Question Answering on SQuAD 2.0 using BiDAF and QANet

Stanford CS224N Default Project (IID SQuAD Track)

TA Mentor: Eric Mitchell

Matthew Kaplan
Department of Computer Science
Stanford University
mkaplan1@cs.stanford.edu

Jacklyn Luu
Department of Biomedical Informatics
Stanford School of Medicine
jacklynl@stanford.edu

Jane Boettcher
Department of Computer Science
Department of Linguistics
Stanford University
jmsb@stanford.edu

Abstract

In this project, we explored the BiDAF model [1] and QANet [2] to tackle SQuAD 2.0. First we improved the BIDAF model by implementing character level embeddings. Next, we implemented our own version of QANet, a non-recurrent model whose encoder consists exclusively of convolution layers (which model local interactions), self-attention layers (which model global interactions), and a multi-layer perceptron feed-forward net. We experimented with QANet's architecture by varying the number of encoder blocks in the model encoder block stack. We found that our best model (F1 score of 66.05 and an EM score of 62.87 on the test set) came from following the original authors' QANet architecture specifications (namely, 7 encoder blocks in the model encoder block stack and the use of layer dropout). Finally, we performed qualitative analyses which showed that our models performed worse on questions that asked for descriptions and and better on questions that had more distinctive expected answers such as date questions.

1 Introduction

Question answering (QA) aims to answer questions about a passage or document in natural language. The ability to read text and answer questions about it is a challenging task for machines, as it requires not only understanding of natural language and the real world, but also deeper reasoning abilities.

The Stanford Question Answering Dataset (SQuAD) is a large, high-quality dataset created specifically for machine recognition of text. It offers a large number of questions and answers created by humans through crowdsourcing. Unlike traditional datasets, SQuAD answers do not come from a set of candidate answers and have variable length; therefore, end-to-end neural architectures cannot rely on candidate answers and assume that the answer is a single token. Furthermore, SQuAD 2.0 extends the original SQuAD with unanswerable questions. To perform well on SQuAD 2.0, QA systems must not only answer questions when possible, but also determine if no answer is supported by the paragraph and then refrain from answering.

Recent years have demonstrated an ongoing trend towards attention-based models for natural language processing tasks [3]. In this work, we explored two models that utilize attention:

bidirectional attention flow (BiDAF), which uses context query attention, and QANet, which utilizes self-attention. Our main focus in this project was to implement the QANet model and understand its strengths and weakensses towards SQuAD 2.0.

2 Related Work

Previous attempts have been made by several authors to solve the SQuAD dataset, including through the use of BiDAF and QANet [1] [2]. While these models have very distinct architectures, one common layer worth discussing is the attention layer.

Attention allows a system to focus on a targeted area within a context paragraph. Prior to BiDAF, attention mechanisms typically had one or more of the following characteristics: First, by summarizing the context into a fixed-size vector, the computed attention weights were used to extract the most relevant information from the context for answering the question. Second, in the text domain, they are often temporally dynamic, whereby the attention weights at the current time step are a function of the attended vector at the previous time step. Third, they are usually uni-directional, wherein the query attends on the context paragraph [1].

BiDAF offered the following modifications to previous attention paradigms. First, the attention layer was computed for every time step, and the attended vector at each time step was allowed to flow through the next modeling layer. This reduced the information loss caused by early summarization. Next BiDAF used a memory-less attention mechanism, where the attention at each time step is a function of only the query and the context paragraph at the current time step and does not directly depend on the attention at the previous time step. Finally, BiDAF used attention mechanisms in both directions, query-to-context and context-to-query, which provided complementary information to each other. Through these modifications, BiDAF had outperformed all previous approaches on the SQuAD test set leaderboard at the time of submission in 2016 [1].

One major disadvantage of BiDAF was that it relied on recurrent neural networks (RNNs), which are difficult to parallelize and slow to train. Yu et al [2] offered a solution to this problem through QANet, a transformer based model that does not use RNNs. Instead, QANet encodes the question and context separately using convolution to model the local features and self-attention [3] to process global interactions. The authors estimated that QANet was 3x to 13x faster in training and 4x to 9x faster in evaluation compared to previous models like BiDAF. QANet surpassed all existing approaches to SQuAD at the time that it was published [2]. The success of QANet motivated us to implement our own version of the model for this project. Our work builds upon the original QANet paper by offering in depth analyses on what types of questions, ideas, and categories does the model perform well and poorly on.

3 Approach

3.1 BiDAF

We first aimed to improve the Bidirectional Attention Flow (BiDAF) model, whose architecture is described in [1]. The baseline BiDAF model used only word-level embeddings, which were initialized from pre-trained 300-D GloVe word vectors [4]. To immediately improve this model, we implemented support for character level embeddings completely from scratch. To implement the use of character level embeddings, we used CharCNN as described by [5] to append 200 dimensional character embeddings (for each word) to the word embeddings. The character level embeddings allowed the model to condition on the internal structure of words and better handle out-of-vocabulary words.

3.2 QANet

For our final model, we implemented QANet entirely from scratch. QANet is a transformer-based model created by Yu et al [2]. The core feature of this model architecture is the encoder blocks, which include a positional encoding layer as described by the original authors of the

transformer model [3], followed by a series of layer normalizations, convolution, self-attention, and feed-forward net layers to process the context and query.

QANet first processes the context and query embeddings in an identical way as BiDAF (using CharCNN to process the character embeddings for each word into vectors of length 200, then appending these character embedding vectors to the word embeddings) [2] [5]. It then applies a projection layer and highway encoder to the embeddings similar to BiDAF [1] [2].

Then, the embeddings for both the context and query are sent through separate encoder blocks: 8 attention heads are used for self-attention, and a 2-layer multilayer perceptron (MLP) consisting of linear layers with a ReLU activation make up the feed-forward net. Similar to BiDAF, context-query attention is performed to generate embeddings for a single sequence the length of the context. This layer allows attention to flow both from the context to the query and the query to the context.

Next, the output of the context-query attention is passed through 3 stacks of 7 model encoder blocks, each block with the same design as the encoder blocks used to process the context and query embeddings. Following the original QANet authors, the weights in each stack are shared, meaning the sequence is sent through the same 7 encoder blocks repeatedly [2].

Finally, the outputs from each of the 3 stacks of encoder blocks are concatenated and projected through a linear layer before the softmax is taken to generate the start and end point probabilities.

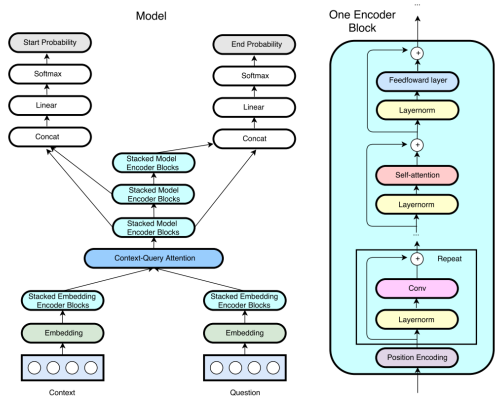

Figure 1: The QANet Architecture as described by Yu et al. [2]

4 Experiments

4.1 Data

For this project, we use the SQuAD 2.0 dataset provided by the CS224N Course Staff [6]. The dataset includes examples of passages (contexts) and questions (queries), and the task of the QA system is to find the answer to a given question in the provided passage (if it exists). As mentioned by the CS224N course staff, paragraphs in the SQuAD 2.0 dataset are from Wikipedia, and specific to SQuAD 2.0, half of the questions do not have an answer [6]. If the questions are answerable, the answers are found directly from the paragraph; therefore, our model simply needs to predict the start and end points of the answer in the provided paragraph as an output.

4.2 Evaluation method

For our evaluation, we use two metrics: an Exact Match (EM) score and an F1 score [6]. The EM score simply looks at whether the model's output matches the labelled answer exactly. If so, the EM score is 1, else, the EM score is 0. The F1 score takes the mean of the model

answer's precision (measuring whether the model's answer lies within the true answer), and recall (the fraction of words from the true answer in the model's answer). The maximum F1 and EM scores are used across 3 human-provided labels for each question, and these metrics are averaged across the entire evaluation dataset (train, development, or test) to get the final scores.

4.3 Experimental details

For this project, we trained 4 total models: the baseline BiDAF model, BiDAF with character embeddings, a simple QANet model, and a full QANet model. For all models, we used the Adadelta optimizer with a learning rate of 0.5 and no weight decay. We used a constant learning rate throughout training.

The original BiDAF baseline model used only word embeddings. Therefore, we implemented character embeddings to the baseline BiDAF model. Both baseline BiDAF model and BiDAF with character embeddings took roughly 3.5 hours to train.

For our simple QANet model, we followed the high-level model architecture but this model had the following limitations: the stack of model encoder blocks consisted of only 5 blocks, instead of the 7 used by the original authors. This allowed for training with batch sizes of 32, meaning we could more quickly check whether our implementation was sufficient. Furthermore, this model did not use layer dropout in the encoder blocks as described by the original authors [2].

The full QANet model builds upon our simple QANet model by including layer dropout in the encoder blocks as described by the original QANet authors [2], and using 7 blocks in the model encoder block stack. Additionally, to make the model more expressive, we added a ReLU function after each convolutional layer in the encoder blocks. We trained on a batch size of 16, as this was needed to fit the model in CUDA memory.

For QANet, we used a dropout rate of 0.1, and included dropout between all model layers in addition to the layer dropouts within the encoder blocks. Training the simple QANet model took close to 8 hours, while the full QANet model took roughly 13.5 hours to train.

4.4 Results

Table 1: Performance of various models on SQuAD 2.0 development and test sets

Model	Dev F1	Dev EM	Test F1	Test EM
BiDAF (baseline)	61.38	57.94	-	-
BiDAF + character embeddings	64.83	61.42	-	-
QANet Simple	65.19	61.94	63.971	60.541
QANet Full	**68.15**	**65.13**	**66.054**	**62.874**

The results of all our individual models on the SQuAD 2.0 development and test sets are presented in Table 1. Adding character embeddings to the baseline BiDAF model significantly improved performance. Given that character embeddings allow the model to better decipher subword meanings, we expected the addition of character embeddings to improve the baseline BiDAF model.

As seen on the public leaderboard for SQuAD, we expected that QANet should outperform BiDAF; however, we observed that our results did not reach the statistics reported by the original authors [2]. We hypothesize that this difference in performance is due several reasons. The original authors of QANet trained their model using original version of SQuAD, while we are using SQuAD 2.0. SQuAD 2.0 contains unanswerable question written adversarially by crowdworkers to look similar to answerable questions, so it follows tha performance was decrease using this newer data set. In addition, our use of a simple 2 layer MLP for the feed-forward net versus a more expressive feed-forward net in the encoder blocks and our lack of use of data augmentation which the original authors used to better generalize the training set could help explain why our results did not match that of the original authors [2].

Finally, we found that our full QANet model outperformed the simple model. By adding more encoder blocks, an extra ReLU function, and layer dropout to the full QANet model, the model would become both more expressive and would better generalize compared to our simple QANet model. Thus, we expected better performance on both the dev and test sets with our full QANet model than all other models.

5 Analysis

Performance breakdown by question word: In order to better understand the performance of our models, we took a closer look at its performance on different types of questions. The first way we were able to operationalize this goal was simply by looking at how the model performed on different question words. We looked at the performance of all four of our models on questions with the following seven words: "who", "when", "where", "what", "why", "which", "how". We also had a category "other" for questions that didn't have any of these words. As seen in Figure 2, we found that across all models, "when" questions were the easiest to answer, while "why" questions were the hardest.

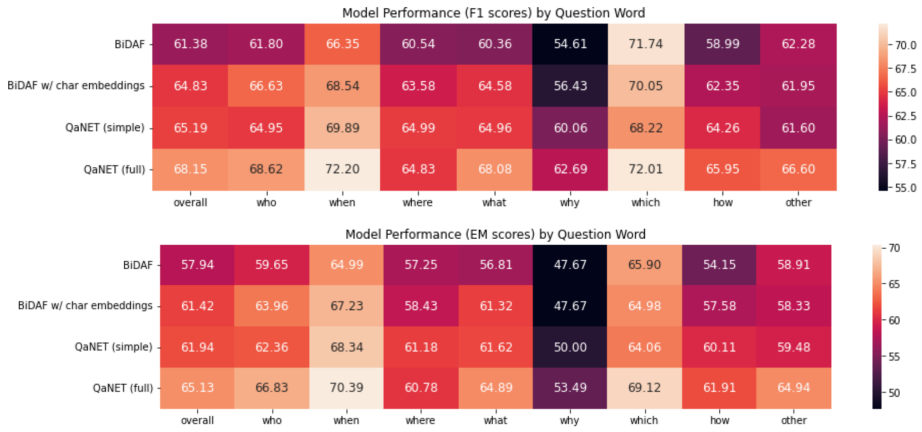

Figure 2: Performance of our models broken down by question words

For example, in Example 1, our full QANet model was able to answer the question "When did Dyrrachium fall to the Normans?" correctly. In the context given to our model, there are clear markers for a date, such as use of a month word and a 4 digit year, making the date characteristic much more identifiable. Further, "when Dyrrachium was betrayed to the Normans" from the context closely matches the wording of the question "When did Dyrrachium fall to the Normans?". This use of "when" in context is possible because "when" can be used as a relative pronoun, as it is in this context. This match seems especially important, as our model was able to return February 1082, when another date was also available. While this is just one example, "when" questions across the board have this more expected structure of dates and also have the possibility of "when" being used as a relative pronoun that might boost these questions' performance.

Example 1	
Question	When did Dyrrachium fall to the Normans?

Context	Robert Guiscard, an other Norman adventurer previously elevated to the dignity of count of Apulia as the result of his military successes, ultimately drove the Byzantines out of southern Italy. Having obtained the consent of pope Gregory VII and acting as his vassal, Robert continued his campaign conquering the Balkan peninsula as a foothold for western feudal lords and the Catholic Church. After allying himself with Croatia and the Catholic cities of Dalmatia, in 1081 he led an army of 30,000 men in 300 ships landing on the southern shores of Albania, capturing Valona, Kanina, Jericho (Orikumi), and reaching Butrint after numerous pillages. They joined the fleet that had previously conquered Corfu and attacked Dyrrachium from land and sea, devastating everything along the way. Under these harsh circumstances, the locals accepted the call of emperor Alexius I Comnenus to join forces with the Byzantines against the Normans. The Albanian forces could not take part in the ensuing battle because it had started before their arrival. Immediately before the battle, the Venetian fleet had secured a victory in the coast surrounding the city. Forced to retreat, Alexius ceded the command to a high Albanian official named Comiscortes in the service of Byzantium. The city's garrison resisted until February 1082, when Dyrrachium was betrayed to the Normans by the Venetian and Amalfitan merchants who had settled there. The Normans were now free to penetrate into the hinterland; they took Ioannina and some minor cities in southwestern Macedonia and Thessaly before appearing at the gates of Thessalonica. Dissension among the high ranks coerced the Normans to retreat to Italy. They lost Dyrrachium, Valona, and Butrint in 1085, after the death of Robert.
Possible answers	['1082', 'February 1082', 'February 1082']
QANet (full) answer	February 1082

In contrast, "why" cannot be used as a relative pronoun, and the expected format of an answer to a "why" question does not follow any particular pattern. For example, in Example 2, QANet model predicted their was no answer when there was an answer to the question "Why was there a depreciation of the industrialized nations dollars?". First, the indicator of the correct answer in the context was "the result was a depreciation of the dollar and other industrialized nations' currencies", which far less closely matches the wording of the question "Why was there a depreciation of the industrialized nations dollars?". This seems in part because "why" cannot be used as a relative pronoun and isn't as closely attached to the desired answer, and also because the answer phrase seems more paraphrased. Second, the given correct answers, which are essentially some variation of "industrialized nations increased their reserves", seem like any other descriptive phrase in the passage in form. Thus we can see how "why" questions might be particularly difficult for our model to handle.

Example 2

Question	Why was there a depreciation of the industrialized nations dollars?
Context	On August 15, 1971, the United States unilaterally pulled out of the Bretton Woods Accord. The US abandoned the Gold Exchange Standard whereby the value of the dollar had been pegged to the price of gold and all other currencies were pegged to the dollar, whose value was left to "float" (rise and fall according to market demand). Shortly thereafter, Britain followed, floating the pound sterling. The other industrialized nations followed suit with their respective currencies. Anticipating that currency values would fluctuate unpredictably for a time, the industrialized nations increased their reserves (by expanding their money supplies) in amounts far greater than before. The result was a depreciation of the dollar and other industrialized nations' currencies. Because oil was priced in dollars, oil producers' real income decreased. In September 1971, OPEC issued a joint communiqué stating that, from then on, they would price oil in terms of a fixed amount of gold.
Possible answers	['industrialized nations increased their reserves', 'industrialized nations increased their reserves (by expanding their money supplies) in amounts far greater than before', 'industrialized nations increased their reserves', 'industrialized nations increased their reserves', 'the industrialized nations increased their reserves']
QANet (full) answer	No answer

Performance breakdown by question type: While this analysis started to reveal key components of how our model performed, question words are not always revealing of what kind of question something is. For example, the question in the development set, "In what country is Normandy located?" would be classified as a "what" question when it really functions more like how we would expect a "where" question to function. As demonstrated in figure 3, "What" questions were most prevalent in our data set; however, these questions could frequently take on the function of other questions.

Further, "how" questions could ask something more concrete like "How many men were in Robert's army?" from the development set or for a description more similar to a "why" question like "How has British art survived in Normandy?" from the development set. As a result, taking inspiration from Zhang and Nunamaker [7], who built a pattern-based question type classifier, we took a heuristic approach to categorizing questions into more informative

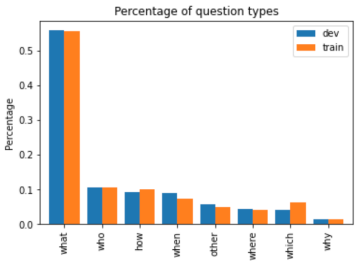

Figure 3: Question word distribution in train and dev set

categories like: "person", "date", "thing", "location", "quantity", and "description". We also had a category "other" for questions that didn't fit any of these categories.

In these categories, we move away from an analysis based on what words exist in the question to an analysis based on what kind of response the question is expecting. This is a useful way to categorize questions because it touches on a linguistic concept called the adjacency pair [8], or a pair of two utterances said by different participants in different turns that are relatively ordered such that the second turn in the pair is responsive to the first. A question-answer pair is an adjacency pair because the question must precede the answer and the answer must respond to that question. Implicit in this idea of the adjacency pair is that in addition to utterances, turns are enacting and accomplishing actions in the real world. These actions limit the next action or utterance, in this case an answer, to a certain number of responses. That is, a question limits the range of next answers, and can do so in different ways. Thus, categorizing a question based on its expectation for what kind of response it will receive is a fitting way to see how our model performs on questions.

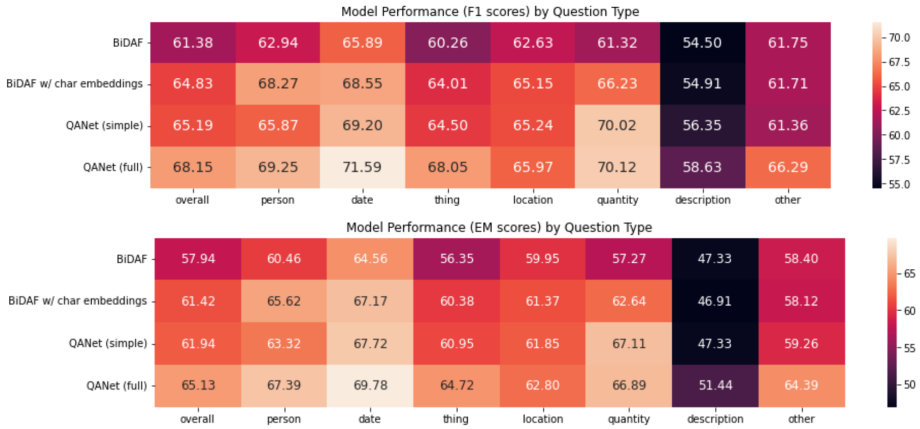

Figure 4: Performance of our models broken down by question types

While we were able to take inspiration from Zhang and Nunamaker [7] in developing our heuristic approach to categorizing question types, they did not actually publish the full extent of their rules. As a result, we developed our own approach which used the word following the question word, part of speech tagging, and named entity recognition to categorize the question type. The specifics of this can be found in Appendix A.3.

We similarly found that description questions are hardest to answer, and predict this might be because description questions require more contextual understanding and don't have a typical answer format. Further, we can see how doing a more complex analysis beyond just question words led to a deeper understanding of what kind of task was difficult for our model. Indeed, in our original categorization, we saw that our improved QANet model got an F1 score of 65.95 for "how" questions. In our new categorization, which broke "how" questions down into "quantity" questions or "description" questions, that "quantity" questions get an F1 score of 70.12, while "description" questions get an F1 score of 58.63. By reanalyzing our

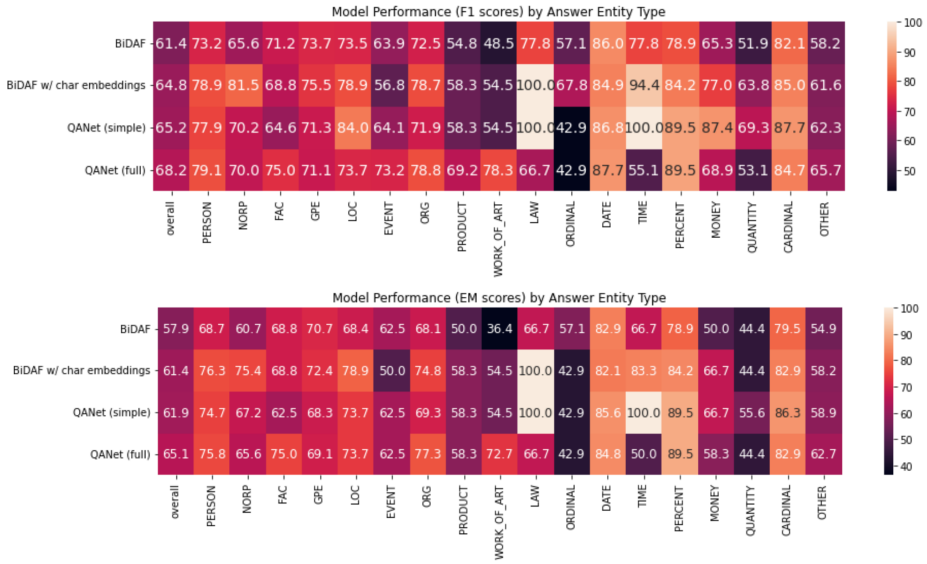

Figure 5: Performance of our models broken down by answer entity types

model from more of a linguistic lens, we were able to provide more quantitative support for our qualitative observation in Example 2 that it was the less distinctive answer format part of the task that was giving our model trouble on "why" question.

Performance breakdown by answer category: Because our exploration of questions by what kind of response they expected was so informative, we also decided to do an analysis of questions categorized by their answers. Because our question type approach already got at some of the answer types, we specifically wanted to target looking at more subcategories of name entities, which already have many built in tools for their categorization. We thus were able to categorize questions by answer entity type: "PERSON", "NORP", "FAC", "GPE", "LOC", "EVENT", "ORG", "PRODUCT", "WORK OF ART", "LAW", "ORDINAL", "DATE", "TIME", "PERCENT", "MONEY", "QUANTITY", and "CARDINAL". See Appendix A.4 for details about each of these codes. We also had a category "OTHER" for questions that didn't fit any of these categories. We find that most named entity categories performed better than non-named entities (categorized as "OTHER"). This may be because these named entities are more concrete or distinctive and can also take very distinctive formats.

6 Conclusion

Overall, we found that using QANet with layer dropout, an added ReLU function in each encoder block, and 7 encoder blocks in the model encoder stack led to the best performance on SQuAD 2.0. While our model far exceeded the performance of the original BiDAF baseline model, we were not able to fully match the original QANet authors' model performance, which could be attributed to the existence of non-answerable questions in this data set or to how we designed the encoder block.

Furthermore, we found that our models performed worst on questions that asked for description and better on questions that asked about date or person. Based on our qualitative analysis, this was likely due to the fact that these descriptions didn't have a specific expected answer format and didn't lend themselves to clear matches in the passages given.

For future work, researchers can explore altering the QANet architecture to see if performance improves. Some possibilities include tweaking the number of blocks in both the embedding and model encoder block stacks. Also, opportunities exist to try and integrate ideas from Transformer XL into QANet to allow the model to learn longer-term dependencies [9]. Lastly, new model and transformer architectures can be explored to further increase question-answering system capabilities.

References

[1] Ali Farhadi Min Joon Seo, Aniruddha Kembhavi and Hannaneh Hajishirzi. Bidirectional Attention Flow for Machine Comprehension. 2016.

[2] Minh-Thang Luong Rui Zhao Kai Chen Mohammad Norouzi Quoc V. Le Adams Wei Yu, David Dohan. Qanet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. 2018.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. 2017.

[4] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[5] Yoon Kim. Convolutional Neural Networks for Sentence Classification. 2014.

[6] CS224N Course Staff. CS224N Default Final Project: Building a QA system (IID SQuAD track). 2022.

[7] Dongsong Zhang, Jay F. Nunamaker. A Natural language approach to content-based video indexing and retrieval for interactive e-learning. *IEEE Transactions on Multimedia*, 6:479–488, 2000.

[8] Emmanuel A. Schegloff. *Sequence Organization in Interaction: A Primer in Conversation Analysis.* Cambridge University Press: Cambridge, UK, 2007.

[9] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. 2019.

A Appendix

A.1 Code

Code for this project is organized in the following GitHub repository. The repository will be made public after the course ends in guidelines with course handout instructions.

A.2 Loss, F1, and EM scores during training for all models

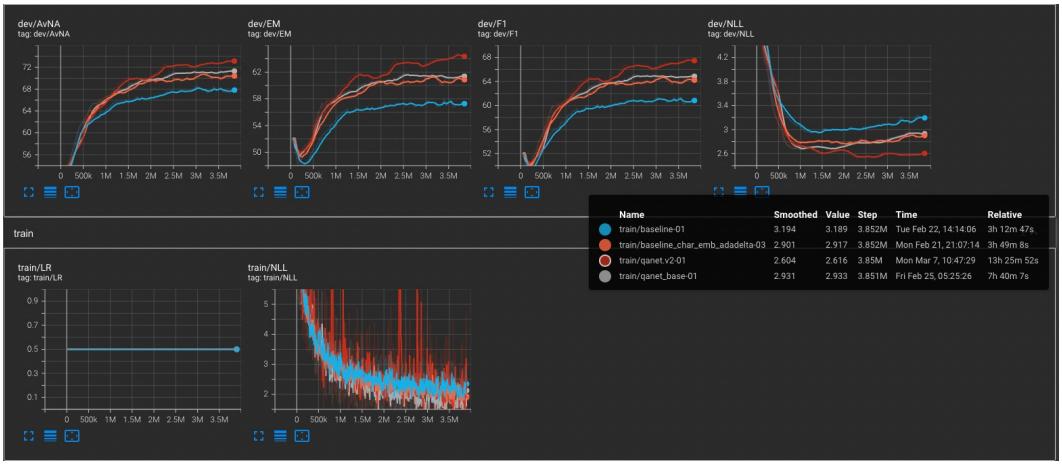

Figure 6: The evolution of the loss function and F1 & EM scores throughout training for all models

A.3 Recategorizing question words as question types

We used the following heuristic to recategorize question words as question types:

First, we recoded "who" as "person", "when" as "date", "where" as "location", "why" as "description", "other" (that is, questions that did not contain "who", "when", "where", "what", "why", "which", "how") remained "other"

We then first attempted to recode "what" and "which" questions by the word following them. If the next word was "city", "state", "place", "country", "countries", "cities", "states", "places", "areas", "area", "region", "regions", "buildings", "building", "county", "counties", we recoded the question as "location". If the next word was "date", "year", "month", "day", "dates", "years", "months", "days", "hour", "hours", "decade", "decades", "millennium", "millennia", "time", "times", "era", "eras", we recoded the question as "date". These words were chosen based on their relationship to place and time. We also attempted to recode "how" questions by the word following them. If the next word was "much", "many", "long", "far", "heavy", "light", "short", we recorded the question as "quantity". This was chosen based on common metrics of quantity. Then we tagged the next word after "how" for part of speech using NLTK's part of speech tagger. If it was a verb, we categorized the question as a "description" because a verb could not be a measure of quantity. Otherwise, we categorized it as "quantity".

Finally, with questions that could not be categorized by previous means, we used spacy's name entity recognition tool on the context for a given question and then searched for the provided answers amongst tagged entities. The tagging was then recoded to a question type category based on the following:

"PERSON":"person", "NORP": "person", "FAC": "location", "GPE": "location", "LOC": "location", "EVENT": "location", "ORG": "thing", "PRODUCT": "thing", "WORK OF ART": "thing", "LAW": "thing", "LANGUAGE": "thing", "ORDINAL": "thing", "DATE": "date", "TIME": "date", "PERCENT": "quantity", "MONEY": "quantity", "QUANTITY": "quantity", "CARDINAL": "quantity"

Otherwise, it was just categorized as "thing".

A.4 Named Entity Recognition Category Descriptions

The following descriptions were taken from the output of the spacy.explain() function, which explains each of the named entity labels it uses:

PERSON: People, including fictional.

NORP: Nationalities or religious or political groups.

FAC: Buildings, airports, highways, bridges, etc.

ORG: Companies, agencies, institutions, etc.

GPE: Countries, cities, states.

LOC: Non-GPE locations, mountain ranges, bodies of water.

PRODUCT: Objects, vehicles, foods, etc. (Not services.)

EVENT: Named hurricanes, battles, wars, sports events, etc.

WORK OF ART: Titles of books, songs, etc.

LAW: Named documents made into laws.

LANGUAGE: Any named language.

DATE: Absolute or relative dates or periods.

TIME: Times smaller than a day.

PERCENT: Percentage, including ”%“.

MONEY: Monetary values, including unit.

QUANTITY: Measurements, as of weight or distance.

ORDINAL: “first”, “second”, etc.

CARDINAL: Numerals that do not fall under another type