Question Answering using modified QANet

Nisha Gandhi 111496495 **Tejas Naik 111425071** **Aditya Yele 111447727**

Abstract

Machine Comprehension is one of the most challenging and exciting areas of research in Natural Language Processing. The ability of the model to answer a given query about a paragraph is a good indicator of how well it is able to comprehend the data. In this project we study state-of-the-art models like BiDAF and QANet which recently have been quite successful on the Stanford Question Answering Dataset(SQuAD). We attempt to improve their performance by using feature engineering. We attempt to combine the architectures of the BiDAF and the QANet model to produce a novel architecture that contains the best of both worlds. We also provide a novel data augmentation method called Query Enrichment to tackle the problem of out-of-context or open-ended questions. We present an analysis and statistics of different combinations of these architectures and their evaluation metric scores on the SQUAD data set.

1 Introduction

Machine Comprehension is a challenging task that requires both the understanding of the Natural Language and the knowledge about the world. One of the ways to access machine comprehension is to tell the model to answer the a question about a paragraph. Building such models are of extreme importance since the ability to comprehend text leads to better search, thus saving on precious human time. For example, it could help improve customer service enquiries, search engines and AI assistants. It could help doctors/lawyers extract relevant information through tons of documents, saving their time for higher value work. Clearly, this is a non-trivial problem, because of the ambiguity and complexity in the language. Sometimes, the lack of supervised data for domain-specific knowledge could also pose an issue.

Traditional statistical NLP techniques have involved multiple steps of linguistic analysis and feature engineering, including syntactic parsing, named entity recognition, question classification, semantic parsing. Apart from the traditional methods, most of the recent approaches have been using Recurrent Neural Networks to handle sequential nature of the data. [1] One of the key advancements is the use of Attention mechanisms, which allows the targeting of a relevant area in the context paragraph.[2]. The model by Bahadanau et al[3] uses early summarization and calculates a fixed vector. The Bidirectional Attention flow mechanism by Seo et. al. [4] uses bidirectional LSTMs along with Attention to obtain Query-Aware text representation. Yu et. al[5] proposes an architecture which uses only Conv-Nets and self attention for the machine comprehension task.

Models described in Bahdanau et. al. and previous works usually involved early summarization leading to information loss, error propagation in each time step and were uni-directional. These issues were addressed by the BiDAF model. It was memory-less, and a bidirectional model which computed the attended vector at each time-step. Yu et al came up with the QANet model which was faster than the BiDAF model as it did not use RNNs. It was able to perform at the same performance level as the BiDaf with less training time. The architecture of the QANet can however be enhanced in a way to achieve better accuracy.

Existing models till now have not considered how the POS tags of the individual words in the context and question affect the performance. It is quite possible that a strong relationship may exist among the POS tags of the question and context. We use POS tags embeddings to see if they improve the accuracy of the model. We study the QAnet and the BiDaf architectures. The primary goal of QANet is to reduce the training time while

keeping the accuracy same. We hypothesize if it is possible to enhance the accuracy of the system by using LSTMs (used in BiDAF) in QANet, comprising a little on the training time. We combine the QANet and BiDAF architectures. The questions in the data-set and the research works that we use assume that all the information in the question is present in the context paragraph. We modify the training data as such to augment information about the unknown context to the question while training. For example "POTUS travels in Air Force One". If you ask the question "What is the name of the jet that Donald Trump travels in?". The answer should be Air Force One. Apart from this we try to experiment with different hyperparameters and with various combination of architectures to improve the accuracy as much as possible.

We assume the QANet model with 10000 iterations as our baseline. We calculate the F1 and EM scores of the baseline. We use the Stanford Question answering dataset by Rajapurkar et. al.[6]. SQUAD is a dataset of over 10000 records consisting of passages, questions and their answers. We calculate the F1 score and EM score for combination of different architectures like the QANet plus POS embeddings, QANet+stacked LSTM+POS embeddings, QaNet+Query Enrichment. We also experiment with different parameters like the number of hidden layers, dropout, learning rate for each of the combinations and find the best parameters.

The main outcomes of this project are:

1. We added POS tags as feature vectors to the input context and question embeddings.

2. Combined the BiDAf architecture with that of the QANet.

3. Augmented our data, with data from wikipedia for Query Enrichment.

4. Our analysis shows that adding POS embeddings to the input embeddings increases the F1 score by 1.

5. Query Enrichment decreases the F1 score and the EM score. The model performs poorly on the data with actual context in question.

6. Using stacked LSTMs increases the EM score by around 1. However using other bidirectional or increasing the LSTMs decreases the score, maybe because of over-fitting or high complexity. Also the training time increases significantly as opposed to our expectation of it only being a fraction larger than the training time of the original QANet.

7. Based on our work we conclude that investigating POS tags, architectural changes along with some modifications to Query Enrichment might be promising avenues for future work.

2 Your Task

The basic input for this task consists of the context paragraph (which can be any paragraph from sources like news articles, books and Wikipedia pages) and a question, which should be relevant to this particular paragraph and ask information that can be found in the paragraph.

State of the art models for solving this problems include the use of Recurrent Neural Networks (or some variations such as LSTM or GRU)[4], [1] or Convolutional Neural Networks[5] along with various implementations of the attention model[7], [3].

2.1 Baseline Model(s)

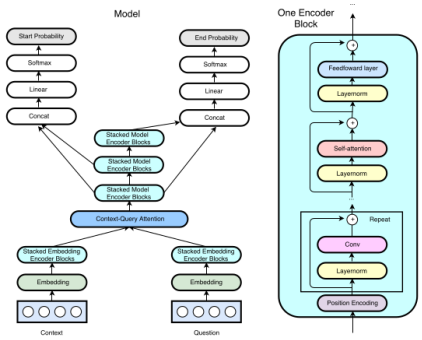


Figure 1: QA Net architecture

The baseline system we use is the QANet architecture by Yu et.al.[5]. The QANet consists of five major components: an embedding layer, an embedding encoder layer, a context-query attention layer, a model encoder layer and an output layer.

The architecture uses residual block, which is a block consisting of (local) Convolutional Networks, (global) self attention and feed forward layer. This is the core of the system.

Input Embedding Layer: The embedding of each word is obtained by concatenating the word embedding with the trainable character embed-

dings. Pretrained GloVe word vectors are used as word embeddings and are fixed during training. The concatenated vector is then passed to a two layered highway network.

Embedding Encoder Layer: The encoder layer passes the embeddings through the residual block. The depth of the Conv layer in the residual block is 7.

Context Query Attention Layer:It calculates the similarity between each pair of Context and Query words in matrix S. Then S is normalized using a softmax function. Then Context to Query Attention is computed using a trilinear function $f(q, c) = W_0[q, c, q.c]$ where W_0 is a trainable variable.

Model Encoder Layer: The output of the Context Query attention layer is the input to this layer.The architecture is same as the Embedding encoder layer with number of convolutional layers as 2 per-block with 7 such blocks.

Output Layer: It predicts the probability of each position in the context being the start and end of the answer span. The objective function is defined as the negative sum of the probabilities of the predicted distributions indexed by true start and end indices, averaged over all the training examples.
$L(\theta) = \frac{-1}{N}\sum_i^N [log(p^1_{y^1_i}) + log(p^2_{y^2_i})]$
where y are the ground truth starting and ending positions. At inference time the the predicted span (s,e) is chosen such that start and end probability are maximized.

2.2 The Issues

Existing models for Q/A do not use the part of speech features within the context and the questions. We imagine that there should be a relationship between the POS tag of the questions and the answer. For example when we ask the question "where is Stony Brook?". Most of the times the answer should refer to a location "In NY". This tells us that there is a strong correlation between certain words like where, why and the corresponding answer. So we explore this idea.

The QA net uses convolutions and self attention model that is described by Vaswani et. al[7]. to find the local and global interactions among the words of the context and the question. The BiDAF model by Seo et. al. [4]achieves this by using bidirectional LSTMs along witht context to query attention. While QANet achieves around the same accuracy we ponder over the question whether a combined QANet + BiDAF model would have a better accuracy than the individual models.

Another problem that we explore is that while all the questions in the data-set may correspond to some information in the passage. Is it possible for the model to answer relatively more open ended questions that do not have any direct information available in the context passage. For instance if The context paragraph states "Miami has 70k netflix subscribers" and we ask the question "Give the total number of subscribers for any one city in the US". The model should be able to answer such questions.

3 Your Approach

We implement our system in such a way so as to the address the issues stated above. Along with the word and character embeddings we pass POS tags information as an input to the encoder layer. We add LSTMs to the encoder layer and the model encoder layer at appropriate places in the residual block in the QANet. In order to be able to address more open ended questions we provide a way for Query enrichment.

3.1 Idea 1: Using POS tags

We find the Part-of-Speech tags of each of the words of Context passage and the Question. The tagger converts each of the sentences into tokens of words and finds out the appropriate POS tag. The POS tag information is calculated for each of the word in the question and the context and the answer. The information is represented as a 37-Dimensional one hot vector. These one hot vectors are then appended to the corresponding GloVe and character embeddings before they are passed on to the embedding encoder layer.

3.2 Idea 2: Query Enrichment

Many existing models do not consider the problem in which a query assumes implicit knowledge of a fact(like Miami is in the US). We try to address this issue of open ended questions by enriching the queries with extra information. While training we add the questions with necessary information about the important words such as nouns from the Wikipedia page for that noun using Wikipedia API.

3.3 Idea 3: Combining BiDAF

We consider if it is possible to improve the accuracy of the entire QANet system by using LSTMs.

As the QANet does not use any kind of RNN it is relatively faster but still achieves the same accuracy. We hypothesize that if we can combine the BiDAF architecture with current QANet architecture how would it affect the result. We also have to consider to keep the training time as less as possible and make sure to avoid over-fitting due to too much information extraction. This is why we use a unidirectional LSTM instead of a bidirectional LSTM. Also, since the convolution operation of the residual block of the QANet reduces the dimensions of the output, as a result, this should not increase the time taken for training significantly.

3.4 Implementation Details

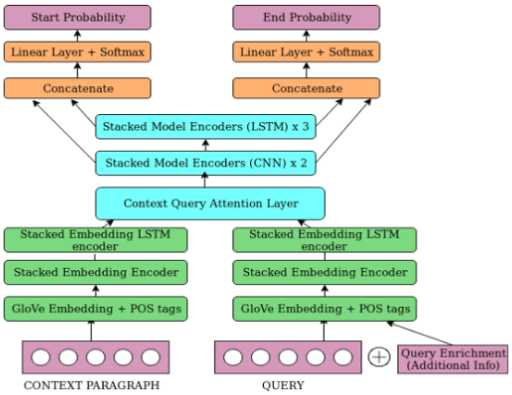


Figure 2: Modified QA Net architecture

We use the NLTK POS tagger which has more than 90% accuracy on various datasets [8]. The POS tags are represented as one hot vectors. We find the POS tags for each of the words in all of the context, answer and the question. Each word already has word embedding which is a GloVe vector of size 300 and character embedding which is a vector of size 64. So for each word we have a combined vector of 364. The POS tagger which is a one hot vector of dimension 37. Concatenating the POS tags makes the input embedding to be 401. This is then passed to a two layered highway network as was in the original model. At this point the dimension of the words are reduced to the number of hidden layers which is fixed.

These input embeddings are then passed on to the embedding encoder layer. This layer now in addition to the residual block that was present in the QANet also has the LSTM layer. As the purpose of the residual block was to gather local and global interaction, we expect the LSTM to capture a more comprehensive interaction of the local as well as the global phenomenon. The embedding encoder layer processes the context and the query separately. The output of this layer is then passed on to the Context to Query attention layer which measures the similarity between each pair of context and query words. This is the standard layer that is used in all other models including QANet, BiDAF, and others Xiong et. al [2], Bahdanau. et. al. [3]. The context to query attention layer uses the trilinear activation function.

We concatenate the context embeddings, the context to query and query to context attentions and then put all that information in a fixed size vector for each word using a CNN. The output then used by the Modeling encoder layer is passed to three layers consisting of the residual block plus the dynamic LSTM. The goal of the LSTM is to capture all the information that contain the global interaction of the output of the residual block. This output is then used by the output layer using the standard Question Answer setup described in the Baseline model.

For Query Enrichment, for each query in the training data, we first find important words (assuming important words are Nouns for simplicity) using NLTK POS Tagger. For each of this important words, we find relevant information from Wikipedia pages and append this information along with the query to enhance it, with the intention of enabling the model to better understand implicit relations in the passage.

4 Evaluation

We evaluate all the above described models approaches using F1 and EM scores, with the intention of finding the best possible model on the given data-set.

4.1 Dataset Details

We have used Stanford Question Answer Dataset (SQuAD)[6], which is one of the most popular datasets for Machine Reading Comprehension. Some details about this dataset:

1. It contains over 10,000 Question Answer Pairs, with 90k/10k for the training development data split and an additional 10k pairs for testing.

2. For each question, the answer is always contained in the paragraph, and is continuous,

i.e. the answer is a span in the context passage.

4.2 Evaluation Measures

We have used two evaluation metrics for model performance:

1. EM Score (Exact Match): The EM score is 1 if the predicted answer is exactly the same as the ground truth i.e. the start and end of the span is exactly the same. It is 0 otherwise.

2. F1 Score: This metric measures the portion of overlap between the predicted answer and the ground truth.

A high value of EM score as well as F1 score is a mark of a good model.

4.3 Baselines

Some implementation details of our baseline model (basic QANet architecture) are listed below:
1. Embedding Dimensions: 300 (GLoVE) + 64 (Char Embedding) = 364
2. Batch Size: 32
3. Number of iterations: 10,000
4. Hidden size: 96
5. Dropout: 0.1
6. Learning Rate: 0.001
7. Decay: 0.9999
8. Grad Clip: 5.0

Here, we chose the above hyperparameter values after experimenting and tuning the values of Hidden size, Dropout and Learning Rate (Highlighted values denote the chosen values).:

Effect of Learning rate on EM F1 score:

Learning Rate	EM Score	F1 Score
0.001	23.169	36.200
0.01	12.024	23.937

Effect of Dropout on EM F1 score:

Dropout	EM Score	F1 Score
0.1	23.169	36.200
0.2	11.561	22.735

Effect of Hidden size on EM F1 score:

Hidden Size	EM Score	F1 Score
96	23.169	36.200
50	9.252	20.206

4.4 Results

This results basically tells us three things:

1. Adding of POS tags helps the model in answering some questions in a better way, hence improving the EM score as well as F1 score by a margin of around 1.

2. Adding stacked LSTM encoder blocks in Embedding encoder as well as Modeling encoder does not perform as well as expected, however, it does give a high EM score when combined with POS tags.

3. Our proposed novel approach of Query Enrichment, performs poorly on the given training data.

Some Tensorboard visualizations for our best model:
EM Score: 66.23
F1 Score: 74.15
Number of Iterations: 30k iterations

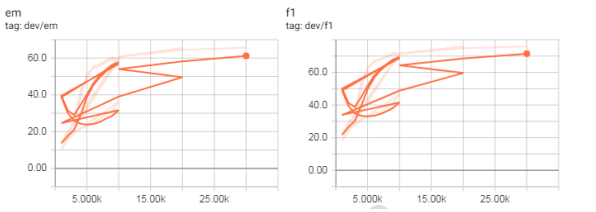


Figure 3: Tensorboard visualization graph for EM/F1 Score on Development set.

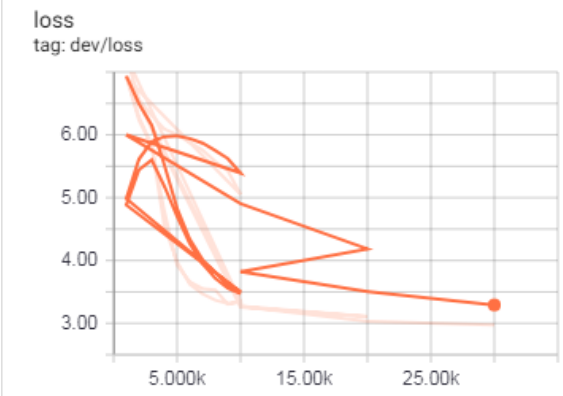


Figure 4:Tensorboard visualization graph for Loss on Dev set.

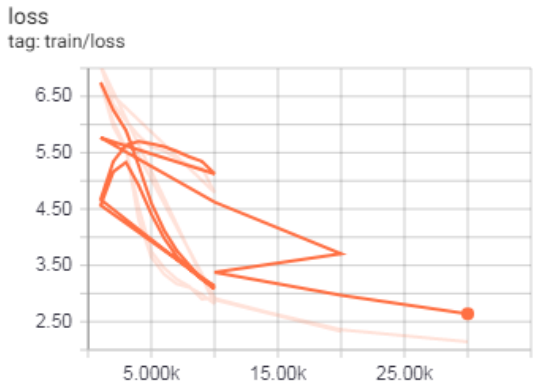


Figure 5:Tensorboard visualization graph for Loss on Training set.

4.5 Analysis

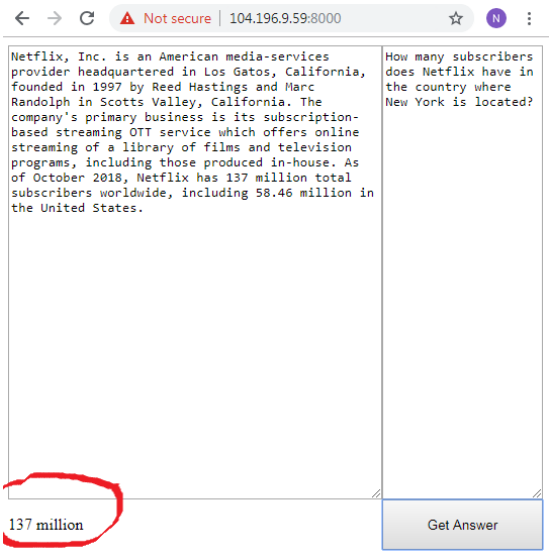
We hypothesize the following why some of our ideas did not work so well or worked well:

1. Firstly, our idea that POS tags contribute towards the relation between the query and the answer seems to work really well, confirming our guess.

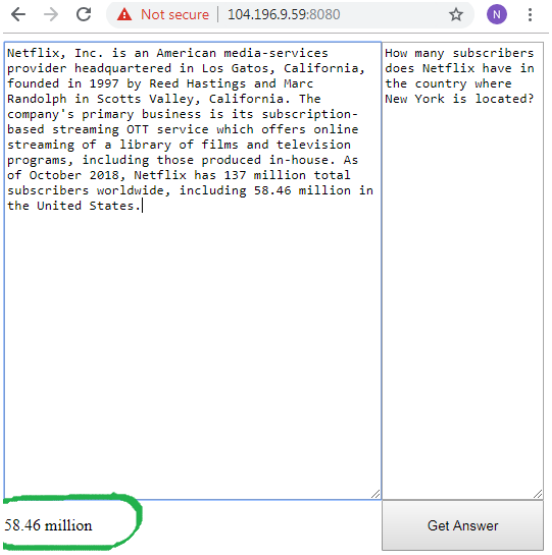
2. Secondly, our idea of combining BiDAF with QANet architecture does not work that well. We speculate that this might be due to the increased complexity of the model and the higher number of trainable parameters. We guess that with increased number of iterations, the LSTM model too should start working well, but again, this is a trade-off between accuracy and time. To confirm our guess, we tried to run the same model (QANet + stacked LSTMs) for increased number of iterations (30k), even if the accuracy does increase considerably (we were unable to test the actual metric scores due to unavailability of resources), however the training time shown increases exponentially.

Time	10k iter.	30k iter.
Train Time	2 hours	12 hours

3. Third, our idea of Query Enrichment by concatenating additional important information does not work that well either. We speculate that this might be due to overdose of information, the model is getting confused (overfitting) and hence does not perform well. Our guess is that not all the questions require additional information, straight-forward questions do not any help. Thus we need to weight how important the external information is according to the question. Weighted relevant external information might prove to be beneficial. To confirm our guess, we tried adding Query Enrichment only to our demo file (i.e. in run-time) and tested queries which actually required additional information. The results confirmed our guesses. The following screen-shots of our web-hosted demo show the difference before adding Query Enrichment and after adding.



Screenshot 1: Before adding Query Enrichment, the model does not know the relation between United States and New York.



Screenshot 2: After adding Query Enrichment, the model does knows the relation between United States and New York and hence gives the correct answer.

4.6 Code

[https://tinyurl.com/y7ndsj2v]

5 Conclusions

In this project, we studied various state-of-the-art models for Machine Comprehension. We modified the original QANet architecture and tried to improve on the accuracy (EM and F1 Score) by feature engineering, architectural changes and some Query Enrichment. Though these models are near the state-of-the-art score, we are still miles away from achieving near human intelligence on Reading Comprehension. The future scope might include having weighted additional information to enrich the queries, maybe trying out difference architectures to see what might work best, given that there is a trade-off between accuracy and training time. Also, altering the way we extract features can also be modified to add more relevance.

References

[1] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[2] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. In *International conference on machine learning*, pages 2397–2406, 2016.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[4] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[5] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

[6] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[8] Semih Yumusak, Erdogan Dogdu, and Halife Kodaz. Tagging accuracy analysis on part-of-speech taggers. *Journal of Computer and Communications*, 2(04):157, 2014.