted an embedding using the pre-trained SRA encoder. These embeddings contain semantic information of the question. At the time of training the encoder-decoder model, extracted embeddings of the associated question with the document is concatenated with the word embedding for the each time stamp. This changes equation 1 as follows:

$$q_{emb} = SRA_{enc}(q_{vec}) \tag{13}$$

$$output, s_t = lstm([w_t, q_{emb}], s_{t-1}) \tag{14}$$

SRA encoder weights are fixed during answer summarization training. Note that the question embedding is constant across all time stamps. Apart from the decoder input, the rest of the model is unchanged during training and test phase.

5 Experiments

5.1 Question Categorization

We carried out experiments in different settings on the TREC dataset with our set up as described in section 3.2. From figure 10, we see that the training loss for all of our experiments starts out high and then decreases uniformly to 0. We started with 50 question classes and 40 question examples per class in the training set as in figure 5. In figures 2 to 6, the training accuracy is blue and the development accuracy is red in the graphs. The training accuracy reaches close to 1 early on by epoch 4. The development accuracy settles to between 64% and 70%, reaching its maximum value of **74%** in epoch 10. We then decided to increase the size of the development set by adding a permutation of every development example, thus having 2 development examples per class. The total number of training examples was the same as before (2000). In figure 6, we see the same trend for the training data, while the development accuracy improves, reaching **80%** in epoch 9. This shows that the model has learnt to be invariant to permutations of the questions, as we intended. With the same settings before, we decided to increase the number of classes to 100, as shown in figure 7. The training accuracy begins to saturate slightly later by epoch 9, and the development accuracy shows slight jumping up and down over the epochs. We get a best development accuracy of **64%** after epoch 30 in this experiment. We decided to try increasing the number of training examples to 80 per class to see if it would reduce the number of epochs required for training. In figure 8, we see that the noise in the development accuracy reduces, but the overall performance does not improve much, with a best performance of **62%**. In our last experiment, we increased the number of classes to 200 with 40 training examples per class, and as we see in figure 9, the model learns very slowly, with the development accuracy not even reaching 50% after 13 epochs. We infer that this is because of the lack of data per class as well as the network not being big enough to adapt to so many classes.

5.2 Answer Summarization

5.2.1 Data Processing

For each original question in the TREC data set, the best rated answer by the human assessors was taken as the gold summary. All the other answers were concatenated together to produce a source document. After each answer, we inserted a token of "**" to mark the end of an answer within a document. Words in the document are added in the vocabulary if they have appeared more than twice across all the documents. This helped us to limit our vocabulary size to 19.9k, however, it resulted in a lot of Unknown ("UNK") words in our training data. Maximum length of the document was 1229

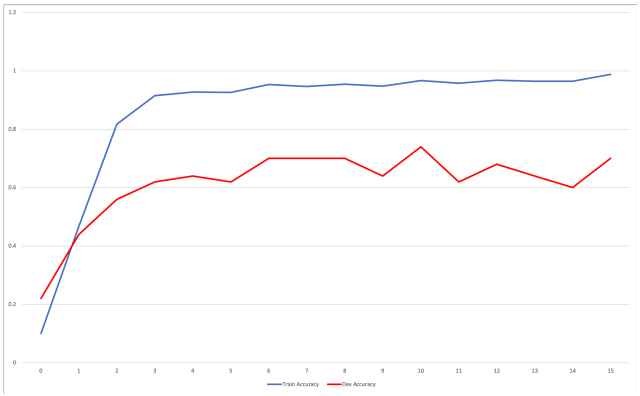

Figure 5: Train loss and development accuracy on TREC Question dataset with 50 classes, 40 training examples per class, and 1 development example per class.

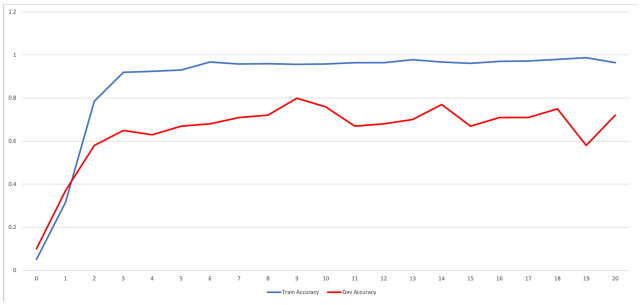

Figure 6: Train loss and development accuracy on TREC Question dataset with 50 classes, 40 training examples per class, and 2 development examples per class.

tokens and maximum best answer length was 252 tokens. We trained our models for 1081 pairs of source document and golden summary. It took an average of 15 hours for training each model for 60 epochs. We understand that this is still not enough for the complete training of the network. Due to resource-constraints, we limited the training to 60 epochs.

5.2.2 Encoder-Decoder Structure

For both the encoder and decoder, we used a single layer unidirectional LSTM with 300 input-embedding dimension and 100 dimensional hidden layer. In the decoder, we extracted attention weights using the encoder hidden states and output hidden state of the LSTM followed by softmax layer for classification over vocabulary. For coverage mechanism and question-centric summarization, we did not change anything in the model structure. For the decoding step, we tried greedy decoder well as beam search decoder with a beam size of 9.

5.2.3 Results

We experimented with three models for summarization: attention-based model, attention-based with coverage mechanism and question-centric attention-based model. Note that the question-centric model includes coverage mechanism also. We randomly chose 10 documents from the dataset for the test purpose. Figure 11 shows one of the examples where we could get a good summarization from our models. It can be observed that the attention based model keeps repeating the same phrases, however, the predicted answer is one of the answers with low rating, not the gold answer. In the second model coverage mechanism helps to remove repetition. The question-centric model performed somewhat better that the coverage-based model. It can be observed that the answers most close to the question were chosen by the model. But they might not be the best rated answer. We observed that the most commonly produced sentence was: "1 It 's not going to get a lot of <UNK>". Models could learn ending of the answers for which we used "**". Models focused on picking phrases and

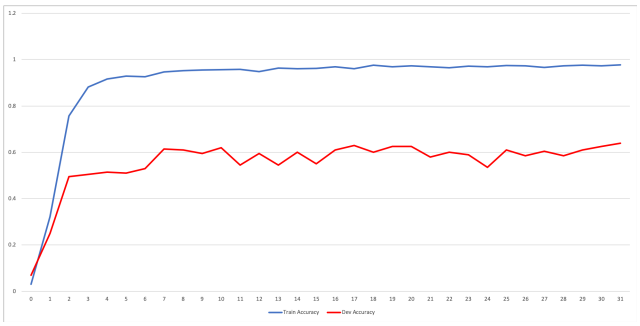

Figure 7: Train loss and development accuracy on TREC Question dataset with 100 classes, 40 training examples per class, and 2 development examples per class .

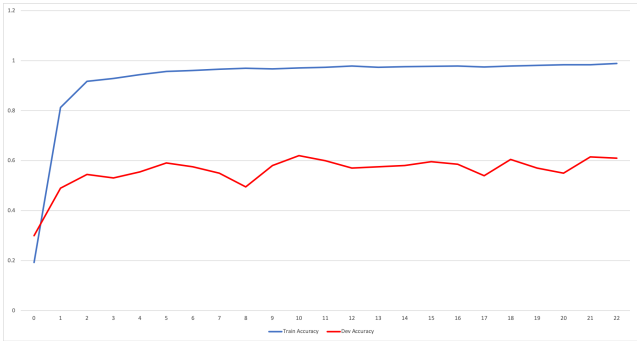

Figure 8: Train loss and development accuracy on TREC Question dataset with 100 classes, 80 training examples per class, and 2 development examples per class .

answers which may not be best rated but are repeated many times in the document. We also tried to get a quantitative analysis, however the performance of the models are not as good compared to the current state-of-the-art [15].

There are two major limitations of our models. First, they exclude rare words present in the dataset. This can be handled with copying mechanism as presented in Pointer-Generator model [15]. Another reason is the nature of dataset. The ideal answer is itself not succinct and contains a lot of information which is not useful, especially for fact-based Q&A. More organized data could have helped us to get better results.

6 Conclusions

In this project, we presented a model for automatic question-answering on websites like Yahoo, Stack Overflow, and Quora. We experimented on two parts: to extract semantically similar questions and to present a summary of previous answers, given those questions. For identifying semantic similarity in questions, we implemented a siamese structure and extended it to a classifier for which we could achieve an accuracy of 80% for question recognition on the development dataset. For answer summarization, we worked on attention-based seq-to-seq summarizer. We also explored a novel idea of including the question's representation in the decoder to give contextual information.

Acknowledgments

We are thankful to Professor Greg Durrett for teaching us this course and giving us hands-on experience in implementing models for Natural Language Processing. We also thank our fellow classmates who motivated us during the course of this project.

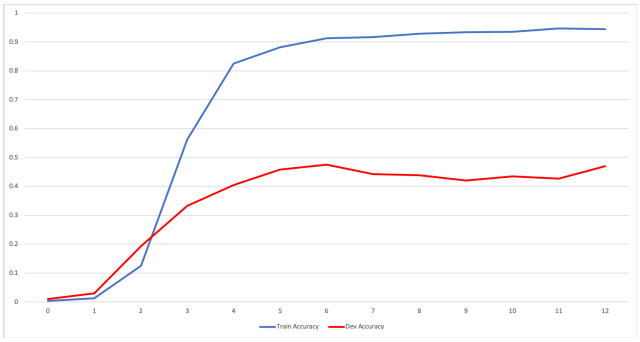

Figure 9: Train loss and development accuracy on TREC Question dataset with 200 classes, 40 training examples per class, and 2 development examples per class .

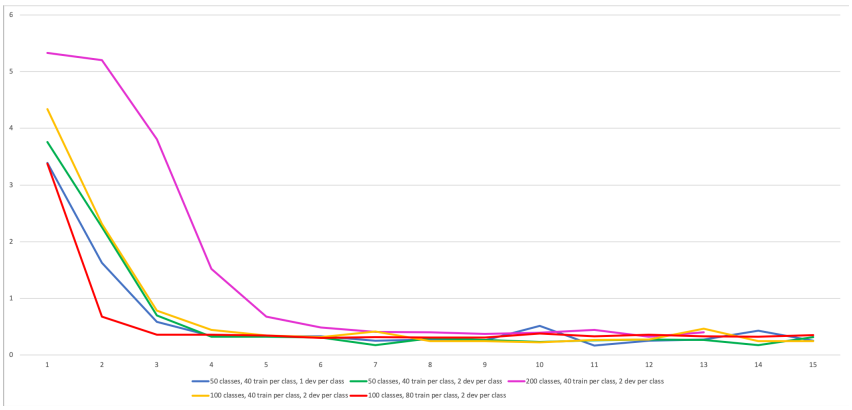

Figure 10: Training Losses for each experiment on TREC Question dataset.

Original Ques:

How can I find the correct adapter for my electric lawnmower battery ?

Reference Answer:

Okay, I'm totally guessing on this, so take this for what its worth: If its a DC current, would reversed polarity matter? I guess it does, since the mower won't kick over. Still shooting in the dark: 1. Jump start it with another battery, see if that does it. Then you'll know the new battery is kaput. 2. Last ditch: if you know the battery is good, swap the cables back to the 'wrong' position and see what happens. You better be quick on the start switch though, and have an extinguisher handy. If those don't work, you might have blown something like the starter solenoid. I've never had a key start mower, so I'm way out of my field here. My Grandpa always said, "If you don't know, go find somebody who does!" On that, I'm out. Good luck.

Attention-based model:

Totally <UNK> on this, so take this for what it's worth so take this so take this so take this so take this so take this <UNK> <UNK> <UNK> <UNK> <UNK> <UNK> If you don't know, go <UNK> <UNK> <UNK> <UNK> If you don't know, go <UNK> <UNK><UNK> <UNK> **

With coverage mechanism:

1. start it with <UNK> <UNK>, see if that does it I've never had a <UNK><UNK> <UNK> , so I'm way out of <UNK><UNK> <UNK> <UNK><UNK> <UNK> . 2. For you don't know, <UNK> <UNK> somebody who does <UNK> <UNK> <UNK> <UNK> <UNK> <UNK>**

Question-centric model:

I've never had a <UNK><UNK> <UNK> so I'm way out of my <UNK><UNK> <UNK>. If you don't know, go find somebody <UNK> <UNK> <UNK> <UNK> **

Figure 11: Example of summarization by the three models.

References

[1] J. Mueller and A. Thyagarajan, "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.

[2] Z. Tu, Z. Lu, Y. Liu, X. Liu, and H. Li, "Modeling coverage for neural machine translation," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 76–85, Association for Computational Linguistics, 2016.

[3] J. M. Zelle and R. J. Mooney, "Learning to parse database queries using inductive logic programming," in *AAAI/IAAI*, (Portland, OR), pp. 1050–1055, AAAI Press/MIT Press, August 1996.

[4] L. S. Zettlemoyer and M. Collins, "Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars," in *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, (Arlington, Virginia, United States), pp. 658–666, AUAI Press, 2005.

[5] G. N. Ramaswamy and J. Kleindienst, "Hierarchical feature-based translation for scalable natural language understanding," in *INTERSPEECH*, 2000.

[6] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *EMNLP*, 2013.

[7] J. D. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, (San Francisco, CA, USA), pp. 282–289, Morgan Kaufmann Publishers Inc., 2001.

[8] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP*, 2014.

[9] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, (New York, NY, USA), pp. 160–167, ACM, 2008.

[10] J. P. C. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *TACL*, vol. 4, pp. 357–370, 2016.

[11] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *HLT-NAACL*, 2016.

[12] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems 27* (Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, eds.), pp. 3104–3112, Curran Associates, Inc., 2014.

[13] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," *CoRR*, vol. abs/1506.03099, 2015.

[14] N. Locascio, K. Narasimhan, E. DeLeon, N. Kushman, and R. Barzilay, "Neural generation of regular expressions from natural language with minimal domain knowledge," *CoRR*, vol. abs/1608.03000, 2016.

[15] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, Association for Computational Linguistics, 2017.

[16] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems 28* (C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, eds.), pp. 2692–2700, Curran Associates, Inc., 2015.

[17] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), pp. 5998–6008, Curran Associates, Inc., 2017.

[19] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, Association for Computational Linguistics, 2015.

[20] S. Iyer, N. Dandekar, and K. Csernai, "First quora dataset release: Question pairs," 2017.

[21] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014.

[22] "Trec 2017 live qa track data." `https://trec.nist.gov/data/qa/2017_LiveQA.html`, 2017.