Machine Comprehension using match-LSTM and Answer-Pointer

Akhil Nair and Ramkishan Panthena

Abstract— **Reading comprehension is the ability to read a piece of text and then answer questions about it. Teaching machines to read, process and comprehend natural language documents is a coveted goal for artificial intelligence. It is also a challenging task requiring understanding of natural language and deep learning. Several benchmark datasets have been created to focus on answering questions to evaluate machine comprehension and the ability of machine to understand text. In this project, we try to explore some NLP and deep learning techniques to solve this problem of Machine Comprehension(MC).**

I. PROBLEM DESCRIPTION

The aim of the project is to implement a neural network architecture for Reading Comprehension using the Stanford Question Answering Dataset(SQuAD)[1]. SQuAD is comprised of around 100K question-answer pairs, along with a context paragraph. The context paragraphs were extracted from a set of articles from Wikipedia. Humans generated questions using that paragraph as a context, and selected a span from the same paragraph as the target answer. The following is an example of a triplet (question, context, answer):

Question: How large in square feet is the LaFortune Center at Notre Dame?

Context paragraph: A Science Hall was built in 1883 under the direction of Fr.Zahm, but in 1950 it was converted to a student union building and named LaFortune Center, after Joseph LaFortune, an oil executive from Tulsa, Oklahoma. Commonly known as ”LaFortune” or ”LaFun,” it is a 4-story building of **83,000 square feet** that provides the Notre Dame community with a meeting place for social, recreational, cultural, and educational activities. LaFortune employs 35 part-time student staff and 29 full-time non-student staff and has an annual budget of $1.2 million. Many businesses, services, and divisions of The Office of Student Affairs are found within. The building also houses restaurants from national restaurant chains.

Answer: 83,000 square feet

Fig. 1. Sample passage in SQuAD dataset

In the SQuAD task, answering a question is dened as predicting an answer span within a given context paragraph.

II. RELATED WORK/MODEL DESCRIPTION

A. Previous work done

- End-to-end neural network models have made significant progress in solving the problem of Machine Comprehension. A common approach is to use recurrent neural networks(RNNs) to process the given text and the question to predict or generate the answers (Hermann 2015)[2]. Question-aware passage representation with match-LSTM, and Pointer Networks to generate answers that contain multiple tokens from the given passage (Wang and Jiang, 2016)[3] have also been used by various models.
- Hermann (2015) first introduced attention mechanism into reading comprehension, which matches the question with the given passage. The co-attention mechanism (Xiong, Zhong, Socher 2017; Seo 2017)[4][5] is also a widely used model which is computed as an alignment matrix corresponding to all pairs of context words and query words, which can model complex interaction between the query and the context.
- Multi-hop reasoning models have also been proposed for MC tasks (Shen 2016; Xiong, Zhong, Socher 2017)[4]. These models typically maintain a memory state which incorporates the current information of reasoning with the previous information in the memory, by following the framework of Memory Networks (Sukhbaatar 2015)[6].

B. Model Description

Our task is to predict an answer span tuple i.e. a sequence of two indices indicating the start position and end position of the answer in the paragraph, given a question and a context (paragraph). Our approach to this task is to implement a Match LSTM with Answer-Pointer model described in Wang et al.,2016 and Jiang et al.,2016.[3] The pointer net (Answer-Pointer) model was adopted into Match LSTM to allow us to generate multiple tokens from the original text rather than a large fixed vocabulary.

What's new in our work?

The original implementation uses a standard one directional LSTM in the preprocessing layer to incorporate contextual information into the representation of each token in passage and the question. In our model architecture we added an additional LSTM in the preprocessing layer to make our preprocessing LSTM bidirectional in nature. The idea was to obtain a representation that encodes the contexts from both directions for each token in the passage and the question.

Our model consists of three layers:

- An LSTM pre-processing layer that pre-processes the passage and the question using LSTM.
  - The initial layer is a bi-directional LSTM over the passage and question separately.

$$H^P_{fw} = \overrightarrow{LSTM}(P), H^P_{bw} = \overleftarrow{LSTM}(P)$$
$$H^Q_{fw} = \overrightarrow{LSTM}(Q), H^Q_{bw} = \overleftarrow{LSTM}(Q)$$

  - As we have Bi-LSTM, we have two outputs, which are not connected. So we merge them to obtain a single output:

$$H^P = concat(H^P_{fw}, H^P_{bw})$$
$$H^Q = concat(H^Q_{fw}, H^Q_{bw})$$

  - The resulting matrices, H^P(lp x h) and H^Q(lq x h) contain the hidden vector representations of the passage and the question.

- A match-LSTM layer that tries to match the passage against the question.
  - Match-LSTM sequentially goes through the passage.
  - It uses the standard word-by-word attention mechanism to obtain attention weight vector(a_k), which indicates the degree of matching between a token in the passage with a token in the question.

$$a_k = \sum_{j=1}^{M} \alpha_{kj} * h^q_j$$

  - where, α_{kj} encodes the degree to which x^p_k in the passage is aligned with x^q_j in the question
  - A weighted version of the question obtained from the attention weight vector is combined with the current token of the passage to form a vector, which is then fed into standard one-directional LSTM to form a Match-LSTM.

$$m_k = [a_k h^p_k]^T$$
$$H^M_{fw} = \overrightarrow{LSTM}(m)$$

  - A similar LSTM is built in the reverse direction to encode the contexts from both directions for each token in the passage.

$$H^M_{bw} = \overleftarrow{LSTM}(m)$$

  - The final hidden state is the concatenation of both the states.

$$H^M = concat(H^M_{fw}, H^M_{bw})$$

- An Answer Pointer layer that uses Pointer-net to select a set of two tokens from the input passage.
  - All the tokens between these two tokens in the passage are treated as the answer.
  - The output from the Match-LSTM is used as the input for the pointer layer.

$$h^a_k = \overrightarrow{LSTM}(H^r \beta_k, h^a_{k-1})$$

  - where β_k is the attention weight vector generated for the k^{th} answer token generated in a similar way as match-LSTM
  - We then model the probability of generating the answer sequence as:

$$h(a|H^r) = p(a_s|H^r) * p(a_e|a_s, H^r)$$

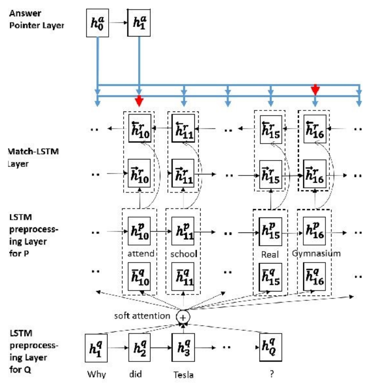

Fig. 2. Model Architecture

III. EXPERIMENTS

In this section we present our experiments and results obtained.

A. Data pre-processing

The SQuAD dataset that we use contains two files:

- **train-v1.1.json:** a train dataset with around 87 triplets
- **dev-v1.1.json:** a dev dataset with around 10k triplets

The data is in the form of triplets: context, question and answer-index span (Answer along with initial and final indices of the answer within the context passage). We used the original files to generate four files, each containing a tokenized version of the question, context, answer and answer span. We also made sure that the lines of each of these files are aligned. Each line in the answer span file contains two numbers: the first number refers to the index of the first word of the answer in the context paragraph. The second number is the index of the last word of the answer in the context paragraph.

We used Stanford GloVe embeddings to obtain vector representations of words. GloVe performs training on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. We have used GloVe word embeddings of dimensionality d = 50, 100, 200 and 300, 6B tokens and vocab size of 400k pre-trained on Wikipedia 2014 and Gigaword 5. Words not found in GloVe have been initialized as zero vectors. A basic tokenizer, split was used for word tokenization. Finally, the context and question were converted to token ids indexed against the entire vocabulary.

B. *Experiment Settings*

The entire model was build and trained using tensorflow. The tensorflow abstractions and functions made it easy to implement a Match LSTM by making small changes to the existing LSTM layer. Out of the two attention mechanisms tensorflow provides for constructing seq2seq models, we used the BahdanauAttention model[7]. A basic attention wrapper was used to do the intermediate calculations at each time step. To train our model, we compute the gradients and minimize the loss function. We used AdamOptimizer (tensorflow) to control the learning rate as we felt Adam algorithm has an advantage over the simple GradientDescentOptimizer (tensorflow) because Adam algorithm uses moving average of the parameters.

C. *Evaluation metric*

The initial SQuAD paper introduces the ExactMatch metric to evaluate the performace of the model. Exact match is a metric that measures the percentage of predictions that match one of the ground truth answers exactly.

D. *Results*

For training the original model, each epoch of batch-size 32 took us about 3 hours to run. We ran 30 such epochs, making the total run time of the entire model to almost 4 days. For training our bi-directional pre-processing LSTM model, each epoch took us about 7 hours to run, making the total run time of the entire model to over 8 days.

We obtained an Exact Match score of .59 on the test set using the original model. Our new model which uses bi-directional pre-processing LSTM obtained an Exact Match score of .63 which gave slightly higher score as compared to the original model.

E. *Future work*

Due to computational restrictions, we were only able to use GloVe word embeddings of dimensionality d = 50 to train our model. In future, we plan to run our model again on a GloVE word embeddings of dimentionality d = 300.

We would also like to explore various other state of the art models like R-Net[8], BiDAF[9] that achieved high Exact match score on the SQuAD dataset.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2016

[2] Karl Moritz Hermann, Tom Koisk, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, Phil Blunsom. Teaching Machines to Read and Comprehend, 2015.

[3] Shuohang Wang, Jing Jiang. Machine Comprehension Using Match-LSTM and Answer Pointer, 2016.

[4] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. arXiv preprint arXiv:1611.01604, 2016

[5] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. CoRR, abs/1611.01603, 2016.

[6] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al.End-to-end memory networks. In Advances in neural information processing systems, pages 24402448, 2015.

[7] Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio.Neural Machine Translation by Jointly Learning to Align and Translate, 2014

[8] R-NET: Machine Reading Comprehension with Self-matching Networks

[9] Christopher Clark, Matt Gardner.Simple and Effective Multi-Paragraph Reading Comprehension, 2017.

[10] Shikahr Murty. Question-Answering, TensorFlow implementation of Match-LSTM and Answer pointer

[11] SQuAD dataset.Preprocessing starter code. http://web.stanford.edu/class/cs224n/assignment4/index.html