Partial Re-Implementation of R-Net for SQuAD

Stanford CS224N Default Project

Danny Tse
Department of Computer Science
Stanford University
dannytse@stanford.edu
Mentor: Yuyan Wang

Abstract

An abstract should concisely (less than 300 words) motivate the problem, describe your aims, describe your contribution, and highlight your main finding(s).

1 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.

2 Related Work

This section helps the reader understand the research context of your work, by providing an overview of existing work in the area.

3 Approach

This section details your approach(es) to the problem. For example, this is where you describe the architecture of your neural network(s), and any other key methods or algorithms.

We wish to develop a model that performs relatively well on the Stanford Question Answering Dataset (SQuAD). To do this, we reimplement the complex neural model described in the paper *R-Net: Machine Reading Comprehension with Self-Matching Networks* [1], including its contributions such as the gated attention-based recurrent network and self-matching mechanism. Furthermore, we wish to determine how different ways of processing character embeddings interact with the rest of the model architecture, and how the self-attention mechanisms described in the paper compare to the given baseline RNN model for question-answering. We suspect that, since the self-attention mechanism allows hidden states to consider previous hidden states, this model can record long-distance dependencies, and as a result have more complete answers to questions. The project is a direct reimplementation of the paper, even training on a similar dataset (SQuAD 2.0), so reimplementing the model, tuning hyper-parameters, and making few adjustments should suffice. Variants of self-attention have proven to be vastly successful in modern natural language processing models (ex. usage of multi-headed self-attention in a Transformer Encoder-Decoder block introduced in Vaswani et al. (2017)'s *Attention Is All You Need* [2]), thus in theory, this model should outperform our Bidirectional Attention Flow baseline.

We describe the model architecture below. First, we process the question and passage through a bi-directional recurrent network. Then, we match each question and passage with gated attention-based recurrent networks, followed by self-matching attention. This provides us with a question-aware representation for the passage. Then, we refine the passage representation with self-matching attention and feed it into the output layer to return the boundary of the answer span. The model layers are:

1. **Question and Passage Encoder** Passing in a question $Q = \{w_t^Q\}_{t=1}^m$ and a passage $P = \{w_t^P\}_{t=1}^n$, we begin by converting the question and passage into their corresponding word embeddings ($\{e_t^Q\}_{t=1}^m$ and $\{e_t^P\}_{t=1}^n$) respectively). We obtain the character-level embeddings by taking the final hidden states of a bi-directional recurrent neural network (RNN) applied to the embeddings of characters in the token. Then, we apply a Gated Recurrent Unit (GRU) to obtain representations for words in the question and the passage. That is,

$$u_t^Q = \text{GRU}_Q(u_{t-1}^Q, [e_t^Q, c_t^Q])$$
$$u_t^P = \text{GRU}_P(u_{t-1}^P, [e_t^P, c_t^P])$$

2. **Gated Attention-Based Recurrent Networks** Now, given question and passage representations u_t^Q and u_t^P from the Question and Passage Encoder, we use a gated attention-based recurrent network to match question information with passage representation. Wang & Jiang (2016) [3] propose match-LSTM:

$$v_t^P = \text{GRU}(v_{t-1}^P, [u_t^P, c_t])$$

where c_t is given by the attention-pooling vector of the entire question u^Q:

$$s_j^t = v^T \text{tanh}(W_u^Q u_j^Q + W_u^P u_t^P + W_v^P v_{t-1}^P)$$
$$a_i^t = \text{exp}(s_i^t) / \sum_{j=1}^m \text{exp}(s_j^t)$$
$$c_t = \sum_{i=1}^m a_i^t u_i^Q$$

Then, we add another gate to the input to the input $[u_t^P, c_t]$:

$$g_t = \text{sigmoid}(W_g[u_t^P, c_t])$$
$$[u_t^P, c_t]^* = g_t \odot [u_t^P, c_t],$$

which is establishes a relationship between the current passage word and its attention-pooling vector of the question. $[u_t^P, c_t]^*$ is utilized in subsequent calculations in place of $[u_t^P, c_t]$, and is called the "gated attention-based recurrent network".

3. **Self-Matching Attention** Now, we are given the question-aware passage representation from previous layers, or $\{v_t^P\}_{t=1}^n$. An issue with such a representation is that it lacks extensive knowledge of context. To circumvent this issue, the paper proposes using attention to match $\{v_t^P\}_{t=1}^n$ against itself, obtaining h_t^P which encodes evidence from words in the passage and their matching question information:

$$h_t^P = \text{GRU}(h_{t-1}^P, [v_t^P, c_t])$$

where c_t is given by the attention-pooling vector of the whole passage v^P:

$$s_j^t = v^T \text{tanh}(W_v^P u_j^P + W_u^{\bar{P}} u_t^P)$$
$$a_i^t = \text{exp}(s_i^t) / \sum_{j=1}^n \text{exp}(s_j^t)$$
$$c_t = \sum_{i=1}^n a_i^t v_i^P$$

Once again, an additional gate as stated above is applied to $[v_t^P, c_t]$.

4. **Output Layer** Finally, pointer networks are used to predict the start and end position of the answer. Attention-pooling over the question representation is used to generate the initial hidden vector for the pointer network. Given $\{h_t^P\}_{t=1}^n$ from the previous layers, we utilize the attention mechanism to select start and end positions p^1 and p^2 from the passage:

$$s_j^t = v^T \text{tanh}(W_h^P h_j^P + W_h^a h_{t-1}^a)$$
$$a_i^t = \text{exp}(s_i^t) / \sum_{j=1}^n \text{exp}(s_j^t)$$
$$p_t = \text{argmax}(a_1^t, \ldots, a_n^t)$$

where

$$c_t = \sum_{i=1}^n a_i^t h_i^P$$
$$h_t^a = \text{GRU}(h_{t-1}^a, c_t)$$

Then, using h_a^{t-1} as the initial hidden state and r^Q as the initial state,

$$s_j = v^T \tanh(W_u^Q u_j^Q + W_v^Q V_r^Q)$$
$$a_i = \exp(s_i) / \sum_{j=1}^{n} \exp(s_j)$$
$$r^Q = \sum_{i=1}^{m} a_i u_i^Q$$

Though the paper describes exactly what layers to use, and what model architecture components to incorporate, we may experiment with other adjustments that have proven to work. One example of such an adjustment is described in Seo et al. (2017)'s *Bi-Directional Attention Flow for Machine Comprehension* [4], where a Convolutional Neural Network (CNN) Layer is utilized to process character embeddings. In this modification, we simply obtain the character-level embedding of each word using a CNN, instead of a GRU. Characters are embedded into vectors whose size is the input-channel size of the CNN, which are then fed as 1-dimensional inputs. Then, we apply a max-pool on the CNN's outputs over the entire width to obtain a fixed-size vector for each word.

The baseline we use is a Bi-Directional Attention Flow (BiDAF) model on the word embeddings. This model is described in the Default Final Project Handout (SQuAD Track), and consists of a Word Embedding Layer, an Encoder Layer, an Attention Layer, a Modeling Layer, and an Output Layer.

4 Experiments

This section contains the following.

4.1 Data

As data, we use the Stanford Question Answering Dataset 2.0 (SQuAD 2.0), which has the splits train (129,941 examples), dev (6078 examples), and split (5915 examples). In this dataset, passages are selected from the English Wikipedia (usually 100-150 words), questions are crowd-sourced, and each answer is either contained within the passage, or does not exist at all (not answerable). Also, each answerable SQuAD question has three answers, each from a different crowd worker.

Describe the dataset(s) you are using (provide references). If it's not already clear, make sure the associated task is clearly described. Being precise about the exact form of the input and output can be very useful for readers attempting to understand your work, especially if you've defined your own task.

4.2 Evaluation method

Describe the evaluation metric(s) you use, plus any other details necessary to understand your evaluation. Some projects will have clear metrics from prior work on given datasets, but we realize that other projects will define their own metrics. If you're defining your own metrics, be clear as to what you're hoping to measure with each evaluation method (whether quantitative or qualitative, automatic or human-defined!), and how it's defined.

4.3 Experimental details

Report how you ran your experiments (e.g. model configurations, learning rate, training time, etc.)

4.4 Results

Report the quantitative results that you have found so far. Use a table or plot to compare results and compare against baselines.

- If you're a default project team, you should **report the F1 and EM scores you obtained on the test leaderboard** in this section. Make it clear whether you are on the non-PCE or PCE leaderboard. You can also report dev set results if you like.
- Comment on your quantitative results. Are they what you expected? Better than you expected? Worse than you expected? Why do you think that is? What does that tell you about your approach?

5 Analysis

Your report should include *qualitative evaluation*. That is, try to understand your system (e.g. how it works, when it succeeds and when it fails) by inspecting key characteristics or outputs of your model.

6 Conclusion

Summarize the main findings of your project, and what you have learnt. Highlight your achievements, and note the primary limitations of your work. If you like, you can describe avenues for future work.

References

[1] Microsoft Research Asia Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. 2017.

[2] Niki Parmar Jakob Uszkoreit Llion Jones Aidan N Gomez Lukasz Kaiser Ashish Vaswani, Noam Shazeer and Illia Polosukhin. Attention is all you need.

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

[4] Ali Farhadi Hananneh Hajishirzi Minjoon Seo, Aniruddha Kembhavi. Bi-directional attention flow for machine comprehension. In *arXiv preprint arXiv:1611.01603*.