CS 224N Assignment 4: Reading Comprehension

Chiraag Sumanth
Stanford University

Jayanth Ramesh
Stanford University

Suraj Heereguppe Radhakrishna
Stanford University

{`csumanth, jayanth7, hrsuraj`}@stanford.edu

Abstract

Machine comprehension tests the systems ability to understand a piece of text through a reading comprehension task. We train an end-to-end neural reading comprehension (RC) model that is able to extract and rank a set of answer candidates from a given document to answer questions. Starting off with the baseline suggested in the Assignment handout, we were able to make considerable improvements in performance over the described baseline by exploring and combining several techniques inspired from previous work relevant to Machine Comprehension using the Stanford Question Answering Dataset (SQuAD), in addition to our own ideas and modifications to come up with an effective model.

1 Introduction

Natural Language Understanding (NLU) is a subtopic of natural language processing in artificial intelligence that deals with machine reading comprehension. The process of disassembling and parsing input is more complex than the reverse process of assembling output in natural language generation because of the occurrence of unknown and unexpected features in the input and the need to determine the appropriate syntactic and semantic schemes to apply to it, factors which are predetermined when outputting language. Reading comprehension-based question answering is the task of answering a question with a chunk of text taken from a related document. A variety of neural models have been proposed recently either for extracting a single entity or a single token as an answer from a given text, or for choosing the correct answer by ranking a small set of answer candidates. It is important to note that in both of the above setups, the answer boundary is fixed or can very easily be determined.

1.1 Dataset

SQuAD is comprised of around 100K question-answer pairs, along with a context paragraph. The context paragraphs were extracted from a set of articles from Wikipedia. Humans generated questions using that paragraph as a context, and selected a span from the same paragraph as the target answer.

Context Paragraph: The idea was to create a network of wholly and partially owned channels, and affiliates to rebroadcast the network's programs. In 1959, this rerun activity was completed with program syndication, with ABC Films selling programs to networks not owned by ABC. The arrival of satellite television ended the need for ABC to hold interests in other countries; many governments also wanted to increase their independence and strengthen legislation to limit foreign ownership of broadcasting properties. As a result, ABC was forced to sell all of its interests in international networks, mainly in Japan and Latin America, in the 1970s.

Question: When did ABC Films begin selling programs to other networks?
Answer: 1959

In the SQuAD task, answering a question is defined as predicting an answer span within a given context paragraph. This poses a major challenge to machine comprehension systems as they have to now identify answers in arbitrary positions, and arbitrary lengths, in a context paragraphs, which are also of varying lengths. This significantly increase the search space complexity. For example if n is the number of passage words, the number of possible candidates to consider is in the order of O(n^2).

1.2 Overview of Approach

We first started off implementing the baseline described in our assignment handout. This involved us building a bidirectional LSTM encoder that took in each question and produced it's encoded representation, which was used to then conditionally encode the representation of the context paragraph. We then employed a sequence-to-sequence attention mechanism to produce an attention vector over the context paragraph representation based on the question representation, and this was multiplied with corresponding context paragraph vector encoding. We then had an output layer that was used to determine the start index and the end index of the answer span.

Subsequent to the successful baseline implementation as stated above, we went ahead and explored the various techniques used by other researchers who have worked on this task (see *Related Work*), and decided to build a novel, unique model by combining various ideas which we will describe in the *Model* section of our report. In addition to the baseline features described above, we included a filtering layer in an attempt to give more importance to relevant sections of the paragraph, even before the encoding phase, an additional character-level encoding layer using Convolutional Neural Networks, as well as a paragraph chunk layer in order to improve prediction accuracy. We also implemented a modified version of sequence-to-sequence attention. We saw a considerable improvement in both F-1 and Exact Match (EM) scores over the baseline model.

2 Related Work

The MatchLSTM paper (Wang and Jiang, 2016) [1] bears similarity with the baseline described. The architecture is based on match-LSTM, a model they proposed previously for textual entailment, and Pointer Net, a sequence-to-sequence model proposed by Vinyals et al.(2015) to constrain the output tokens to be from the input sequences. The concepts of LSTM encoding and sequence-to-sequence attention are validated through their work as effective techniques for this task.

In the paper, Multi-Perspective Context Matching for Machine Comprehension (Wang et al., 2016) [2], they describe a Filtering layer, to filter out redundant or irrelevant information from the passage using the concept of a relevancy score for each word in the context paragraph. This has been incorporated into our final model.

Bi-Directional Attention Flow (Seo, Kembhavi et al., 2017) [3], introduces a novel concept of combining both character and word level representations using a highway network (Srivastava et. al, 2015) [7] to improve performance of Reading Comprehension Systems. We also take inspiration from their work in our model.

We also adopted techniques from the work of Yu et al., 2016 [4] in End-to-End Answer Chunk Extraction and Ranking for Reading Comprehension, where they applied a novel word-by-word attention mechanism to acquire question-aware representations for the paragraph, followed by the generation of chunk representations and a ranking module to propose the top-ranked chunk as the answer.

All these models clearly outperformed the Logistic Regression model proposed by Rajpurkar et al., 2016 [5] and have obtained close to state-of-the-art performance on the SQuAD dataset, and we imagined using a mix of techniques as described above in our model would serve us well in improving over the baseline model we first implemented.

3 Methods and Models

In this section we will describe the various models we implemented in this assignment, finally culminating in our model as described in section 3.2.

3.1 Baselines

Three different models were explored as part of training the baseline model, each of which have been described briefly.

3.1.1 Baseline 1: Feed-Forward Neural Network

All possible windows with a fixed possible maximum length were generated from a given context paragraph in the training data. The sum of the word vectors corresponding to the words in the generated windows were used as input to the neural network. A similar procedure was employed to obtain the question representation. The input to the first layer of the network were the question representation and the vector representation of a particular window of the context paragraph. The second layer was fed the same question representation and the hidden layer output of the first layer. The final layer of the network output the softmax probability that the given window is the answer to the question. After getting the probabilities for each such window, the window with the highest probability was selected to be the final prediction for the training example. Cross-entropy loss was used to train the model, with one-hot vector representation of the candidate windows as the ground truth. The index corresponding to the correct window was marked with **one** while all other indices were **zero**. This model reported F1 scores of 4.61% with corresponding EM scores of 3.54%.

3.1.2 Baseline 2: Recurrent Neural Network

Next, a simple RNN was implemented with a naive form of attention. The inputs to the RNN at every time step were: 1) Vector representation of the question, 2) Vector representation of the current context word, and 3) The hidden layer vector output from the previous time step. The question vector was fed at every time step in order to simulate a person looking for the answer in a paragraph while keeping the question in mind constantly. The model was made to output two probabilities for every context word: 1) The probability that the current word is the starting word of the answer window, and 2) The probability that the current word is the ending word of the answer window. The loss was decomposed as the sum of two cross-entropy losses, one for predicting the starting word and the other for predicting the ending word. This model reported F1 scores of 7.95% and EM scores of 5.38%, a small improvement over the simple feed-forward neural network.

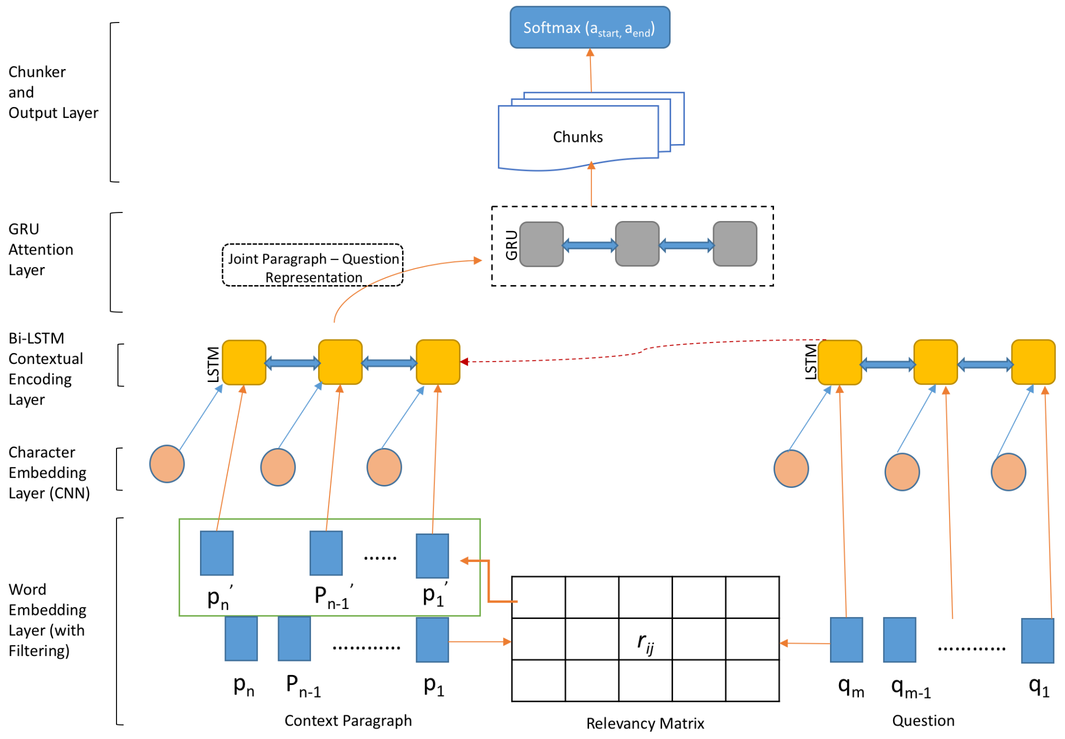
3.1.3 Baseline 3: Bidirectional Long-Short-Term-Memory

The third and final model we explored for the baseline was a BiLSTM. The question representation was arrived at by running a BiLSTM over the question and concatenating the two hidden vectors. A hidden state representation of the context paragraph was also evaluated by running a BiLSTM over it and concatenating the two resulting hidden state vectors. The attention vector was calculated as described in the work of Wang and Jiang, 2016 [1]. A final LSTM was run over the fused representation of the question and context paragraph to predict the probabilities that each word in the context paragraph is the starting and ending word in the answer. This model reported scores of 35.87% and EM scores of 26.46%, a leap in results over the two previously implemented baseline models.

3.2 Final Submission Model

Figure 1 depicts the architecture of our final submission model. It is comprised of the following layers.

Word Embedding Layer (with Filtering): The Word embedding layer to map each word to a high-dimensional vector space. We use pre-trained word vectors, GloVe (Pennington et al., 2014) [8], with each word vector having a dimension of 100. Additionally, upon studying the dataset, it is clear that there are a large number of examples, the relevant parts to answer the given question is contained in a relatively small part of the context paragraph. Therefore this layer helps filter out redundant or irrelevant information from the context paragraph, and allows more importance to be given to the more relevant words in the paragraph, with respect to the given question. We implement it in a way as described by Wang et al., 2016 [1]. We compute a relevancy score $\boldsymbol{r_{ij}}$ between every word $\boldsymbol{p_j}$ in the context paragraph and every word $\boldsymbol{q_i}$ of the given question using the cosine similarity



Chunker and Output Layer
GRU Attention Layer
Bi-LSTM Contextual Encoding Layer
Character Embedding Layer (CNN)
Word Embedding Layer (with Filtering)
Softmax (a_{start}, a_{end})
Chunks
GRU
Joint Paragraph – Question Representation
LSTM
LSTM
p_n'
P_{n-1}'
p_1'
p_n
P_{n-1}
p_1
r_{ij}
q_m
q_{m-1}
q_1
Context Paragraph
Relevancy Matrix
Question

Figure 1: Overview of the architecture of the final model

measure: $\boldsymbol{r_{ij}} = \frac{q_i \cdot p_j}{||q_i|| * ||p_j||}$. The relevancy degree $\boldsymbol{r_j} = \mathbf{max_i} \boldsymbol{r_{ij}}$. Finally we perform the transformation $\boldsymbol{p'_j} = \boldsymbol{r_j} \cdot \boldsymbol{p_j}$, and pass $\boldsymbol{p'_j}$ to the next layer.

Character Embedding Layer: We adopt the technique described in BiDAF (Seo, Kembhavi et al., 2017) [3] to map each word to a vector space using character-level CNNs. In this approach, we train a simple CNN with one layer of convolution on top of the GloVe word vectors, backpropogating through the vectors, and then use max-pooling at the output layer of the CNN to get the character level embedding vector for that word.

Highway Network: Subsequently, we concatenate the character and word embedding vectors and pass it to a two-layered Highway Network (Srivastava et al., 2015) [7]. The outputs of the Highway Network are two matrices, $\mathbf{Q}^{\mathbf{d} \, x \, \mathbf{m}}$, representing the every token of the question as a d-dimensional vector, and $\mathbf{P}^{\mathbf{d} \, x \, \mathbf{n}}$, representing every token the context paragraph as a d-dimensional vector. Adopting Highway Networks allows us to train increasingly layered neural networks optimized directly with Stochastic Gradient Descent by smmothening gradient flow across the various layers, duly noted by the authors of the BiDAF paper.

Contextual Embedding Layer: This layer is seen in almost every paper relevant to Machine Comprehension. It serves the purpose of encoding contextual information into the hidden representation of every token in the question, as well as paragraph. The paragraph is in fact, encoded conditionally with respect to the question representation. As standard we used a Bi-Directional LSTM for this purpose and the final representation of the question and context paragraph is the concatenation of the LSTM outputs in the two directions. Therefore we get two matrices, $\mathbf{H_q}^{\mathbf{2d} \, x \, \mathbf{m}}$ from matrix $\mathbf{Q}$, and $\mathbf{H_p}^{\mathbf{2d} \, x \, \mathbf{n}}$, from matrix $\mathbf{P}$.

Attention Layer: This layer is inspired by the work of Yu et al., 2016, that proposes a novel, simplified word-to-word attention mechanism leading to a joint Paragraph-Question representation. For every vector $\mathbf{h_k}^{\mathbf{q}}$ in $\mathbf{H_q}$, and $\mathbf{h_j}^{\mathbf{p}}$ in in $\mathbf{H_p}$, we compute the dot product $\alpha_{jk} = h_j^p \cdot h_k^q$. Then we compute a weighted pooling vector across all the tokens in the question, as $\beta_j = \Sigma_k^m \alpha_{jk} h_j^p$. Thus we now have a 2d-dimensional vector β_j and concatenate it with h_j^p to get a 4d-dimensional vector c_j which is the joint paragraph-question representation vector for every token j in the context paragraph. We then pass c_j to a Bi-Directional GRU to get a d-dimensional vector in each direction, and finally concatenate the two into a 2d-dimensional vector γ_j = [$\vec{\gamma_j}$,$\overleftarrow{\gamma_j}$]

Chunker and Output Layer: In this layer we first construct candidate answer chunks, i.e., starting from a given p_j in the paragraph, we construct all possible answer chunks up-to a maximum chunk length **U** (which we determined as 1.2 times the longest answer span in the training data times, in lieu of slightly longer possible answer spans in unseen data). As noted in Yu et al., 2016 [4], the chunk representation s_{mn} [spanning from tokens m through n] is best represented as a concatenation of the hidden state of the first word in a chunk in the forward GRU and that of the last word in the backward GRU, i.e s_{mn} = [$\vec{\gamma_m}$,$\overleftarrow{\gamma_n}$]. We then choose the chunk s_{mn} that maximizes the probability Pr(A_{mn} | P,Q), given a question Q, and context paragraph P, where A_{mn} is the answer constructed from the corresponding tokens in the chunk s_{mn}. The start and end indices a_{start} and a_{end} are thus determined, based on the start and end indices of the most optimal chunk.

Comment about indices: It is interesting to note here that this chunking technique to determine the best answer, and subsequently determining the start and end indices of that predicted answer ensures that the start index is always lesser than or equal to that of the end index, avoiding a common problem seen in cases where two separate predictions are made, one each for the start index and end index, that may lead to the end index being predicted as something that is lesser than the start index.

4 Results and Discussions

4.1 Results:

Table 1 summarizes our results for the various models described above, and also includes layer ablation effects.

Model	Train F1	Test F1	Train EM	Test EM
Feed-Forward Neural Network	5.24%	4.61%	4.81%	3.54%
Without masking	3.47%	2.86%	2.26%	1.47%
Without dropout	5.33%	4.47%	4.98%	3.35%
Without regularization	5.27%	4.56%	4.85%	3.48%
Recurrent Neural Network	8.13%	7.95%	6.02%	5.38%
Without masking	3.94%	3.46%	4.35%	3.51%
Without dropout	8.22%	7.76%	6.08%	5.19%
Without regularization	8.19%	7.91%	6.10%	5.31%
Bidirectional Long-Short-Term-Memory	38.76%	35.87%	29.15%	26.46%
Without masking	28.64%	27.36%	19.86%	17.17%
Without dropout	36.12%	34.52%	28.17%	24.73%
Without regularization	36.91%	34.85%	28.43%	24.89%
Final Submission Model	**61.15**%	**55.95**%	**45.84**%	**43.60**%
Without character embedding	60.86%	55.31%	44.27%	43.53%
Without filtering	57.74%	53.08%	42.46%	41.83%
Without chunking	54.57%	50.42%	40.15%	39.87%

Table 1: Overview of the architecture of the final model

During **layer ablation**, it can be seen that character embeddings indeed help towards better model performance on the test set. This maybe attributed to the fact help better encode out-of-vocabulary words unseen during training, as opposed to just using word embeddings. *Filtering* is also of considerable value addition in helping the downstream attention layer by initially weighing down less relevant parts of the paragraph tokens. A major contributor to our model architecture is the *chunking layer*. We feel this immensely helps the model make better predictions as it sees smaller localized chunks of the paragraph as opposed to making direct start/end index predictions over the entire paragraph length. Additionally, as noted in the comment about indices in section 3.2, this layer helps alleviate the problem of index ordering.

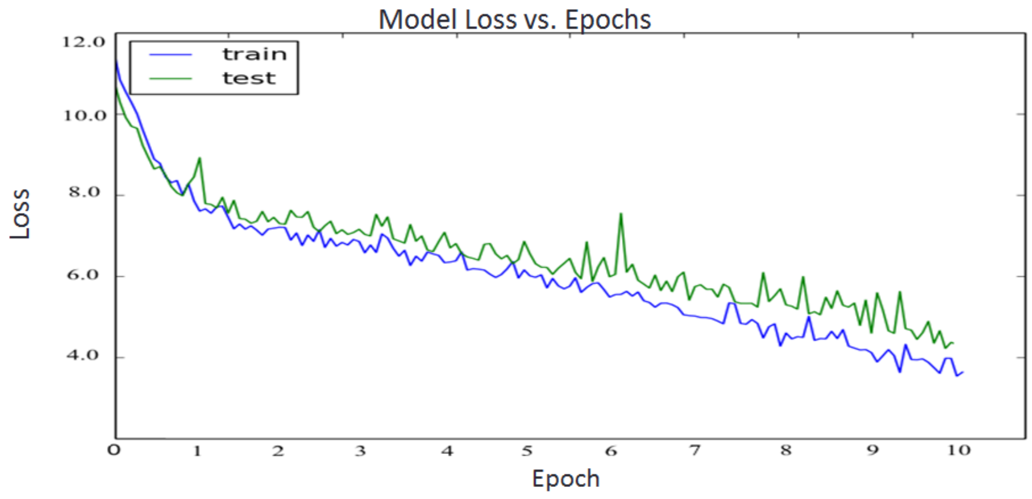
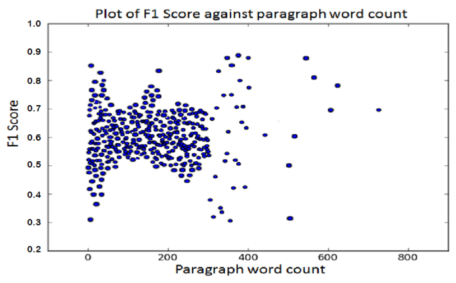
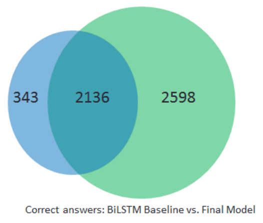


Figure 2: Train and Test loss vs Number of epochs

Model Hyper-parameters: Hyper parameter search via evaluation on the validation set led us to the following configuration. For the character level CNN, we used 100 1D filters each of width = 3, coupled with max pooling. Both GRU ans LSTM had a hidden state dimension d = 200, and we used dropout with keep probability as 0.8 in all our neural network layers. We used the Adam Optimizer with learning rate of 0.01, batch-size of 32, and trained for 10 epochs where we observed convergence, as observed in Figure 2. We also used d = 100 for GloVE vectors trained on the 6B Wiki dataset. In future, we hope to explore with word embeddings of a larger dimension, trained on a bigger corpus to improve model performance.



(a) Figure 3a: F1 scores v word count of paragraphs



(b) Figure 3b: Final Model (green) vs Baseline (blue)

The final model was evaluated on its F1 scores as a function of the number of words in a context paragraph. The results are presented in Figure 3a. Two hundred random training examples were chosen for this analysis. As seen, there is no significant performance degradation as the length of a paragraph increases. This proves that the model is able to focus on short sections of relevant text even in a long document.

Figure 3b visually depicts how our final model performs when compared to the Bi-LSTM baseline in terms of the number of questions in the dev set correctly answered by our model (shown in blue), when compared to the baseline (shown in green).

4.2 Discussion and Error Analysis

Observation of the performance of the models shows that masking is very important in achieving good results. As we see from Figure 4, most paragraphs have word counts within 300. However, while training all our models, we have padded our training examples to the maximum length of a paragraph in the entire training set, which is 766. Hence, while predicting without masking, the model will tend to predict indices more than 300 with a higher probability. For all training examples whose length of the context paragraph is less than 300, these are invalid predictions. However, with masking the model is forced to predict an index that falls in the range of the length of a given context paragraph, i.e. a training example. Hence, masking is critical to the performance of a model.

None of the models showed high tendency to overfit the training data. Dropout and regularization only help in cases where a model overfits the training data and performs poorly on the test set. Since the model generalizes fairly well over the test set, dropout and regularization contribute little in helping increase the performance of the model over the test set. This is reflected well in the performance of the models, which is documented in the Table 1.

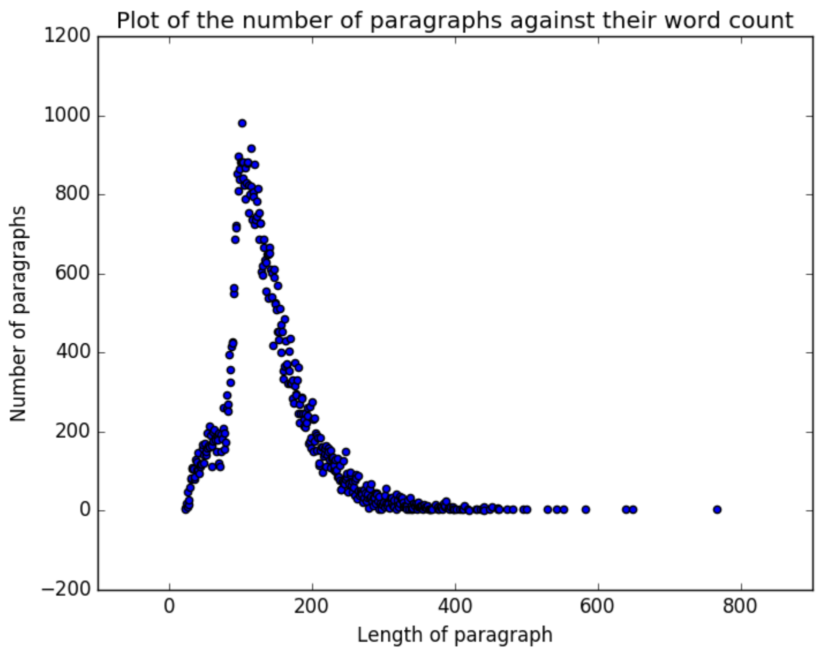


Figure 4: Plot of the number of paragraphs vs. Respective word counts

Figure 5 represents the percentage of questions answered correctly by our model versus the Bi-LSTM baseline, across different classes of questions such as *Why, What*, etc. We can see that models seem to consistently perform relatively poorly on the *Why* class of questions, and better on the *When* class of questions, which are perhaps typically just time or date-like facts, as opposed to more complex answers for the *Why* class.

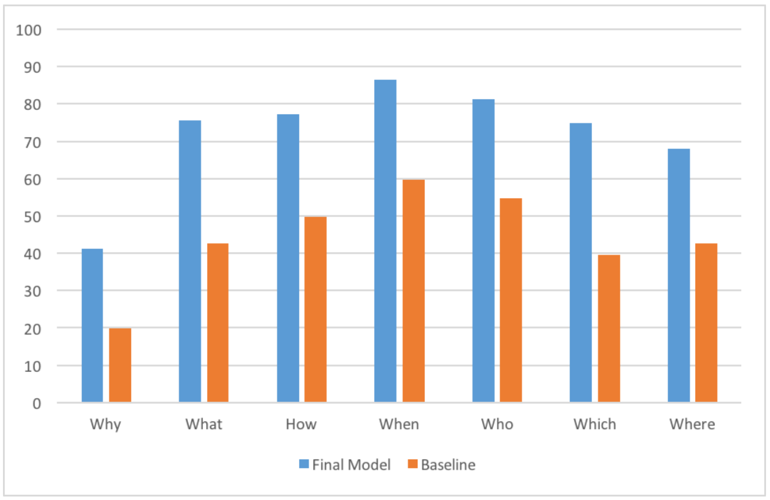


Figure 5: Percentage of questions answered correctly across different question types

Table 2 below lists samples we chose to capture the different types of errors our model makes.

Error Category	Example
Non-exact answer boundary	***Question***: The Amazon rainforest makes up what amount of Earth's rainforests? ***Context***: The Amazon represents over half of the planet's remaining rainforests, and comprises the largest and most biodiverse tract of tropical rainforest in the world, with an estimated 390 billion individual trees divided into 16,000 species. ***Prediction***: half ***Ground truth***: over half
Ambiguities and subtle semantic differences	***Question***: What color was used to emphasize the 50th anniversary of the Super Bowl? ***Context***: As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives ***Prediction***: golden ***Ground truth***: gold
Answer spanning multiple phrases	***Question***: What three things did the Windows Everywhere campaign emphasize on ? ***Context***: Microsoft premiered the first ad in its "Windows Everywhere" campaign , which promoted Windows 8 , Windows Phone 8 , and the company 's suite of online services as an interconnected platform . ***Prediction***: promoted Windows 8 ***Ground truth***: Windows 8 , Windows Phone 8 , and the company 's suite of online services
Incorrect tokenization	***Question***: How many jobs in 2008 in Greece were somehow related to the tourism industry ? ***Context***: The number of jobs directly or indirectly related to the tourism sector were 840,000 in 2008 and represented 19% of the country's total labor force . ***Prediction***: 840, ***Ground truth***: 840,000
External knowledge [*In this specific example, unable to differentiate between major and non-major cities*.]	***Question***: What is the major US city that the is the university located? ***Context***: with campuses throughout the Boston metropolitan area: its 209-acre (85 ha) main campus is centered on Harvard Yard in Cambridge, approximately 3 miles (5 km) northwest of Boston. ***Prediction***: Cambridge ***Ground truth***: Boston

Table 2: Error Analysis Table

5 Conclusion and Future Work

In this project, we tackled a fairly difficult problem of machine comprehension. The problem is difficult because the answer to the question can be any contiguous chunk of the paragraph, which means that the number of possible answers and the number of words in an answer are variable. We presented various techniques, and our final model is a unique combination of our own ideas, in addition to exploring ideas inspired from previous literature. We were able to obtain significant improvement over the baseline model, and believe that given more time, we can iron out inefficiencies and undiscovered nuances in our implementation and considerably improve on our current performance.

In future we would be exploring different attention mechanisms above and beyond traditional sequence-to-sequence ones. We would also like to research more on alternative approaches to data-flow and representation fusion like fine-grained gating [6].

References

[1] Wang, Shuohang, and Jing Jiang. ”Machine comprehension using match-lstm and answer pointer.” arXiv preprint arXiv:1608.07905 (2016).

[2] Wang, Zhiguo, et al. ”Multi-Perspective Context Matching for Machine Comprehension.” arXiv preprint arXiv:1612.04211 (2016).

[3] Seo, Minjoon, et al. ”Bidirectional Attention Flow for Machine Comprehension.” arXiv preprint arXiv:1611.01603 (2016).

[4] Yu, Yang, et al. ”End-to-End Answer Chunk Extraction and Ranking for Reading Comprehension.” arXiv preprint arXiv:1610.09996 (2016).

[5] Rajpurkar, Pranav, et al. ”Squad: 100,000+ questions for machine comprehension of text.” arXiv preprint arXiv:1606.05250 (2016).

[6] Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, Ruslan Salakhutdinov. ”Words or Characters? Fine-Grained Gating for Reading Comprehension.”

[7] Srivastava, Rupesh Kumar, Klaus Greff, and Jrgen Schmidhuber. ”Highway networks.” arXiv preprint arXiv:1505.00387 (2015).

[8] Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. ”Glove: Global Vectors for Word Representation.” EMNLP. Vol. 14. 2014.