QANet for Reading Comprehension

Stanford CS224N Default Project: IID SQuAD

Reetika Agarwal
Stanford University
reetika@stanford.edu

Rohit Kulkarni
Stanford University
rohitnk@stanford.edu

Ravi Rajagopal
Stanford University
rravi77@stanford.edu

Abstract

In this project, we built a non-pre-trained model for Question Answering task for SQuAD 2.0 dataset. Our primary contribution is a re-implementation of the original QANet architecture which outperforms BiDAF model. In particular, we showed that enriching input features by adding charcter-level embedding and other token features such as Part-of-Speech, Named Entity Recognition and Term Frequency can lead to significant improvements over the baseline architecture. We apply these enhancements to both BiDAF and QANet to study the generalizability of this approach. Additionally, we also study further extensions, such as modifying the output layer and data augmentation. Finally, we take an ensemble of our best performing models, which achieves a score of **72.5/69.9/77** F1/EM/AvNA score on the dev set and **69.73/67.22** F1/EM on the test set.

1 Key Information to include

- Mentor: Allan Yang Zhou

2 Introduction

Reading Comprehension is a challenging NLP task. The model needs to develop an understanding of the individual facts described in the context paragraph, distinguish between subtle nuances and lexical ambiguities, and synthesize information to correctly answer a question, or determine that the question is unanswerable. Given its difficulty, performance on Q&A tasks is commonly used to evaluate the efficacy of novel end-to-end architectures in natural language processing. A system that can read comprehension, and answer questions worded differently or distinguish tricky questions that are unanswerable, can be treated as a proxy for understanding human "text".

In recent times, Q&A is dominated by large-scale pre-trained language models (PLMs), such as ALBERT [1] and ELECTRA [2]. Even though PLMs have outperformed humans in the SQuAD leaderboard, they offer little insights into further architectural improvements. For this project, we choose non-PCE methods with a goal to implement architectural changes from scratch, explore model extensions and analyze the flow of information through a network.

Amongst non-pre-trained methods, QANet [3] is a top performer on SQuAD 1.1 leaderboard. QANet leverages recent trends in the field of NLP by combining multi-head self-attention as described in [4] and with the context-query attention layer as in BIDAF [5]. In addition to re-implementing QANet, we also explored modifications to the base architecture. We added character-level embedding in conjunction with the word-level embedding to address morphemes, mis-spelled or out-of-vocabulary words. For both models, we improve the semantic and structural representation of the context by including part-of-speech (POS), name-entity recognition (NER) and normalized term-frequency (NTF) tagging [6]. We also implemented data augmentation techniques, to identify the limits of the model, specifically using machine translation from English to French, and synonym word replacement. Our most notable improvements come from the QANet architecture, character embedding, and tagging.

3 Related Work

The Bi-Directional Attention Flow (BIDAF) model performed better than the then start-of-the-art when it was introduced in 2016. The success of the model relied on the enhancements to the Context-to-query attention mechanism. The BIDAF model performed well against the SQUAD 1.0 Dataset which had no unanswerable questions. However, with SQUAD 2.0, which contain unanswerable questions, the model faced challenges. Also BIDAF uses RNN, which are inherently sequential. This speed-bottleneck inhibits training on large-scale datasets and creates a ceiling on achievable performance.

QANet proposes a new model based only on convolution to capture the local structure of the text, as well as multi-headed self-attention to learn global interactions between pairs of words. This is a transformer-basel model which achieves superior result compared to the sequential RNNs. The model also used data augmentation to improve the accuracy against the SQUAD database.

Both BIDAF and QANet are closed, extractive and factoid answering model. Closed models rely on a given context to answer the question. It is an extractive model because the answer is a sub string of the context. It's a fact-based which means that the model can only answer factual questions but cannot provide any additional qualitative commentary.

DrQA (Document reader Question Answering) [6] extends the QA model to answer open domain questions. It uses a search component based on bigram hashing and TF-IDF matching. DrQA also uses feature engineering by adding manual tokens related to the context including POS, NER, NTF and EM (exact-match) tokens. This model achieved impressive performance when tested with Wikipedia as the knowledge source.

4 Approach

4.1 BiDAF

Baseline BiDAF is the baseline architecture [7] that consists of 5 layers:embedding layer, encoder layer, attention layer, modeling layer and output layer. For brevity's sake, the reader is referred to [5] for a full description.

Character level embedding The input embedding layer of the BiDAF architecture uses a 300-D pre-trained GloVE [8] word embedding. In addition to words, word-parts also carry meaningful information in English, and this can be encoded through character-level embedding. Characters are represented by 200-D learnable parameters. For each word, embedding for 16 characters are taken by either padding or truncating words, and passed through a 1d-convolutional layer with kernel size 5 and max-pooled along the word length dimension, to give a fixed size embedding x_c for each word. The final embedding is a concatenation of [x_w,x_c], where x_w=300-D and x_c=128-D in our experiments. The resulting word representation is passed through a projection and 2-highway network [9]. A highway network allows a fraction of the input to be transformed through a feed-forward network and the remaining fraction to pass through the network un-transformed. Intuitively, this allows the model to weight different parts of the encoding for different words.

We tried another novel approach to character embedding. Instead of using a stack of 1d-conv, ReLU, and Max pool, we use 2 stacks of 1d-conv, ReLU and max pool to create a deeper conv-net. We saw an improvement in character embedding when using two stacks of conv/pool layers instead of one.

4.2 QANet

Inspired by the current trend in NLP towards transformer-based models, we chose to implement QANet, which has higher speed and accuracy over BiDAF. We utilize the baseline code for word embedding, highway network, and context-to-query attention. The multi-headed causal attention is adapted from Assignment 5. We have implemented encoder and the output layer from the original paper, by utilizing open-source implementation of some standard blocks, such as sinusoidal positional embedding and depth-wise separable convolutions.

Input embedding We used the same word+char input embedding as in BIDAF. Through experiments, we observed that sharing the input embedding weights between the question and the context is very important for good performance. This not only enriches the representation of both, but also preserves the relationship between similar words in context and question, by transforming them similarly through the input embedding layer.

Encoder The core improvement of QANet comes from the encoder layer. Due to the lack of sequential information, the encoder uses sinusoidal positional encoding [4] with varying frequencies to encode relative positions of words in a paragraph. This is followed by three types of sub-blocks: several layers of depth-wise separable convolution, multi-head attention layer, and fully connected layer. In order to easily train deep networks, each of these sub-blocks (one of conv/self-attention/ffn) is wrapped inside residual block, and has layer normalization along the feature dimension [10]. For the feed-forward net, we follow the Transformer structure, by using 2 feed-forward layers with ReLU activation in the middle.

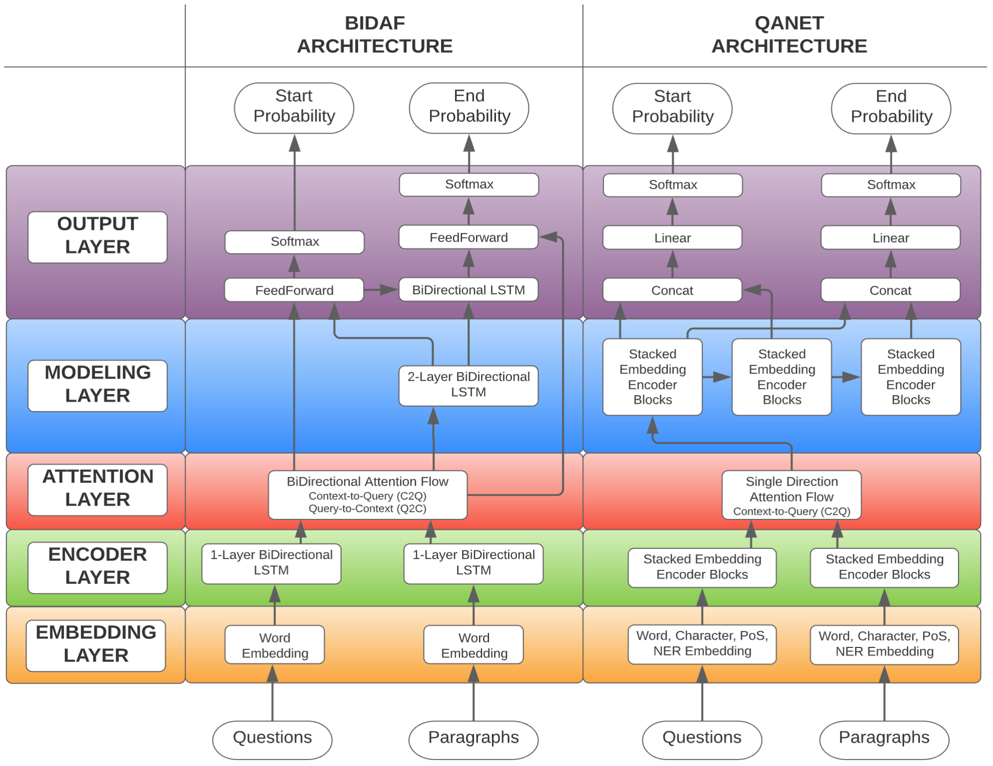

Figure 1: Both BiDAF and QANet has five main layers, main differences are in Encoder, Modeling and Output layers.

The same structure is used for both the encoder (2^{nd}) layer and the modeling (4^{th}) layer of the QANet architecture in Fig 1. For the 2^{nd} layer, we use a single encoder block with 4 convolutions per block, and a kernel size of 7. For the 4^{th} layer, we use 3 stacks of 5 or 7 encoder blocks, with 2 convolutions per block, and a kernel size of 5. We experiment with 1/4/8 heads for all multi-head attention layers. Similar to [3], we share weights between context and query embedding encoder layer, and all 3 model encoder layers.

Context-Query Attention This module is same as for both BiDAF and QANet model. We first compute a trilinear similarity matrix [5] S between each pair of context and query words. We then apply softmax along each row (along the query direction) to get matrix $\bar{S}$. We use this to compute context-to-query attention A = $\bar{S}.Q^T$. Next, to include query-to-context question, we apply softmax along each column (along the context direction) to get matrix $\bar{\bar{S}}$. This is then multiplied by context and transformed by $\bar{S}$ to get query-to-context attention $B = \bar{S}.\bar{\bar{S}}^T.C^T$. The final output is $[c, a, c \cdot a, c \cdot b]$ where c is the context, and a and b are a row of attention matrix A and B.

Output layer The output layer computes two vectors p_{start} and p_{end}, which is the probability of each position in the context being the start and end of the answer span respectively. If the output of the three model encoder stacks are M_0, M_1 and M_2, then, $p_{start} = softmax(W_0[M_0, M_1])$ and $p_{end} = softmax(W_1[M_0, M_2])$, where W_0 and W_1 are trainable weight matrices.

Inference Even though the p_{start} and p_{end} are generated independently, they're jointly used to determine output at inference time. We maximize the probability of $p_{start}(i).p_{end}(j)$ subject to the conditions $i \leq j$ and $j - i + 1 \leq L_{max}$. Here, L_{max} is the maximum allowed length of the answer and is kept at 15. This last step is a sanity check, the end should be prior to the beginning, and the answer shouldn't be too long, to decrease the likelihood of picking wrong answers.

4.3 Further extensions

Token Features The dataset has many factual Q&A, such as related to person name, place, location or time. Specifically tagging these features can improve the model's ability to pin-point relevant information. We use Python's spacy library to generate POS and NER tags for tokenized words. The spacy library supports 50 categories of POS (example- noun, punctuation, numeral, etc) and 26 categories of NER (example- person, org, work of art, date, etc). These are mapped to 16-D and 8-D embedding respectively. We also concatenate normalized occurrence frequency for each word in the context, resulting in a 453-D embedding for each word in the context. The question encoding is simpler, we only have 428-D concatenation of word+char embedding.

Through experimentation, we observe that it's important to share the input embedding weights and highway-layer between the context and question. This way same words in the context and query would have same embedding, keeping the one-to-one correspondence before attention is applied. In order to share the weights, we try two original ideas. In the first experiment, we pad zeros to the question encoding to match the dimension to context encoding (453-D). In the second experiment, we also encode the question with POR, NER and NTF tags. They both show improved performance over baseline with the first method being slightly superior.

Data Augmentation We tested different techniques for Augmenting both Questions and Paragraphs. The first technique was applying back-translation using different languages in order to rephrase the text and introduce diversity. We utilized several different languages: French, Chinese, Spanish, and Hindi. We also utilized Synonym Replacement, where we replaced any word that is either a Noun, Verb, Adjective, or Adverb with a 30% Probability.

Back-translation wasn't always successful, as shown in Figure 2a and often resulted in loss of the meaning of the original text. In order to fix this, we introduced some validation checks. The most effective check was to compare the Named Entities in the Original Text vs the Augmented Text. If any Named Entities were lost, we did not use the Translated Text. Figure 2b shows that once we added this validation tests, the Back Translation was very effective in adding Variety to our input Text.

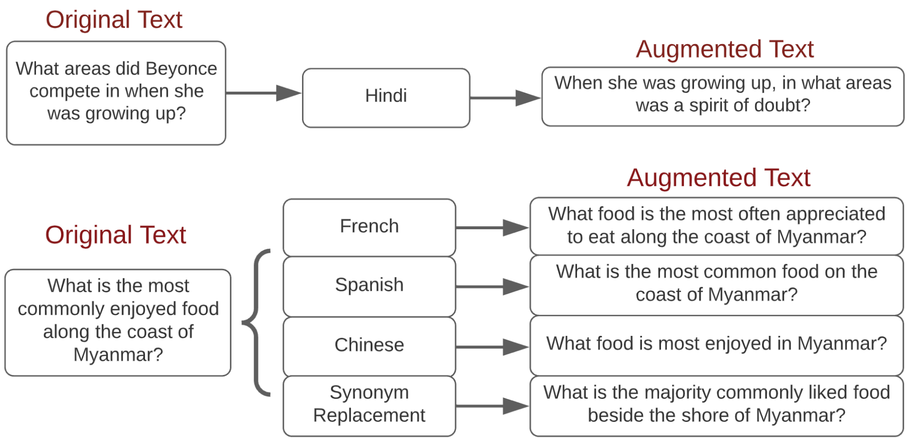
Original Text
What areas did Beyonce compete in when she was growing up?
Hindi
Augmented Text
When she was growing up, in what areas was a spirit of doubt?
Original Text
What is the most commonly enjoyed food along the coast of Myanmar?
French
Spanish
Chinese
Synonym Replacement
Augmented Text
What food is the most often appreciated to eat along the coast of Myanmar?
What is the most common food on the coast of Myanmar?
What food is most enjoyed in Myanmar?
What is the majority commonly liked food beside the shore of Myanmar?

Figure 2: **Fig 2a(top)** shows an example of back-translation without validation checks, **Fig 2b(bottom)** shows examples of back-translation with named entity validation checks

Output Layer Inspired by [11] and [12], we try two methods to condition the p_{end} token on p_{start}. Both methods are variants of the same underlying idea, we take the probability distribution of p_{start}, multiply with its input, and concatenate it with the input layer to the end token. More specifically, the below describes the two methods tried.

$$\begin{aligned}\textbf{Method A} :& X_{p1} = p_{start} \cdot ([M_0, M_1]) \\ & X_{p2} = [W_1.[M_0, M_2] X_{p1}] \\ & p_{end} = softmax(W_2.X_{p2})\end{aligned}$$

$$\begin{aligned}\textbf{Method B} :& X_{p2} = F.ReLU(W_2.[M_0, M_2]) \\ & p_{end} = softmax(W_3.[X_{p1} X_{p2}])\end{aligned}$$

Ensemble We ensemble multiple "weaker" models to build a "stronger model" with better accuracy. We use two techniques for ensembling - average probability and majority voting. For average probability, we iteratively load models, and take an average of the p_{start}, p_{end} across all the models. We use the averaged probabilities to determine the final answer. For majority voting, we look for consensus amongst models and pick the most commonly-predicted answer. For breaking ties, we give preference to the model with the highest score.

5 Experiments

5.1 Data

We used Stanford Question Answering Dataset SQuAD 2.0 [13] which includes unanswerable questions. The dataset consists of triplets of context, query, and an answer which is a contiguous span from the context. The training set consists of 130K examples, and the dev, test set consists of 6K examples each.

5.2 Evaluation method

Three quantitative metrics are used for evaluation: EM, F1 and AvNA. EM (Exact Match) checks for exact match. F1 is the harmonic mean of the precision and recall of the test value. AvNA scores the accuracy of getting answer vs no-answer correct.

5.3 Experimental details

5.3.1 BiDAF Training details

BiDAF model was trained with a hidden size of 100, batch-size of 80, a constant learning rate of 0.5 with AdaDelta optimizer. A weight decay of $3e{-}7$ and dropout of 0.2 was used to prevent over-fitting. As in original BiDAF paper, we used EMA (exponentially-weighted moving average) of 0.999 to improve predictability through ensembling which gives a gain of 1.4F1/EM scores. For including character embedding and input tokens, we train BiDAF with same hyper-parameters for 30 epochs.

5.3.2 QANet Training details

We trained all QANet models for 30 epochs, with varying batch sizes between 40-96 depending upon the model complexity, hidden size 128 and character dimension of either 64 or 200. For the hyper-parameters, we followed the original QANet paper and used an Adam Optimizer [14] $\beta_1 = 0.8$, $\beta_2 = 0.999$ and $\epsilon = 10^{-7}$. We used a learning rate warm-up scheme with an inverse exponential rise from 0.0 to 0.001 in the first 100 steps, and then maintain a constant learning rate of 0.001 for the rest of the training. Similar to BiDAF, EMA with decay rate 0.999 is applied to all trainable variables.

QANet has a tendency to overfit beyond 10-15 epochs, where the training loss decreases but the dev loss starts to increase. To avoid over-fitting, we apply L2 regularization on all trainable parameters, with $\lambda = 3e - 7$. We additionally use word dropout with $p_w = 0.1$ and character dropout with $p_c = 0.05$. For the module layers, we apply dropout, with $p_L = 0.1$, for every alternate conv layer, attention layer, fully-connected layer, highway layer, and context-to-query attention layer. Drop-out is also added in between every 2 model encoder stacks. Stochastic depth is often used to train deep ConvNets, shortening the network during training, and using the full deep network during test time [15]. Similar to the QANet paper, we also adopt this method (layer dropout) within each encoder layer, where each sublayer l drops out with a probability of $\frac{l}{L}(p_L)$, where L in the last layer in the encoder stack. We train the models on a Tesla V100 GPU, where it takes about 10-12 hours for 4M steps.

5.3.3 Input token features

We pre-process training, dev and test set to include the token features- POS, NER and NTF for both context and question. For a fair comparison, we train each of the BiDAF and QANet models, with their respective hyper-parameters, with and without token features.

5.3.4 Data Augmentation

We tested several combinations of Questions and Paragraph Augmentations. A few examples are: all questions augmented with several languages, paragraph augmented with French & Chinese, questions augmented with synonyms and paragraphs augmented with French. We couldn't combine too many techniques, as the data-set would explode drastically.

Synonym replacement didn't work very well. We used NLTK Wordnet Library to pick synonyms, however, we noticed that these synonyms weren't always very accurate. It was very hard to add some sort of validation on the synonyms picked, thus the augmented data was not very reliable.

On the contrast, Language Back-translation worked extremely well in augmenting the data. It added a lot of variety into the text. However, it took a very long time to prepare the data, especially with all of the input token features. As data grew from 2x-4x with augmented data, it would take over 8 hours to extract all of the features for the new data set, which is comparable to the training time.

Our two best methods were augmenting questions with French, Chinese, Spanish, and Hindi, and augmenting paragraphs with French.

5.4 Results

The table below show the results of all our experiments. Since we had limited submissions allowed for test set, the table below reports performance on the development set.

	Description	F1	EM	AvNA
	Single models			
BiDAF	baseline	61.29	57.99	68.07
BiDAF	char_emb	63.34	60.07	70.04
BiDAF	char_emb, 3token_feat	66.09	62.70	72.26
QANet	5Conv, 1head, 96d_model, 64char_emb	66.02	62.51	72.93
QANet	5Conv, 4head, 128d_model, 64char_emb	67.32	63.64	73.90
QANet*	5Conv, 8head, 128d_model, 200char_emb	68.51	64.93	74.84
QANet	7Conv, 8head, 128d_model, 200char_emb	67.98	64.21	75.00
QANet**	QANet*, 3token_feat, ques zero_pad	**69.44**	**65.89**	**75.77**
QANet	QANet*, 3token_feat, ques 3token_feat	68.29	64.71	74.81
QANet	QANet**, output layer Method A	68.37	64.86	74.71
QANet	QANet**, output layer Method B	68.86	65.4	74.95
QANet	QANet**, question augmented	66.99	64.38	72.81
QANet	QANet**, paragraph augmented	67.97	64.39	74.79
	Ensemble models			
QANet ensemble	average prediction	71.73	68.73	76.66
QANet ensemble	majority voting	71.4	68.55	76.17
All models ensemble	majority voting	**72.5**	**69.9**	**77**

Extending the baseline by adding **character embedding** improves the model by nearly +2.1/+3 F1/EM score. This is expected because even though the GloVe dictionary contains millions of tokens, the model still encounters an OOV token fairly often. By running 1d-conv on groups of characters, the model learns meaning of sub-words or groups of character, and infers more information about the inputs.

QANet far outperforms BiDAF model, achieving 68.51/64.93 F1/EM scores. The results show superiority of QANet architecture over BiDAF. We start with a simpler QANet model and gradually add complexity to understand the relative importance of each element to the overall performance. We see almost no benefit in increasing the model encoder stack from 5 to 7. On the contrary, increasing the number of heads from 1 to 8, the hidden size from 96 to 128, and character embedding from 64-d to 200-d, gives nearly +2.5F1 score. Qualitatively, we can infer two things from this observation: first, the model prefers to "pay attention" to multiple segments of the context to better predict the answer and second, the information flow bottleneck is along the hidden size dimension, not the vertical direction.

We also observe that QANet tends to converge faster than BiDAF, at around 20 epochs. It is a bigger model with many parameters and is prone to over-fitting, if continued to train for longer epochs.

The addition of **token features** extends the descriptive power of the inputs. Given the nature of the tasks being factual Q&As, parts-of-speech and named-entity-recognition may be particularly useful token features. For ex- BiDAF gets a huge gain of +2.75 F1 score just from token features.

The changes on the **output layer** did not show any noticeable improvement and in fact, slightly degraded performance. We suspect that since QANet is a deep network, only changing the output layer, without changing any of the preceding layers offers little gain. Further, part of the model encoder output is shared between p_start and p_end and our methods break symmetry which might adversely affect the accuracy.

Although the Language Back-Translation was successful in introducing a large variety into the Data-Set, we didn't see any big boost in the scores with **data augmentation**. We found that some of the approaches provided a small increase in score (+0.1-0.3 F1 Score).

Lastly, we ran an **ensemble** of the models, which gave better prediction than any stand-alone model. We compare the accuracy of average probability and majority-voting ensemble methods by running both of them on the same five models. Average probability is superior as it uses the probability distribution of p_{start}, p_{end} and has more fine-grained information.

For the final **test** submission, we ensembled a total of 12 models including QANet and BiDAF and achieved a score of **69.73/67.22** F1/EM. The difference in scores can be explained by shifts in data distribution as well as ensembling based on the performance on the dev set.

6 Analysis

AvNA We observe that one of the weaknesses of the model is trying to guess an answer for the questions with no answer, given the context paragraph. The dev set has more unanswerable questions than answerable questions, however the models over-predicts answers. As shown in the confusion matrix in Table 2, the model has 74%precision and 76.5% recall for answer vs no-answer.

		Predicted		
		No Ans	Ans	Total
Ground truth	No Ans	2370	773	3103
	Ans	667	2181	2848
		3037	2954	

Table 1: Answer vs No Answer Confusion Matrix for the ensembled model on development set

Analysis on Question type To further study the model's strengths and weaknesses, we classify its performance on types of questions. For questions with more than one match, we pick the first question word. Figure 3a shows that the F1 and EM scores of the best ensemble model. We see that the performance declines when none of the conventional question words appear in the question, as shown by the poor F1 score achieved in the "Other" category. Further, the model does better on

"Whose", "When" and "Who" questions which normally have an named-entity answer. It does poorly on reasoning question such as "Why" or "How".

Next, we attempt to answer why input token features help. We plot the performance of a single model QANet, with and without token features, in Figure 3b. As can be seen, the performance in categories like "Which", "Whose", "When" and "How" improves with POS, NER and TF, whereas the performance in "Other" category drops.

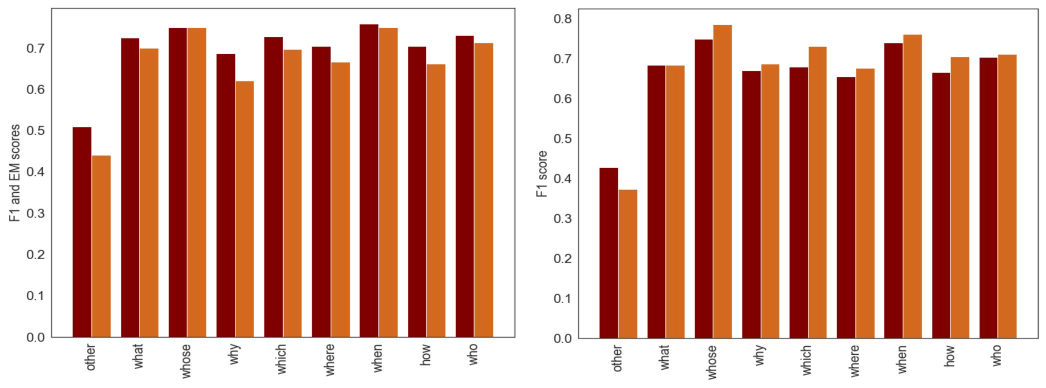

Figure 3: **Fig 3a (Left)**: Performance of the ensemble model on different question types [Red/Orange: F1/EM], **Fig 3b (Right)**: Comparison of F1 score of a single model QANet with and without token features [Red/Orange: No token features, with POS/NER/NTF]

Manual Error analysis We picked a handful of examples for manual error analysis. The first two examples in Table 2 in the Appendix are instances of tricky questions. The models give a closely-related but factually incorrect answer. For ex- context contains the phrase, "Normandy, a region in France", question is "What is France a region of", and the model answers "Normandy". As a second example, context contains the phrase, "The Norman dynasty had a major political, cultural and military impact on medieval Europe', the question is "What type of major impact did the Norman dynasty have on modern Europe", and the model answers "political, cultural and military". Notice the change in one word medieval to modern which confuses the model. These are hard examples, and the model fails to distinguish between subtle change of words.

The second category of error is pattern matching. Here, the model learns that the answer to "who" should be a person, "when" should be time, and "where" should be a place. It tries to look for the specific entity in the paragraph without understanding the meaning of the text. The third example in Table 2 shows for a question that begins with "When ..", the model answers "9th century" which is the only time mentioned in the paragraph, even though it is incorrect.

7 Conclusion

In this project, we have implemented and explored popular Question-Answering models like BIDAF and QANet. We have found ways to enrich the input representation by adding character embedding and token features. While these methods were successful, other approaches such as data augmentation and modifying the output layer did not yield much additive improvements.

Based on our experiments, it is clear that the transformer-like structure QANet is more superior to BIDAF. Our best single-model result is 69.44/65.89 F1/EM score on the dev set. Our best ensemble results are **72.5/69.9 F1/EM** on the dev set and **69.73/67.22** F1/EM score on the hidden test set.

For future work, we would like to experiment with adding separate head for answer vs no-answer to improve the model's ability to differentiate between those cases. Another interesting areas of study could be exploring Transformer-variants, such as Transformer XL and Reformer, and extending the model for open domain question-answering.

References

[1] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[2] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[3] 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.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

[6] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[7] CS224N. `http://web.stanford.edu/class/cs224n/project/default-final-project-handout-squad-track.pdf`.

[8] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[9] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[10] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[11] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *arXiv preprint arXiv:1608.07905*, 2016.

[12] CS224N Win 2021 Default Project. `https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1214/reports/final_reports/report269.pdf`.

[13] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016.

A Appendix

A.1 Attention analysis

We plot the heat-map of the context-to-query attention layer to analyze the interactions. Figure 2a shows an example of an attention map where the model predicts the correct answer. The word "century" is mapped to "11th centuries", which is the part of the correct answer to the question. In Figure 2b, the model predicts the wrong answer "Norse" instead of "Catholicism". The attention heat-map provides some indication that the word religion in the question was mapped to the same word in the context, and the model guessed the preceding word to the be answer. In general, we notice the model tends to learn high-level patterns like word-matching between context and picking the prior/after word as the probable answer.

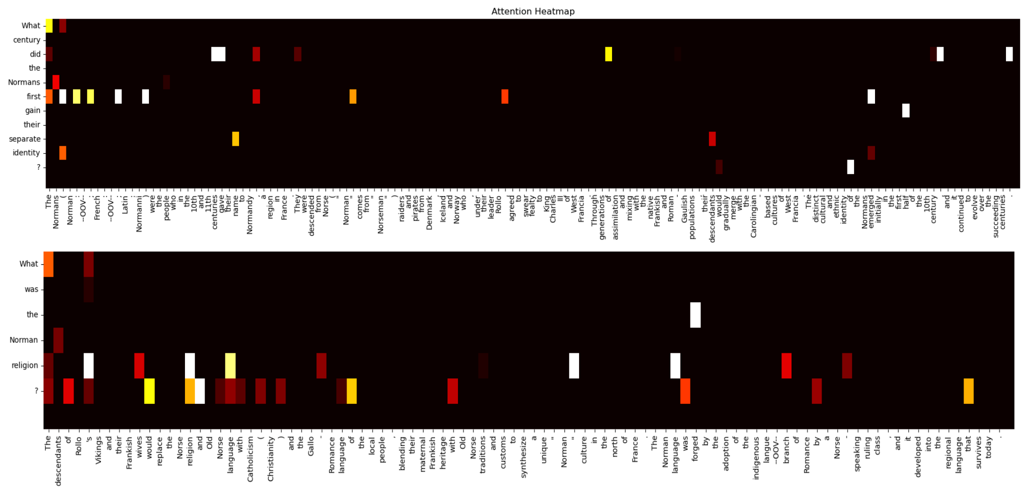

Figure 4: a. shows an example with the attention is on the correct answer and b. shows an example when the attention is on an incorrect answer on 2 samples picked from the development set

A.2 Training curves for selected models

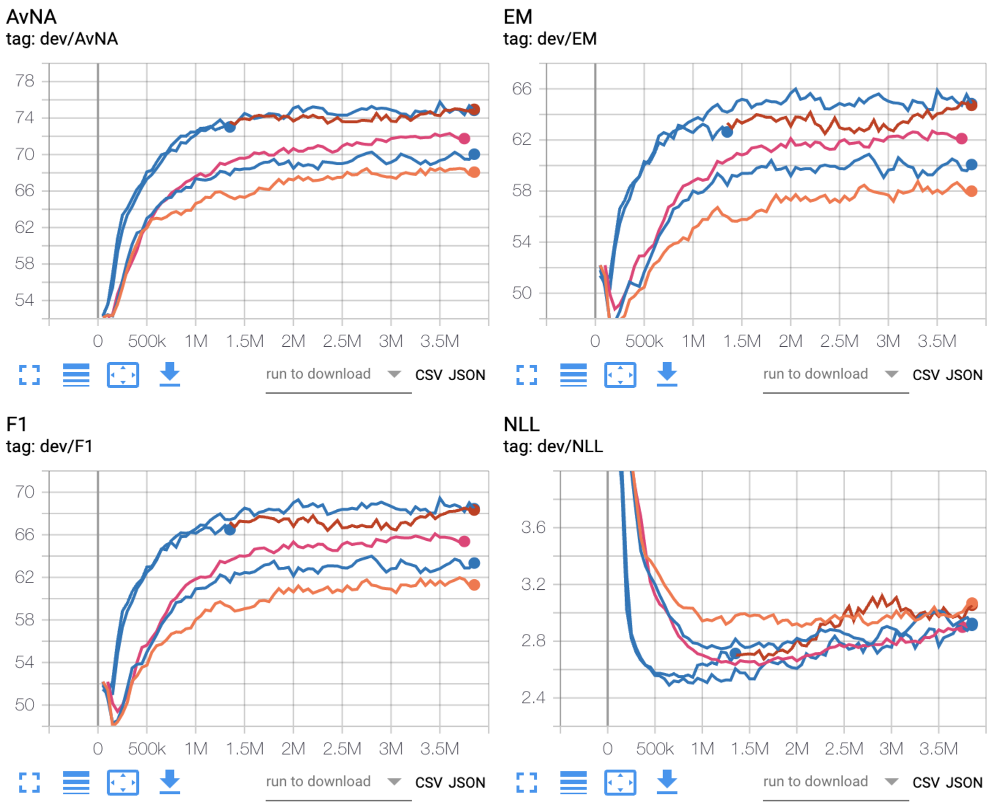

Figure 5: Selected models in order of increasing F1 score : baseline(orange), baseline+ char_emb (blue), baseline+char_emb+token_feat (pink), QANet(red), QANet+token_feat(blue)

A.3 Manual analysis of few incorrectly classified answers

Paragraph	The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries gave their name to **Normandy, a region in France**. They were descended from Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland and Norway who, under their leader Rollo, agreed to swear fealty to King Charles III of West Francia. Through generations of assimilation and mixing with the native Frankish and Roman-Gaulish populations, their descendants would gradually merge with the Carolingian-based cultures of West Francia. The distinct cultural and ethnic identity of the Normans emerged initially in the first half of the 10th century, and it continued to evolve over the succeeding centuries.'
Question	'What is **France a region** of?
Gnd truth	No answer
Answer	Normandy
Paragraph	**The Norman dynasty had a major political, cultural and military impact on medieval Europe** and even the Near East. The Normans were famed for their martial spirit and eventually for their Christian piety, becoming exponents of the Catholic orthodoxy into which they assimilated. They adopted the Gallo-Romance language of the Frankish land they settled, their dialect becoming known as Norman, Normaund or Norman French, an important literary language. The Duchy of Normandy, which they formed by treaty with the French crown, was a great fief of medieval France, and under Richard I of Normandy was forged into a cohesive and formidable principality in feudal tenure. The Normans are noted both for their culture, such as their unique Romanesque architecture and musical traditions, and for their significant military accomplishments and innovations. Norman adventurers founded the Kingdom of Sicily under Roger II after conquering southern Italy on the Saracens and Byzantines, and an expedition on behalf of their duke, William the Conqueror, led to the Norman conquest of England at the Battle of Hastings in 1066. Norman cultural and military influence spread from these new European centres to the Crusader states of the Near East, where their prince Bohemond I founded the Principality of Antioch in the Levant, to Scotland and Wales in Great Britain, to Ireland, and to the coasts of north Africa and the Canary Islands.
Question	What type of major impact did the Norman dynasty have on **modern Europe**?
Gnd truth	No answer
Answer	cultural and military
Paragraph	The English name "Normans" comes from the French words Normans/Normanz, plural of Normant, modern French normand, which is itself borrowed from Old Low Franconian Nortmann "Northman" or directly from Old Norse Norðmaðr, Latinized variously as Nortmannus, Normannus, or Nordmannus (recorded in Medieval Latin, **9th century)** to mean "Norseman, Viking"
Question	**When** was the French version of the word Norman first recorded?
Gnd truth	No answer
Answer	9th century

Table 2: Manual Analysis of common mistakes made by the ensembled model on the development set