CS224N Winter 2019
Default Project Proposal

Matt Linker, Zhilin Jiang, Zheng Nie
{mslinker, zjiang23, zhengnie}@stanford.edu

1 Introduction

Question-answering (QA) has been an important task and popular research topic of natural language processing (NLP). There are various existing QA datasets available, among which the Stanford Question Answering Dataset (SQuAD) is an state-of-the-art reading comprehension dataset for training and evaluating QA systems [Rajpurkar et al., 2018].

In this project, we will explore building and improving a system for the QA tasks defined by SQuAD 2.0, in order to contribute to this popular research topic as well as gaining practical experiences and better understanding of applying neural models to NLP tasks in general.

2 Related Work

In preparing for our project, we chose to dive into the work of Chen et al. [2017] on DrQA, a robust question-answering (QA) system building upon a vanilla RNN. In essence, DrQA seeks to mitigate the problem of determining the existence of, locating, and identifying relevant passages of text, which can then be fed into a more traditional RNN-based QA system. Such an idea was inspired by the immense breadth of Wikipedia – it would be far too computationally-intense to run an RNN over the whole of the Wikipedia corpus for a single query – some efficient means is needed to find the most relevant documents or paragraphs such that attention (figuratively, not neural-network attention) can be focused there by the actual RNN.

The implementation of this system is fairly straightforward. Essentially, it utilized an inverted index (an index that maps words or other similar features to documents/passages in which they are found) with TF-IDF weighting in order to identify and extract 5 Wikipedia pages determined to be the most likely to contain the relevant information. TF-IDF weighting is a deterministic scoring system for relevance of a given search query – essentially, it computes a weighted similarity score between queries and documents, with higher weightings given to less common words found in both (e.g. amygdala), and lower weighting to more common words (e.g. the). From there, the authors compared their results

to those of actual Wikipedia search, and found that they were able to obtain more relevant results in comparison – indicative of a reasonably-good localizing system for finding relevant documents.

From there, the system largely looks like that given in the baseline model, with the exception that the top 5 candidate documents/passages are fed in to the RNN, instead of just a single passage as in the baseline. The model then sees if a suitable answer is found, and if so, returns that answer. Performance was strong compared to other non-BERT models, with an F1 score of 79% on version 1.0 of the Stanford Question Answering Dataset (SQuAD) [Rajpurkar et al., 2016].

Overall, DrQA is a strong model with many ideas worth trying. In particular, the usage of TF-IDF weighting at the sentence level within a passage could help us to eliminate irrelevant information, threshold for a more robust model, or even feed separately into our attention calculation (perhaps in conjunction with an attention-based pipeline such as TransformerXL [Dai et al., 2018]).

In preparing to work on our own model, we also read and investigated the previous work of Dai et al. [2018] on TransformerXL. The idea here is essentially to build upon a standard Transformer-based approach by introducing recurrence into the self-attention network, reusing previous hidden states from previous segments to build up a "memory" over time. Doing so in theory should be substantially faster than "vanilla" Transformer, as we are not starting a new attention calculation "from scratch" each time. Over time, attention length is iteratively increased until no (or very little) incremental improvement is seen. More concretely, the approach is to re-parameterize to remove absolute position information from our data, replacing it with a trainable parameter, and seeing how that updates our attention distribution. Result-wise, Dai et. al found that this approach significantly lowered perplexity compared to previous Transformer-based models and works much faster comparatively, especially with longer attention parameters. This is particularly promising to us, as considering several sentences (or even paragraphs) in context may yield additional information not found in shorter contexts. For us, it could be useful to inspect whether this perplexity reduction aspect has an impact on model performance on our dataset (which is comprised of relatively-short passages).

3 Project Description

3.1 Task and Project Goals

Our task is identical to what has been formally defined by the SQuAD 1.0/1.1 [Rajpurkar et al., 2016] and SQuAD 2.0 [Rajpurkar et al., 2018] datasets. Specifically, SQuAD 1.0/1.1 defines the task where a system takes in text passages and corresponding reading comprehension questions as inputs, and attempts to output text sentences/phrases as answers to the input questions, based on information extracted from the input text passages. SQuAD 2.0 extends the challenge by including 50,000 adversarially constructed unanswerable questions,

to which the system is expected to abstain from producing an output answer.

Our ultimate goal is to design, implement, train, and evaluate a neural model-based system for the question-answering tasks defined in SQuAD 2.0 [Rajpurkar et al., 2018] that attains a high evaluation score on the official SQuAD 2.0 leaderboard (available at `https://rajpurkar.github.io/SQuAD-explorer/`). We plan to reach this goal by incorporating a combination of several existing techniques, as well as experimenting with and evaluating various optimizations along the way. Specifically, we plan to extend our existing baseline model with DrQA and TransformerXL, reach reasonably good results on the default SQuAD 2.0 tasks, and explore incorporating other methods and/or techniques to further improve the evaluation results.

Additionally, once we reach satisfactory performance on SQuAD 2.0 evaluation with the aforementioned methods, we will attempt to improve the model's training and inference speed while minimizing accuracy loss, by attempting to reduce the model complexity as well as replacing computation-heavy components with recent techniques such as QANet [Yu et al., 2018].

3.2 Dataset

We will start with the SQuAD 2.0 training and dev set. As we explore different methods to improve our performance, we might eventually introduce external data as an attempt to extend our models' capabilities. The intuition against using external data from the very start is to first ensure our model performs well under the default training data, without the risk of exposing it to data that potentially under a different distribution. This is further supported by the fact that SQuAD 2.0 dataset contains a sufficient number of samples for our basic training purposes.

3.3 Methods

3.3.1 DrQA

We will start by extending our baseline model with DrQA, the RNN-based question-answering system that we have already introduced in detail in section 2. We will modify our baseline model so that the DrQA implementation, including the TF-IDF weighting, is incorporated properly, and we will evaluate our model after this first stage to observe the change in performance.

3.3.2 Transformer-XL

Although we may focus on DrQA model implementation, however, after we successfully implement the model and achieve the target model accuracy, we may explore in conjunction with an attention-based pipeline Transformer-XL in this case. We may want to try to use the reusable hidden state self attention concept, so that the encoder can get extend dependencies for long sentence to try to get some improvement.

3.4 Baseline

The baseline model we will use is based on Bi-Directional Attention Flow (BiDAF) model without a character-level embedding layer.

Embedding Layer (`layers.Embedding`)

Given some input word indices $w_1, ..., w_k \in \mathbb{N}$, the embedding layer performs an embedding lookup to convert the indices into word embeddings $v_1, ...v_k \in \mathbb{R}$. Perform this for both context and the questsion. Producing embedding $c_1, ...c_N \in \mathbb{R}^D$ for the context and $q_1...q_M \in \mathbb{R}^D$ for the question

Encoder Layer (`layers.RNNEncoder`)

The encoder layer uses a bi-directional LSTM to allow the model to incorporate temporal dependencies between time steps of the embedding layers's output.

Attention Layer (`layers.BiDAFAttention`)

Use Bi-Directional Attention Flow (BiDAF) layer flow from Context-to-Question(C2Q) and from Question-to-Context(Q2C) to get the output by combining hidden state, C2Q attention output,and Q2C attention output for hidden state.

Modeling Layer (`layers.RNNEncoder`)

A two-layer LSTM comes after the attention layer to refine the sequence of vectors after the attention layer to integrates temporal information between context representations conditioned on the question.

Output Layer (`layers.BiDAFOutput`)

Final layer of the model is to produce a vector of probabilities corresponding to each position in the context for answer begin location p_{start} and answer ending location p_{end}. Output layer uses attention layer outputs and modeling layer outputs applies a bi-directional LSTM to get p_{start} and p_{end}

3.5 Evaluation

We will use F1 score and Exactly Match score (EM) to evaluate the implemented model.

To calculate F1 score, we need to compute precision(P) and recall(R).

$$P = \frac{true\ positives}{true\ positives + false\ positives} \tag{1}$$

$$R = \frac{true\ positives}{true\ positives + false\ negatives} \tag{2}$$

$$F1 = 2\frac{PR}{P+R} \tag{3}$$

To calculate exactly match score (EM score), we need compute the percentage of exactly matched gold and predicted answers verses the total evaluated questions. Then multiply by 100.

$$EM = \frac{exactly\ match\ answers}{total\ evaluated\ questions} \times 100 \tag{4}$$

F1 and EM score generated by implemented model in this project should above the baseline model F1 and EM score.

References

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

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-xl: Language modeling with longer-term dependency. 2018.

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

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

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.