Team Xuber: Question Answering via Modified R-NET Construction

Stanford CS224N Default Project
Track: SQuAD

Damon Zuber and Eric Xu
Department of Computer Science
Stanford University
dzuber@stanford.edu and ericxu24@stanford.edu

Abstract

Following the release of the Stanford Question Answering Dataset (SQuAD), rapid innovations have been made in the realm of question answering. The dataset, developed through crowdsourcing, provides a passage and an associated question. The purpose of the dataset is to train deep learning models that can comprehend the provided passage and answer the associated question solely using information found within the passage. To achieve better results than the baseline model, we will augment the algorithm with character-level embeddings to help handle out-of-vocabulary words. More importantly, character-level embeddings are key building block for implementing a self-attention mechanism, which is going to be the main part of our final model. Additionally, we utilize a use a gated recurrent unit (GRU) instead of LSTM to help solve vanishing gradient problems. Although we have yet to fully implement the self-attention mechanism, our intermediate model is already outperforming the baseline, which shows promise for the future. In terms of long-horizon tweaks post self-attention implementation, we are also planning to experiment with hyperparameter tuning, data augmentation, end prediction conditioning, as well as further exploring current research centered around these mechanisms.

1 Key Information to include

- TA mentor: Kathy Yu
- External collaborators (if no, indicate “No”): No
- External mentor (if no, indicate “No”): No
- Sharing project (if no, indicate “No”): No

2 Introduction

The introduction explains the problem, why it’s difficult, interesting, or important, how and why current methods succeed/fail at the problem, and explains the key ideas of your approach and results. Though an introduction covers similar material as an abstract, the introduction gives more space for motivation, detail, references to existing work, and to capture the reader’s interest.

In the recent decades, previously foreign phrases such as "artificial intelligence," "machine learning," and "neural networks" have become household buzzwords. This rise in power and implementation can be seen permeating throughout all aspects of daily life, even something so characteristically "human" as language itself. Whether it’s through increasingly convincing voice assistants such as Siri or changing the landscape of customer service with automated chatbots, the uses are truly endless. However, the limited sizes of currently available datasets limits the complexity of models

that researchers are able to analyze. The recently released Stanford Question Answering dataset (SQuAD) addresses the weakness of the previous datasets. It is orders of magnitude larger than all previous hand-annotated datasets, and is challenging: all the answers can be arbitrary spans within context paragraphs rather than a limited set of multiple choices.

Question: Why was Tesla returned to Gospic?
Context paragraph: On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit. On 17 April 1879, Milutin Tesla died at the age of 60 after contracting an unspecified illness (although some sources say that he died of a stroke). During that year, Tesla taught a large class of students in his old school, Higher Real Gymnasium, in Gospic.
Answer: not having a residence permit

Figure 1: *Example of a question-answering task; the blue is where the model predicted the answer to lie based on the question and context paragraph.*

For reading comprehension style question answering, a passage and related question are given, the task is then to predict an answer to the prompted question based on information found in in the context paragraph. The SQuAD dataset further constrains the "correct" answer to be a continuous sub-span of the context passage. The sub-span answer often includes non-entities and can be much longer phrases. This setup challenges the researcher to understand and reason about not only the question but also the context passage in order to infer the answer

Although it may seem like an arbitrary online challenge, this question-answering problem (e.x. Figure 1) that SQuAD presents has important implications. The end goal, like many other AI/ML projects, is to create a model that not only replaces the need for a human, but eventually outperforms–whether in speed, accuracy, or even both–what an human is capable of. The applications of such a model are endless: education, service, research, etc.

In out project, we explore several neural network architectures for the SQuAD task. We begin with the Stanford-provided baseline model that has basic encoding, attention and decoding layer, upon which we made incremental tweaks based on analyzing what the baseline model did and did not do well. Specifically, we decided to augment the baseline model with character-level embeddings and tweaking the attention layer to give the model more specificity within the context paragraph.

3 Related Work

For the majority of this project, we were primarily inspired by a 2017 Microsoft Research Asia paper [2]. Their approach to the question-answering problem was a propriety model dubbed "R-NET"; the structural schematic is illustrate in Figure 2 below. Their custom model consisted of four main parts: "1) the recurrent network encoder to build representation for questions and passages separately, 2) the gated matching layer to match the question and passage, 3) the self-matching layer to aggregate information from the whole passage, and 4) the pointer-network based answer boundary prediction layer" [2].

The current scores of the 2017 R-NET paper achieved scores of (72.3 EM, 80.7 F1) and (76.9 EM, 84.0 F1) for the single and ensemble model respectively [2].

At an intuitive level, R-NET performs reading comprehension in a way that is pretty similar to how an average human would: by ‘reading’ (applying a Recurrent Neural Network) the text multiple times, and ‘fine-tuning’ (using Attention) the vectorial representations of the terms better and better in each iteration.

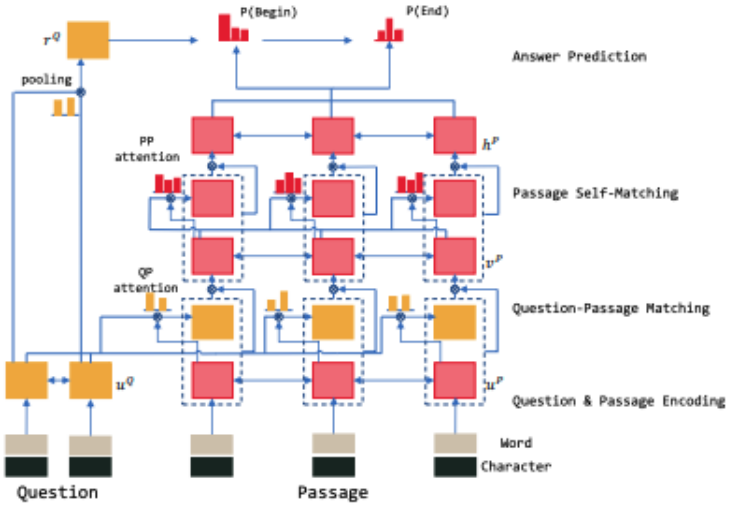

Figure 2: *Structural overview of the R-NET model constructed by the Microsoft Research Asian team.*

Additionally, within our model, we also implemented a character-level embedding layer. Our main inspiration for this augmentation came from Seo's University of Washington paper [3]. In this paper, the model greatly resembles the baseline model provided, however a character-level embedding layer (shown in pink in Figure 3 below) was added before the word-level embedding layer.

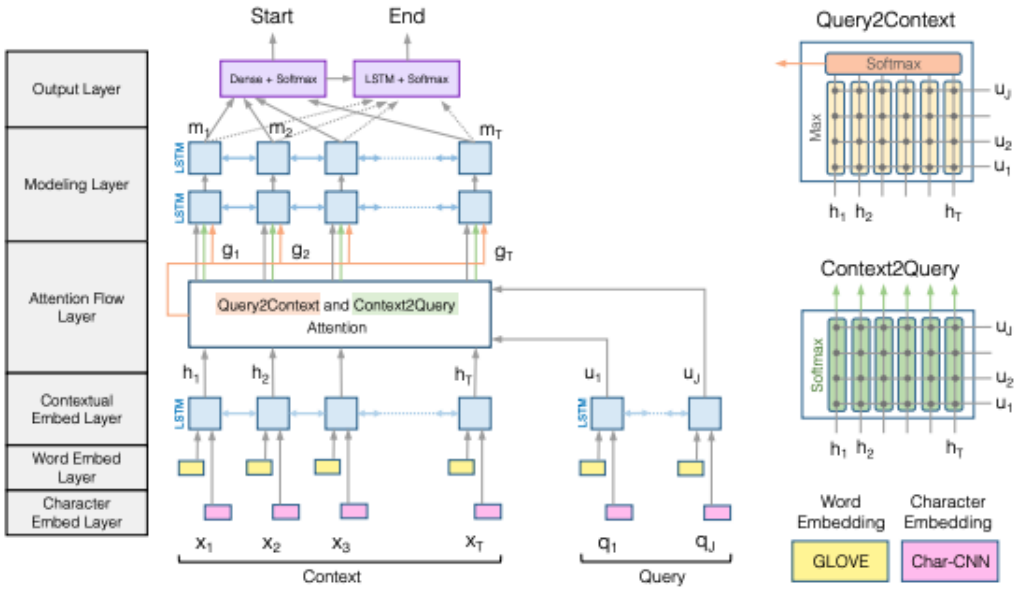

Figure 3: *Flow Model used by Seo's University of Washington team. The added character-level embedding layer is shown in pink.*

This character embedding layer maps each word to a high-dimensional vector space. Characters are embedded into vectors, which can be considered as 1D inputs to a Convolutional Neural Networks (CNN), and whose size is the input channel size of the CNN. The outputs of the CNN are max-pooled over the entire width to obtain a fixed-size vector for each word. In plain English, character-level embeddings find numeric representation of words by looking at their character-level compositions. Thus, character-level embeddings provide several advantages: ability to handle out-of-vocabulary

words, ability to handle infrequently used words and reducing vector sizes, which reduces model complexity and improves speed.

4 Approach

For this project, our main source of comparison of performance is the provided baseline model based on Bidirectional Attention Flow (BiDAF) [1]. However, this model doesn't does not include a character-level embedding layer. thus, as a first step, we implemented such a layer to help handle out-of-vocabulary words. More importantly, character-level embeddings are key building block for implementing self-attention, which is going to be the main part of our final model. Specifically, we will be emulating our central Microsoft paper's R-NET construction, which utilizes a self-matching attention mechanism [2].

Given time and expertise restraints, we decided to focus on what we thought was the most impactful improvement made by the Microsoft Research Asia team, which was the self-matching attention layer. Attention in Neural Networks is modeled after the way humans focus on a particular subset of their sensory input, and tune-out the rest. "Self-matching" in this case means that given a text length n each hidden state h_t attends to all the hidden states, $h_1, \ldots, h_n$, including itself. In essence, by directly matching the question-aware passage representation against itself, we can incorporate the passage context necessary to infer the answer of a potential query. The main idea of self-attention is that the query vector is from the same set as the set of value vectors. Thus, by introducing a self-matching attention layer, 3.

Specifically, we tried to emulate what the Microsoft Research Asian team did, which was dynamically collect evidence from the whole passage for words in passage and encodes the evidence relevant to the current passage word and its matching question information into the passage representation h_t^P:

$$h_t^P = BiRNN\left(h_{t-1}^P, \left[v_t^P, c_t\right]\right) \tag{1}$$

where $c_t = att\left(v^P, v_t^P\right)$ is an attention-pooling vector of the whole passage $\left(v^P\right)$:

$$s_j^t = \mathrm{v}^{\mathrm{T}} \tanh\left(W_v^P v_j^P + W_v^{\tilde{P}} v_t^P\right) \tag{2}$$

$$a_i^t = \exp\left(s_i^t\right) / \Sigma_{j=1}^n \exp\left(s_j^t\right) \tag{3}$$

$$c_t = \Sigma_{i=1}^n a_i^t v_i^P \tag{4}$$

Additionally, we used a gated recurrent unit (GRU) instead of long short-term memory (LSTM) to solve the vanishing gradient problem faced by standard recurrent neural networks (RNN). Although GRUs share many properties of LSTM-both algorithms use a gating mechanism to control the memorization process-GRU are typically less complex and significantly faster to compute [4]. GRU uses two gates: the update gate and the reset gate whereas LSTM uses three: the input gate, the forget gate, and the output gate. Unlike LSTM, GRU does not have an output gate and combines the input and the forget gate into a single update gate. To motivate this change, we reference several papers that have already compared and proven the similar relative performance of GRU and LSTM [5].

Finally, we also changed the batch size from 64 to 32, as increased batch size can decrease generalization ability of model. Thus, a slightly smaller batch size would not exploit vectorization to as full of an extent, but may perform better.

5 Experiments

5.1 Data

We will be using the official SQuAD 2.0 dataset, which contains the following splits:

- train (129,941 examples): All taken from the official SQuAD 2.0 training set.
- dev (6078 examples): Roughly half of the official dev set, randomly selected.

- test (5915 examples): The remaining examples from the official dev set, plus hand-labeled examples

Like the project handout states, The SQuAD dataset contains many (*context, question, answer*) triples. Each *context* (sometimes called a passage, paragraph or document in other papers) is an excerpt from Wikipedia. The *question* (sometimes called a query in other papers) is the question to be answered based on the context. The *answer* is a span (i.e. excerpt of text) from the context.

5.2 Evaluation method

Like the default model, we will utilize two metrics to evaluate model performance of SQuAD: Exact Match (EM) and F1 score. EM measures the percentage of the prediction that matches one of the ground truth answers exactly. F1 measures the overlap between the prediction and ground truth answers which takes the maximum F1 over all of the ground truth answers.

5.3 Experimental details

The main tweak we made to the experimental process was changing the batch size from 64 to 32, as increased batch size can decrease generalization ability of model. Thus, a slightly smaller batch size would not exploit vectorization to as full of an extent, but may perform better.

With each incremental edit, we also played around with slightly varying learning rates but these changes ultimately yielded negligible results in performance.

5.4 Results

For our baseline model, we achieved scores of:

$$EM : 57.08 \tag{5}$$

$$F1 : 60.42 \tag{6}$$

During our model-curation process, we managed to achieve the following *dev* scores:

$$EM : 59.14 \tag{7}$$

$$F1 : 62.51 \tag{8}$$

We also had the following scores on test:

$$EM : 59.17 \tag{9}$$

$$F1 : 62.48 \tag{10}$$

Our scores are about as expected. Given that we only implemented a sub-portion of the broader R-NET model, it makes sense that our scores are still significantly lower than their final scores. However, our augmentations to the model showed promise as the scores did improve from the baseline, as expected (and as hoped).

6 Analysis

After testing our model, we realized that many of the incorrect answers seemed to follow three main patterns.

First, we noticed that our model is overemphasizing the proximity of the keywords in the question. Although the model answer contains an albeit rephrased correct answer, which is a product of the self-attention layer learning the entire context of the paragraph, it also contains irrelevant text. We might find this to be the case due to the self-attention layer itself, which may have falsely given weights to certain contexts within the passage.

Second, consider the following example:

Context: "Politically, the system of government created by Kublai Khan was the product of a compromise between Mongolian patrimonial feudalism and the traditional Chinese autocratic-bureaucratic system. Nevertheless, socially the educated Chinese elite were in general not given the degree of esteem that they had been accorded previously under native Chinese dynasties. Although the traditional Chinese elite were not given their share of power, the Mongols and the Semuren (various allied groups from Central Asia and the western end of the empire) largely remained strangers to the mainstream Chinese culture, and this dichotomy gave the Yuan regime a somewhat strong colonial coloration. The unequal treatment is possibly due to the fear of transferring power to the ethnic Chinese under their rule. The Mongols and Semuren were given certain advantages in the dynasty, and this would last even after the restoration of the imperial examination in the early 14th century. In general there were very few North Chinese or Southerners reaching the highest-post in the government compared with the possibility that Persians did so in the Ilkhanate. Later the Yongle Emperor of the Ming dynasty also mentioned the discrimination that existed during the Yuan dynasty. In response to an objection against the use of barbarians in his government, the Yongle Emperor answered: ... Discrimination was used by the Mongols during the Yuan dynasty, who employed only Mongols and Tartars and discarded northern and southern Chinese and this was precisely the cause that brought disaster upon them"

Question: How did the unequal treatment of Chinese versus Mongols in the Yuan make the dynasty seem?

Correct Answer: colonial

Model's Answer: fear of transferring power to the ethnic Chinese under their rule

Our above results may suggest an over-fitting to the data for our model, which means that we could benefit from additional drop-off and less training epochs. Our answer showed too much adherence to the wording of the question itself, and also resembled some answers regarding Mongols from the training set.

Lastly, we found that our model said that certain answerable questions could not be answered, while several questions that could not be answered received answers. While we cannot pinpoint the exact part of our model that would cause this issue, this could be a result of our start and end probabilities not reaching a certain threshold in some areas, skewing our model's predictive capabilities.

7 Conclusion

We sought to implement a variation of R-Net with code that initially served as a BIDAF model. In order to shift towards R-Net, we implemented character-level embeddings to build embeddings which are more resistant to character-level deviations. Furthermore, in the case of out of context words, character embeddings improve our classifications, and thus preventing cases where word-level embeddings assign random values and skew the model. We honed in on the implementation character-level embeddings as a critical feature of our model. We then concatenated our character-level and word-level embeddings, and then sought out to create a self-matching attention structure similar in nature to Microsoft's R-Net paper.

Our results after the implementation of our self-matching attention were less strong than when we used the BIDAF attention layer, so we know that there was room for improvement for our functionality there. However, the process of coding this function showed just how powerful variations of additive attention can be, and if we were to have nailed down the self-matching attention, we could have seen vastly better results. We can say with confidence that adding character-level embeddings and lowering batch size could improve upon a BIDAF model, but we did not find that our implementation of self-attention could beat out the attention layer from the starter code.

We also found that replacing an LSTM cell type with GRU improved performance. Over the course of the project, we have learned a great deal about how certain training parameters, cell types, and embedding types can affect training time and improve performance. Additionally, in training a large

data set, we put into action the principles of scaling and performance effects, as discussed by guest lecturer Jared Kaplan. Furthermore, we were exposed to the critical nature of having access to increased computing power.

We were proud to have made additions to the starter code such that we could beat the baseline scores, and feel as if more time could enable us to reach higher heights in question answering. With more time, we think it would also be interesting to continue analyzing Microsoft's R-NET model structure incrementally. By rigorously implementing and testing each step, we could not only gain a deeper understanding about their motivations behind each modification, but maybe find potential spots or tweaks that they overlooked.

Unsurprisingly, since we were only able to implement a model that was less intricate than their R-NET, we weren't able to achieve the same level of results, but the end goal would be to build on top of their already completed work to continue improving the evaluation scores.

8 References

[1] CS 224N Default Final Project: Building a QA System (Iid ... Stanford University Department of Computer Science, *https://web.stanford.edu/class/cs224n/project/default-final-project-handout-squad-track.pdf.*

[2] Microsoft Research Asia, Natural Language Computing Group. "R-NET: MACHINE READING COMPREHENSION WITH SELF-MATCHING NETWORKS." Natural Language Computing Group.

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

[4] Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing* (EMNLP 2014)

[5] Chung, J., Gulcehre, C., Cho, K., Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*, December 2014

[6] Kaplan, J., McClandish, S., Henighan, T. "Scaling Laws for Neural Language Models." Computer Science. January 2020