Answering Reading Comprehension

Group No.: 8

Group Members:

- Ashish Kumar Singh (14142)
- Bhangale Pratik Anil (14173)
- Kunal Chaturvedi (14342)
- Rohith Mukku (14402)
- Shubham Agrawal (14671)
- Swati Gupta (14742)

1 Problem Statement

The problem statement of this project is to build a learning algorithm that enables the machine to read a comprehension and subsequently answer questions based on it. The ability of machines to read a comprehension and answer queries is one of the most important part of machine human interaction. This field is growing rapidly. It has an important role in generating replies for chatbots or smart answers of Google inbox.

Formalization: We have formulated the above problem as follows. We have a paragraph $p = p_0, . . . , p_m$ of m length and a question $q = q_0, . . . , q_n$ of n length related to the paragraph. The goal is to generate an answer span a with starting index a_i and ending index a_e of paragraph p. Machine learns a method to generate this span of words from paragraph, and here, it learns a predictor function,
$f(q, p) \rightarrow a$ from a training dataset of (q, p, a) triplets.

2 Datasets

(a) **SQuAD**
Stanford **Q**uestion **A**nswering **D**ataset (SQuAD)[1] is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. It has 100,000+ question-answer pairs on 500+ articles.
Related link: `https://rajpurkar.github.io/SQuAD-explorer/`

(b) **bAbI**
Developed by facebook[2], this dataset provides a set of training and test data. The training set is given by the true answers to questions, and the set of relevant statements for answering a given question, which may or may not be used by the learner. The tasks are set up so that correct answers are limited to a single word or a list of words.
Related link: `https://research.fb.com/downloads/babi/`

3 Literature Review

Following are the state-of-the-art algorithms which we have used in our approach to find answers of question based on the paragraph :

1. **Traditional QA systems:** In this approach, we induce a machine learning objective function that maps open-domain questions to queries over a database of web extractions, using manual features that take into consideration lexical and syntactic patterns occurring in the question text as well as the knowledge bank (Fader et al [3]).

2. **Deep Neural Networks:** In this approach, we use deep neural networks that uses memory units like RNNs and LSTMs. These are powerful sequence predictors that can be efficiently trained to learn to do inference over long term dependencies in the text. The drawback of this approach is the deficiency of structured network.

3. **Memory Networks:** In this approach, we aim to learn how to reason with inference components and a long-term memory component. Memory network serves as a knowledge base to recall facts from the past. The model tries to learn a scoring function to rank relevant memories (Graves et al [4]).

4 Proposed Approach

Memory Networks (MemNNs):
MemNNS tries to learn a scoring function to rank relevant memories. At prediction time, the model finds k relevant memories according to the scoring function and conditions its output based on these relevant memories. We intend to use MemNNs method as LSTMs conditioned on the relevant memories are much more effective than LSTMs conditioned on all the past memories. (Weston et al, [4]) describes the MemNNs model in detail. Given an input x:

1. Convert x to an internal feature representation $I(x)$.

2. Update memories m_i given the new input: $m_i = G(m_i, I(x), m), \forall i$.

3. Compute output features o given the new input and the memory: $o = O(I(x), m)$.

4. Finally, decode output features o to give the final response: $r = R(o)$.

MemNNs effectively break down the QA task into two steps: finding the most relevant pieces of memory for the given question, and then using those memories to generate a natural language answer, while both components can be trained together under a common loss function.

5 Novelty in our Approach:

1. **PointerNet:** Pointer Networks[5] are a neural architecture which learn the conditional probability of an output sequence with elements that are discrete tokens corresponding to positions in an input sequence rather than a large fixed vocabulary. Pointer Networks solve the problem of variable size output dictionaries using a mechanism of neural attention. We plan to use pointer networks in our approach to the problem to help insure consecutivity in the obtained answers without assuming consecutivity in the input.

2. **RaSoR:** Recurrent span representation[6] is an end-to-end neural network architecture to identify answer spans. The objective of this approach is to incorporate question as it plays an important part in predicting the answer. This is incorporated by creating concatenation of three embeddings. First is the question independent passage embedding, second is the question based embedding and third is passage combined with question based embedding.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[2] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698, 2015.

[3] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Paraphrase-driven learning for open question answering.

[4] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *CoRR*, abs/1410.5401, 2014.

[5] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2692–2700. Curran Associates, Inc., 2015.

[6] Kenton Lee, Tom Kwiatkowski, Ankur P. Parikh, and Dipanjan Das. Learning recurrent span representations for extractive question answering. *CoRR*, abs/1611.01436, 2016.