General Question Answering for the MS MARCO Dataset

TJ Gilbrough and **Lane Aasen** and **Omar Alhadlaq**
{tgilbrou, aaasen, hadlaq}@cs.washington.edu

Abstract

Machine comprehension is an integral piece in the quest for general artificial intelligence. Recently published models have utilized attention mechanisms to achieve promising results on question answering datasets such as SQuAD. In this paper, we attempt to extend these models to the MS MARCO dataset in order to build a general question answering system, capable of tapping into the knowledge held within a wide array of internet documents.

1 Introduction

Traditionally, search engines have focused on referring people to web pages with answers to their questions. More recently, some search engines have begun to directly answer simple, factual questions using curated databases of basic knowledge.

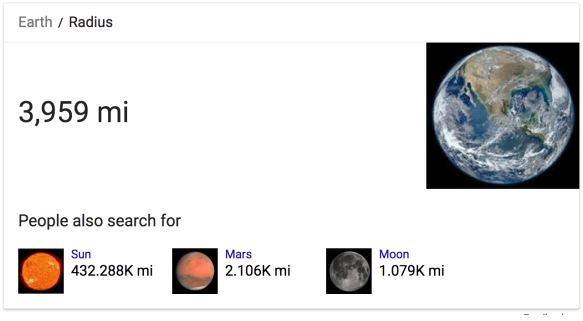

Figure 1: Google directly answers the question 'what is the radius of the earth?'

However, current approaches to question answering are rigid and limited to encyclopedic questions. What if we could create a system that could utilize the entirety of the public internet to answer complex questions spanning the sum of human knowledge? This would be the holy grail of question answering. It would revolutionize search engines, personal assistants, and other artificial intelligence products.

Recent advancements in machine comprehension have made it possible to create a basic version of such a system. In October of 2016, the Stanford Question Answering Dataset (SQuAD) was published, which contains over 100,000 questions based on over 500 Wikipedia articles. When SQuAD was released, it was significantly larger than previous reading comprehension datasets. However, since SQuAD is based on a relatively small and well curated set of Wikipedia articles, it is a poor choice for training reading comprehension models for the entire public internet.

At NIPS 2016, Microsoft released the MS MARCO (Microsoft MAchine Reading COmprehension) dataset. Questions in the MS MARCO dataset are sampled from real-world Bing queries, and answers are human-generated based on excerpts from Bing search results. This makes MS MARCO the most promising dataset so far for training general purpose question answering models.

Recent research into recurrent neural networks and attention mechanisms has resulted in considerable improvements to machine comprehension models. Models such as BiDAF (Seo et al., 2016) and Dynamic Coattention Networks (Xiong et al., 2016) have achieved impressive results on the SQuAD dataset and pushed the state of the art forward. In this paper, we hope to adapt these and other models to the more complex MS MARCO dataset, with the goal of creating a truly general purpose question answering system with access to the entirety of human knowledge.

2 MS MARCO Dataset

To create the MS MARCO dataset, people were shown real Bing search queries along with passages extracted from the top Bing search results. They were then asked to select the relevant passages and use them to answer the question.

The MS MARCO dataset contains 102,023 question-answer pairs with a train-validation-test split of approximately 80-10-10. There are between 1 and 12 passages for each question with an average of 8.2. Passages are between 19 and 1,167 characters in length, with an average of 422. There are an average of 1.07 'selected' passages, passages marked as the source of the answer, for each question. Virtually all of the dataset is in English.

Unlike previous reading comprehension datasets like SQuAD, the answers in MS MARCO may not come directly from the context passages. Only about 60% of answers are contained in the context passages verbatim. This presents a unique and novel challenge, since models must have some generative capabilities in order to achieve high performance on the MS MARCO dataset.

Questions are separated into five types based on the form of their answer. These labels are provided for all questions, including those in the test dataset.

- Description (54.6%)
  - e.g. What is vitamin A used for?
- Numeric (27.6%)
  - e.g. How long does it take for a pulse of laser light to reach the moon?
- Entity (10.4%)
  - e.g. What is a hummingbird moth?
- Location (4.9%)
  - e.g. Where do Roseate Spoonbills live?
- Person (2.5%)
  - e.g. Who first invented the anemometer?

Of these question types, description questions are both the most common and complex. Description answers commonly synthesize information from multiple passages and are only taken verbatim from the context passages 54% of the time, versus 68% for non-description answers. Answers to description questions can also have much more variability than those to other types of questions. For example, there are thousands of correct answers to the question 'What is vitamin A used for?' which makes training a model for MS MARCO even more complex. This variability in humans answers explains why humans can only achieve Bleu-1 scores of 46/100 and Rouge-L scores of 47/100 on the validation set, which is quite low.

3 Methods

We viewed this project as having two main modules, the answer extractor and the passage relevance model. The answer extractor is given a passage and a query, and is expected to find the answer to that query within the given passage. The passage relevance model ranks passages according to how relevant they are to the given query.

The output of the answer extractor is simply the predicted start and end index of the answer in the text. Along with these indexes, we can also extract the probabilities of those indexes by taking a softmax of the logits produced by the model.

To pick the final answer for the query, we pass each of the passages through the answer extractor. From this we get the probability of the start and end indexes of the answer within each individual passage. Define $P(s_p = i)$ and $P(e_p = j)$ as the probability of the start and end indexes of the answer at i and j respectively, within passage p. We also pass through each passage to the passage relevance model to get its relevance weight, w_p. To pick the best answer, pick the passage and indexes with the following formula:

$$\arg\max_{p,i,j}(w_p \cdot P(s_p = i) \cdot P(e_p = j))$$

3.1 Training

The only training in the model takes place within the answer extractor. Therefore, we trained the answer extractor separately on queries with single passages before using it in our multi-passage model. Since many of the answers to the queries cannot be found verbatim within the passages, we had to filter the training data. To do this, we selected only samples that had its answer contained verbatim within a 'selected' passage. A selected passage is one which contained a 1 within its `is_selected` field within the dataset. The

reason the passage must have been selected is because we needed to ensure that the answer is used in the correct context. For example, if the query is 'where is the Space Needle?' and the true answer is 'Seattle', we would not want to train the model on a passage that happened to mention Seattle but had no reference of the Space Needle. It would only confuse the model.

In addition to filtering the training data, we decided to train five separate models for the five different query types. Each of the different query types have distinctive patterns of where the answer falls in the passage, and by training five separate answer extractors, the model can exploit these patterns. One disadvantage to this approach is that we are further reducing the training data that each answer extractor uses. Since we were limited with our resources and time, this was a trade-off we were willing to take.

3.2 Word Embeddings and Unknown Tokens

The first step in each of the answer extractor models is to convert the tokenized text into a series of word-level embeddings. To reduce the complexity of the model, we utilized pretrained GloVe embeddings (Pennington et al., 2014). In order to experiment with different word embedding sizes, we primarily stuck with the Wikipedia + Gigaword word embeddings which had sizes 50, 100, 200, and 300. The vocabulary size for each of these sets of embeddings is 400,000.

Query Type	Val Unk %	Test Unk %
Person	1.969	1.926
Location	2.134	1.990
Numeric	2.110	2.099
Entity	1.802	1.803
Description	1.917	1.915

Table 1: Percentage of unknown word in each query type

Even with a large vocabulary, there is still a good portion of the words that do not have corresponding pretrained embeddings. Displayed in table 1 is the percentage of unknown tokens in the multi-passage datasets. Instead of embedding these words as the zero vector, we implemented special, trainable, unknown vectors that represent particular unknown classes. The four that we settled on were:

- numeric: unknown token that contains a number in it
- punctuation: unknown token entirely consisting of punctuation
- alphabetic: unknown token made up of letters a-z (often rare nouns or misspellings)
- wildcard: unknown token that does not match the above classes

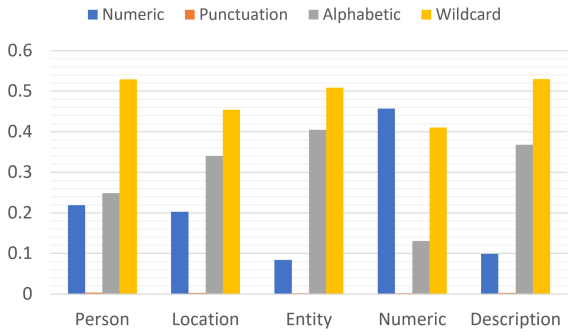

Figure 2: Percentage of unknown classes for each query type

In figure 2 we see the distribution of unknown tokens of the different query types across our defined unknown classes. In all but the numeric dataset, the majority of the unknown tokens fell into the wildcard class. This means that most of the unknown tokens are some combination of alpha-numeric characters with some sort of punctuation, or even some sort of foreign language or unicode characters. If given more time, other classes we would have liked to experiment with include capitialized words, foreign language words, numerical tokens with units, years, zip codes etc.

3.3 Answer-Extractor Models

We found that the answer extractor model is somewhat well defined with many papers written on the subject. In fact, The Stanford Question Answering Dataset (Rajpurkar et al., 2016) leaderboard provided many of these papers. The difference between the SQuAD and MS MARCO datasets is that for SQuAD, each sample contains multiple queries given a single passage, while with MS MARCO, each sample contains multiple passages given a single query. Therefore we based our advanced answer extractor models on the previous work for the SQuAD dataset.

3.3.1 Baseline

For our baseline model, we started with a very basic design. The model has two bi-directional GRU RNNs. The first RNN takes a matrix of the word embeddings of the question: $\mathbf{Q} \in \mathbb{R}^{d \times J}$, where d is the size of the word embeddings and J is the maximum question length. The second RNN takes the word embeddings of the context: $\mathbf{X} \in \mathbb{R}^{d \times T}$, where T is the maximum context length. The output of each RNN is the concatenation of the backward and forward hidden states of that RNN, hence we obtain two matrices: $\mathbf{U} \in \mathbb{R}^{2d \times J}$ from the question RNN, and $\mathbf{H} \in \mathbb{R}^{2d \times T}$ from the context RNN. $\mathbf{U}$ is a matrix representing a vector for each word in the question, such that $\mathbf{U} = [\mathbf{q}_1 ... \mathbf{q}_J]$. The question representation we later consider is an average of the different word vectors outputted from the question RNN:

$$\bar{\mathbf{q}} = \frac{1}{J} \sum_{i=1}^{J} \mathbf{q}_i$$

The question representation $\bar{\mathbf{q}}$ and the output of the context RNN $\mathbf{H}$ are concatenated and inserted in a third bi-directional dynamic GRU RNN. Then, we have a dense output layer that takes the output of the third RNN. From the output layer we get the logits for the starting and ending indices of the answer, from which we take the arg max. All three RNNs have a dropout as a regularization technique to reduce overfitting during training.

3.3.2 Attention

The attention model is very similar to the baseline model. The only difference is in defining the question representation $\bar{\mathbf{q}}$ that we use in the third RNN. Before, we used to average the different word vectors we're getting from the question RNN. In the attention model, for each context word $\mathbf{x}_k$ we get a question representation $\bar{\mathbf{q}_k}$ which is a weighted average with trained weights:

$$\bar{\mathbf{q}_k} = \frac{1}{J} \sum_{i=1}^{J} \mathbf{p}_{k,i} \mathbf{q}_i$$

such that $\sum_i \mathbf{p}_{k,i} = 1$. The attention weights are learned with a softmax and a dense layer:

$$\mathbf{p}_{k,i} = \text{softmax}(\mathbf{W}[\mathbf{x}_k; \mathbf{q}_i; \mathbf{x}_k \circ \mathbf{q}_i] + b)$$

where $\circ$ is the element-wise product and ; is row-wise concatenation.

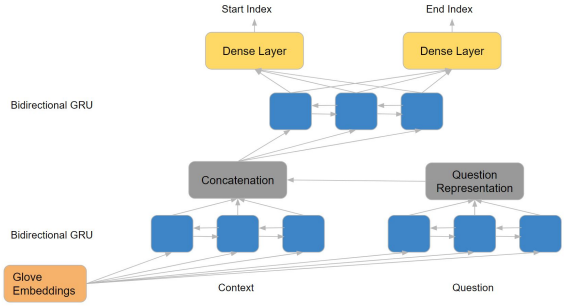

Figure 3: The Baseline and the Attention model

3.3.3 Coattention

The main component of the coattention model is the coattention encoder, an attention mechanism that accounts for the important words in the passage in light of the question, and the important words in the question in light of the passage. Intuitively, this is like reading the question first and then searching the passage for the answer by looking for words that seem particularly relevant. Instead of the dynamic decoder proposed in the Dynamic Coattention Networks (Xiong et al., 2016), we used a simple dense layer for each the start and end outputs. We found that the iterative approach would be ideal in theory but we could not get it to converge nicely. Further details on the implementation can be found here.

3.3.4 BiDAF

BiDAF is a model designed by the University of Washing and Allen Institute of Artificial Intelligence (Seo et al., 2016) which performed very well on the SQuAD dataset. The BiDAF model consists of six layers, of which we implemented five: word embedding, phrase embedding, attention, modeling, and output. Given more time, we could also implement the character embedding layer. Further details on the implementation can be found here.

3.4 Passage Relevance

With the passage relevance model, there were multiple different directions to take it. Since the passages for each sample all contained very similar tokens, we struggled to find training data and train a model that could pick out just the single passage that is relevant to the queries. In addition, since the answer is commonly contained in multiple passages, there are opportunities to exploit this when considering all the passages. Therefore,

the best option was to have a model assign a relevance weights to each passage and then rank the passages based on their weights.

3.4.1 TF-IDF

Term frequency-inverse document frequency, or TF-IDF for short, is a numerical statistic that represents how important a word is to a document in a corpus. The statistic counts the number of occurrences of a particular word in the document and then offsets it by how often the word appears in the corpus. We relied on this statistic to determine how relevant a passage is to the given query, compared to the other passages provided.

Define $\mathrm{tf}(t, d)$ to be the term frequency of term t in document d. Most of the time this is simply the number of times t appears in d, but we tweaked it to be a binary variable indicating if the term appears or not in the document. Since every passage is relevant to the query in some way, we just want the ones that contain the key words the query is looking for. Next, define $\mathrm{idf}(t, D)$ to be the inverse document frequency of term t in corpus D. $\mathrm{idf}(t, D)$ is calculated by:

$$\mathrm{idf}(t, D) = \log\left(\frac{N}{df_t}\right)$$

where N is the number of documents in D, and df_t is the number of documents in D that contain t. Finally, the TF-IDF value can be calculated for a term t in document d within corpus D:

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

For the model, we compute the TF-IDF matrix, where the first row is the query, and each subsequent row is a passage, and each column is a word in the vocabulary. Once the matrix is computed, rows can be compared with any similarity metric. We decided to use a linear kernel, which through experimentation, gave us the best results.

To compute the passage relevance weights, compute the similarity of the first row (the query) with each of the other rows (the passages) in the matrix. Finally, normalize the weights so that they sum to one.

3.5 Evaluation

To evaluate the performance of the model, Microsoft has released evaluation scripts for the MS MARCO dataset [1]. Given both the true and predicted answers, the scripts output the Bleu-1 and Rouge-L scores. The Bleu-1 score measures the precision of unigrams, while the Rouge-L score measures the recall of the longest co-occurring in sequence n-grams. In other words, the Bleu score is how much of the words in the predicted answers appeared in the reference answers, while the Rouge is how much of the words in the reference answers appeared in the predicted answers. Captured in table 2, are benchmark performances at the time of writing this paper.

[1]The evaluation scripts can be found on MS MARCO's website: http://www.msmarco.org/submission.aspx

Model	**Bleu-1**	**Rouge-L**
ReasoNet Baseline	14.83	19.20
R-Net (Leader)	42.22	42.89
Human Performance	46.00	47.00

Table 2: MS MARCO benchmark performances

4 Results

4.1 Answer-Extractor Results

Model	**Bleu-1**	**Rouge-L**
Baseline	46.6	53.8
Attention	52.6	51.8
Coattention	50.6	57.0
BiDAF	**53.9**	**57.6**

Table 3: Answer extractor results on selected passages

Feeding the answer extractor just the selected passages for each sample, we used the evaluation scripts to score how well the answer extractors did. These results are display in table 3. Clearly the more complex the model became, the better the result became as well. Consequently, what it came down to was how much would the multiple passages affect these results? With the multiple passages, there is much more noise surrounding the answer, as well as variation on how the answer expressed.

4.2 Passage Relevance Results

Our passage relevance model selects the 'selected' passage as the most relevant just 20.2% of the time. Although this is quite low, often times the answer is contained in more than just the selected passage, so this is alright. Analyzing the cumulative accuracy of the model in figure 4, it can be noted that the passage relevance model gives

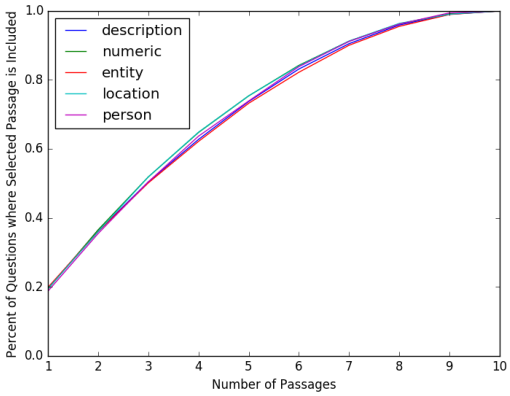

Figure 4: Passage relevance ranking accuracy

at least somewhat of an advantage over random ranking. The model does a nice job in particular burying irrelevant passages at the bottom of the ranking. Since our TF-IDF calculations are using binary variables for the term frequencies, multiple passages score similarly compared to each other. This is by design in our model. We want the answer extractor confidence to play a large role in selecting the answer, not being overshadowed just because a particular passage uses key terms more than others.

4.3 Multi-Passage Results

Model	Bleu-1	Rouge-L
Baseline (Microsoft's)	14.83	19.20
Baseline (Ours)	28.92	**26.88**
Attention	25.88	25.01
Coattention	26.56	25.48
BiDAF	**29.22**	26.30

Table 4: Multi-passage results

Combining the answer extractor and passage relevance models together, we now evaluated our multi-passage model. The resulting scores of the four models are displayed in table 4. Quite surprisingly, we did not see the same trend as we did in the answer extractor models. We believe this is due to the different probability distributions produced by the different models. The more complex models may have produced more confident predictions for irrelevant or shorter passages. Since our method for selecting the final answer out of the multiple candidate answers does not account for this, the results from the more complex multi-passage models suffer.

Drilling further into what produced these results, we analyzed the performance of the models on the different query types (table 5). The first thing to note is that no one model performed the best for every answer type. While the baseline model produced the most consistent results, each model had its strong suit. In addition to this, one other trend to note is how the larger the dataset was, generally the better results. This is a consequence of our decision to train five separate answer extractors for the five query types. The description answer extractors produced much better results than the person type, despite the larger variation. We believe this is because of the fact that the description answer extractors had much more training data to learn from. To account for this, something to look into in the future may be methods involving transfer learning, having the main answer extractor look at every query type, and appending layers to the model for the particular query types.

4.4 Error Analysis

Through analyzing the answers given by our models, we have identified several sources of error, which are enumerated below. Some of these errors would be relatively simple to address given more time, while others would require a significant amount of further research to fix.

4.4.1 Lack of Comprehension

This is the most glaring problem with our model, and also the most difficult to solve. In many cases, our model produces probable but incorrect answers. It recognizes patterns but fails to comprehend enough about the question and context to produce correct answers. For example, figure 5 shows how the baseline model provides a probable but incorrect answer.

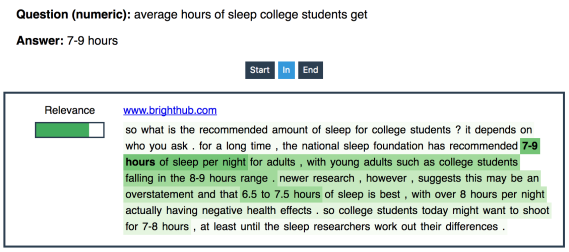

Figure 5: Answer from the baseline model for the question 'Average hours of sleep college students get.'

Bleu-1

Query Type	Baseline	Attention	Coattention	BiDAF
Person	16.32	19.05	**20.43**	13.86
Location	23.26	**27.61**	21.87	20.88
Entity	**25.84**	18.99	23.51	16.91
Numeric	17.27	15.72	**18.29**	14.43
Description	28.17	25.37	26.56	**28.96**

Rouge-L

Query Type	Baseline	Attention	Coattention	BiDAF
Person	**29.91**	26.06	26.21	24.21
Location	24.86	**27.40**	23.61	23.89
Entity	**22.51**	19.33	20.28	18.00
Numeric	29.53	28.49	29.61	**30.39**
Description	**26.27**	23.59	24.05	25.97

Table 5: Multi-passage results on individual query types. Numbers in bold highlight best results for each query type.

4.4.2 Answer Length

The correct level of detail for an answer is subjective and varies significantly across questions. For example, figure 6 shows three possible answers to the question 'What is the pay scale for medical secretary?'. The first, and shortest, includes just the median salary. This answer is acceptable, but doesn't fully answer the question. The next answer includes the average, median, minimum, and maximum salaries for medical secretaries, while the last includes the same information as well as hourly pay rates. In this case, the "correct" answer is entirely subjective and depends on what exactly the user is searching for. There is no simple fix to balance precision and recall like this; it is a complex problem that will be a persistent source of error for any question answering system.

4.4.3 Passage Relevance

When using only the passages which contain answers, our models achieve scores similar to those at the top of the MS MARCO leader board. When we introduce irrelevant passages, those scores drop significantly. Our model for predicting passage relevance generally selects passages that contain some answer to the provided question. The problem is that the passages that it selects may not be the same ones that were selected when the MS MARCO dataset was created. A more complex neural model might be able to achieve higher performance on passage relevance, but the variability of human answers will still be a significant source of error for any model trained on this dataset.

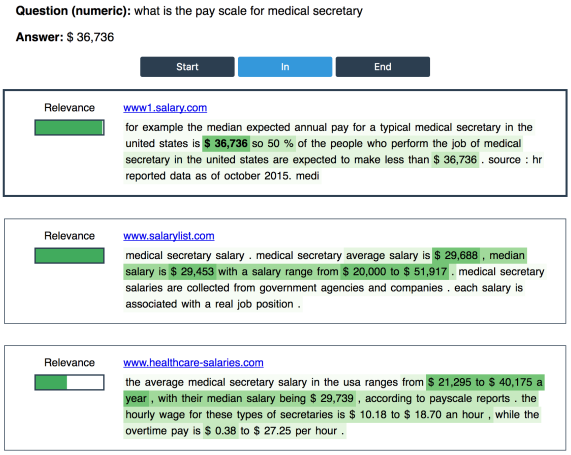

Figure 6: Answer from the coattention model for the question 'What is the pay scale for medical secretary?'

4.4.4 No Answers

In some cases, our model produces no answer whatsoever. When selecting an answer, we have the probability that the answer starts and ends on each token. Our generated answer starts with the most probable start token and ends with the most probable end token. When the most probable end precedes the most probable start, we return no answer. There are many simple fixes to this issue, some lie within the answer extractors themselves, other in the way we select the final answer.

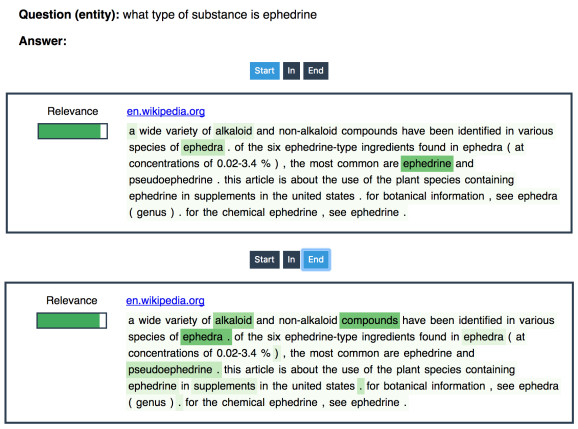

Figure 7: The attention model provides no answer to the question 'What type of substance is ephedrine?' because the most probable start token precedes the most probable end token.

4.4.5 Numeric Answers

Our model has little understanding of numeric values, which poses a significant problem since more than one quarter of MS MARCO questions are numeric. Our models have no understanding of scale and how units relate. We are completely dependent on how well the GloVe embeddings can encode how well units relate, which we suspect is not great. In addition to this, since the models are just looking for patterns within the passages, they do not understand if a particular number fits the appropriate numeric range for the answer. To solve this problem, we would have to investigate how to better encoding the relation between units and scale.

5 Related Work

Microsoft Research Redmond (Shen et al., 2016) achieved state-of-the-art performance in multiple reading comprehension datasets and performed well in the MS MARCO dataset. ReasoNet, their model, uses a multi-turn process, in which it repeatedly processes the context and the question after digesting intermediate information, with different attention weights in each iteration. Results show that multi-turn reasoning have generally outperformed single-turn reasoning, which basically utilizes an attention mechanisms to emphasize specific parts of the context that are relevant to the question. Moreover, ReasoNet uses reinforcement learning to dynamically decide when to terminate the inference process in reading comprehension tasks.

Microsoft Research Asia (Wang et al., 2017) currently have the best performance in the MS MARCO leader board. Their model, Prediction, uses an end-to-end neural network based on Match-LSTM and Pointer Net. Match-LSTM is used to predict textual entailment, where given two sentences, a premise and a hypothesis, it predicts whether the premise entails the hypothesis. Pointer Net generates an output sequence whose tokens come from the input. Microsoft Research Asia propose two ways to use Pointer Net: a sequence model, which outputs pointers to the tokens of the answer in the context, and a boundaries model, which outputs pointers boundaries (first and last tokens) of the answer in the context.

6 Conclusion

We proposed a machine comprehension model that is able to take in multiple passages and a query, and produce an answer to the question. Our model consists of two main modules, the answer-extractor and the passage relevance model. On the MS MARCO dataset, the model surpassed baseline performance posted by Microsoft Research and started to approach other competitors. With more time to refine the model, and experiment with different approaches, now that we have become acquainted with the field, we feel that we could really compete with some of the more advanced models from more experienced researchers.

With less than 10 weeks to ramp up and learn deep learning with no prior experience, learn TensorFlow, and put together an advanced question answering system, we feel we produced respectable results. Looking back, there are many things we would have done differently and routes that we wish we would have explored, but that is all part of the learning process.

6.1 Future Work

Obviously, we can greatly improve upon the current results. The first place to look is the passage relevance model. We believe this was the largest limiting factor for our results. Although TF-IDF works very well in practice, that is often for retrieving documents from giant, broad corpora, not for differentiating between very similar passages. Initial ideas would be to look into other datasets for some sort of training data for a model. Another great place to look for models would be the Fake

News Challenge. In that task, the model is given a headline and a body of text, and the model has to output if the two are related or not. Adapting an effective model from the Fake News Challenge to our task of finding relevant passages could lead to a great improvement in the results.

Another opportunity for improvement lies in the fact that many of the passages contain the correct answer. If each of these passages are passed through the answer extractor, ideally, the correct answer will be chosen multiple times. Exploiting this fact, we could cluster/merge similar answers to give greater confidence to persistent answers.

7.13% of the samples contained yes or no answers, this is something we chose to overlook for the time being. In order to score well on this dataset, this property of the data must be addressed. Initially, building a classifier to identify if the query is a 'yes or no' query would be the first step. From there, if the query is a 'yes or no' query, just run the answer extractor on the query and passage to get the answer. Once we have this answer, run it through yet one final classifier, giving it the query and answer, and it will output either 'yes' or 'no'. Although our current model answers many of these questions correctly, it gives the detailed answer instead of just 'yes' or 'no' and therefore receives no credit for the answer.

The final remark addresses what we believe the gold standard is for the model. Making quite the jump, the ideal model would be a language generative one. Instead of just pointing to where the answer is, the model would need to encode an understanding of the answer, and then generate the language to express this answer. This would take care of the many of the issue our model has, would take much more research on the topic to create even just a baseline model.

References

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. *CoRR* abs/1606.05250. http://arxiv.org/abs/1606.05250.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR* abs/1611.01603. http://arxiv.org/abs/1611.01603.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. *CoRR* abs/1609.05284. http://arxiv.org/abs/1609.05284.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *CoRR* abs/1611.01604. http://arxiv.org/abs/1611.01604.