ABSTRACT

Question Answering (QA) is fundamental to natural language processing in that most NLP problems can be phrased as QA (Kumar et al., 2015). Current weakly supervised memory network models that have been proposed so far struggle at answering questions that involve relations among multiple entities (such as Facebook’s bAbi qa5-three-arg-relations in (Weston et al., 2015)). To address this problem of learning multi-argument multi-hop semantic relations for the purpose of QA, we propose a method that combines the jointly learned long-term read-write memory and attentive inference components of end-to-end memory networks (MemN2N) (Sukhbaatar et al., 2015) with distributed sentence vector representations encoded by a Skip-Thought model (Kiros et al., 2015). This choice to append Skip-Thought Vectors to the existing MemN2N framework is motivated by the fact that Skip-Thought Vectors have been shown to accurately model multi-argument semantic relations (Kiros et al., 2015).

1 INTRODUCTION

Question answering (QA) is a fundamental natural language processing task which requires understanding of the semantics of a text and reasoning over relevant facts. Recently, there has been a resurgence in neural network based models consisting of jointly learned long-term read-write memory and attentive inference components; these models (such as End-to-End Memory Networks (MemN2N) (Sukhbaatar et al., 2015), Dynamic Memory Networks (DMN) (Kumar et al., 2015), and Neural Reasoner (NR) (Peng et al., 2015)) can be trained on raw input-question-answer triplets to perform QA tasks. All of these models struggled with weakly supervised QA tasks that involved relations among multiple arguments (or in the case of Neural Reasoner were never thoroughly tested); weakly supervised QA tasks are tasks in which relevant statements are not labeled beforehand as being relevant.

We believe that models proposed thus far (Sukhbaatar et al., 2015; Peng et al., 2015; Kumar et al., 2015) struggled with these QA tasks because the encodings they used to embed statements and questions failed to capture the compositional semantics of whole sentences. To alleviate this problem, we instead propose using the recently introduced skip-thought encoder (Kiros et al., 2015) to produce embeddings that accurately capture the compositional semantics of whole sentences. Skip-thoughts (Kiros et al., 2015) is an encoder-decoder model that learns a vector representation of the semantics of a passage by training to reconstruct/predict the surrounding sentences of an encoded passage. These skip-thought vector representations of statements/questions are then fed into a memory network model (similar to MemN2N) to answers questions about the statements.

2 SKIP-THOUGHT MEMORY NETWORKS

Skip-Thought memory networks consist of 3 modules: an input embedder, multiple memory layers, and an output generator. More specifically:

input embedder The input embedder maps the sentences from statements and questions to Skip-thought vectors.
multiple memory layers Multiple memory layers compute a vector based on the match between questions and statements.

output generator The output module generates an answer based on a representation received from the memory layers.

2.1 Input Embedder

The input embedder first has to be trained (on the BookCorpus dataset (Kiros et al., 2015)) before it can map new sentences from desired statements and questions to Skip-Thought vectors.

2.1.1 Training the Skip-Thought Encoder

The Skip-Thoughts encoder is trained as outlined in (Kiros et al., 2015):

Sentence tuples are defined as \((s_{i-1}, s_i, s_{i+1})\). \(w_i^t\) defines the \(t\)-th word of sentence \(s_i\) and \(x_i^t\) is the word embedding for \(w_i^t\) gleaned from a word2vec model trained on Google News Corpus. The model consists of a GRU RNN encoder, GRU RNN decoder, and objective function.

Encoder: \(x_1^t, \ldots , x_N^t\) are the word embeddings of the words in sentence \(s_i\) and \(N\) is the number of words contained in sentence \(s_i\). The encoder produces a representation for the sentence \(s_i\) denoted by hidden state \(h_i^N\). Sentences are encoded by iterating the following four equations:

\[
\begin{align*}
    u^t &= \sigma(U_x x^t + W_x h^{t-1}) \\
    r^t &= \sigma(U_r x^t + W_r h^{t-1}) \\
    \bar{h}^t &= \tanh(U_x x^t + (r^t \odot W h^{t-1})) \\
    h^t &= u^t \odot \bar{h}^t + (1-u^t) \odot h^{t-1} 
\end{align*}
\]

\(\odot\) is an element-wise product; \(W_z, W_r, U_z, U_r\) are weight matrices. \(u^t\) will be referred to as the update gate, and \(r^t\) will be referred to as the reset gate. More details concerning gated recurrent units can found in (Chung et al., 2014)

Decoders: The decoders are GRU RNNs that condition on encoder output \(h_i\) to generate the surrounding sentences (one decoder generates \(s_i+1\) and another decoder generates \(s_i-1\)). Sentence representations \(h_i\) are decoded to generate the next sentence \(s_i+1\) by iterating through the following equations (previous sentences \(s_{i-1}\) are decoded for by the same equations by substituting \(s_{i-1}\) for \(s_i\)):

\[
\begin{align*}
    u^t &= \sigma(U_d x^t + W_d h^{t-1} + B_z h_i) \\
    r^t &= \sigma(U_r x^t + W_r h^{t-1} + B_r h_i) \\
    \bar{h}^t &= \tanh(U_d x^t + (r^t \odot W_d h^{t-1}) + B h_i) \\
    h_{i+1}^t &= u^t \odot \bar{h}^t + (1-u^t) \odot h^{t-1} 
\end{align*}
\]

\(B_z, B_r\) and \(B\) are biasing weight matrices for the update, reset, and hidden state respectively. Superscript \(d\) denotes that separate parameters are used by the \(s_i+1\) and the \(s_i-1\) decoders. The probability of word \(w_{i+1}^t\) conditioned on the previous \(t - 1\) words and the hidden state of the encoder \(h_i\) is proportional to the exponent of the product of the vocab \(v_{w_{i+1}^t}\) of each word \(h_{i+1}^t\) times \(h_{i+1}^t\):

\[
p(w_{i+1}^t | w_i^{<t}, h_i) \propto e^{v_{w_{i+1}^t} h_{i+1}^t} \tag{9}\]

Objective function: Given the proportion outlined above, the following objective can be optimized to predict/generate sentences \(s+1\) and \(s-1\) given sentence \(s\):

\[
\sum_t \log p(w_{i+1}^t | w_i^{<t}, h_i) + \sum_t \log p(w_{i}^t | w_{i-1}^{<t}, h_i) \tag{10}\]
2.1.2 Encoding desired sentences from statements and questions as skip-thought vectors

After the Skip-thoughts model finishes training on the BookCorpus dataset, it can be used to encode desired sentences from statements and questions as skip-thought vectors. Sentences are encoded using equations (1-4) outlined in the encoder section above. Arrays of encoded statement are referred to as $s$ and encoded questions are referred to as $q$.

The 4800 dimensional sentence vector representations that are yielded are the concatenation of vectors encoded by two different models that were trained on variations of the BookCorpus dataset. The first model is a unidirectional encoder that yields a 2400 dimensional vector. The second model is a bidirectional model consisting of forward and backward encoders that each yield a 1200 dimensional vector. This bidirectional model uses two encoders with variations of the input: the first encoder trains on the sentence in correct order, and the second encoder trains on the sentence in reverse. Another 2400 dimensional vector is then yielded by concatenating the outputs of the two encodings used by the second model. The 2400 dimensional vectors of the first and second models are then concatenated to form the 4800 dimensional vectors.

2.1.3 Encoding answers

Answers are encoded as a simple one-hot encoding referred to as $a$.

2.2 Multiple memory layers

Our model is first described in the single layer case, in which only a single memory hop is used. We then describe a model that allows multiple hops in memory by stacking multiple (single) layers (as first described in Sukhbaatar et al., 2015).

2.2.1 Single layer details

Input embedding: Arrays of encoded statement sentences $s$ are embedded into the main network as $m$ using a matrix $A$:

$$ m_i = s_i A $$ (11)

Encoded question sentences $q$ are embedded into the main network as $u$ using a matrix $B$:

$$ u = q B $$ (12)

The match between the question $u$ and each statement $m_i$ is computed by taking the softmax of the dot product of $u$ and $m_i$:

$$ p_i = \text{Softmax}(u^T B^T s_i A) = \text{Softmax}(u^T m_i) $$ (13)

Output embedding: Encoded answers $a$ are embedded into the main network as vector $c_i$ using a matrix $C$. The dot product of $p_i$ and $c_i$ is then taken to yield output vector $o$:

$$ o = p_i c_i $$ (14)

Answer prediction: Output vector $o$ and input embedding $u$ are summed; then the softmax of the dot product of this sum and and a weight matrix $W$ produces a predicted label:

$$ \hat{a} = \text{Softmax}((o + u)W) $$ (15)

The size of $W$ is determined by $d$ (the maximum number of input statements in a given triplet which is 110 in this case) times $V$ (the number of parameter used to encode the statement which is 4800 in this case). A cross-entropy loss function between current label $\hat{a}$ and true label $a$ is minimized to embed matrices $A$, $B$, $C$, and $W$. Training is performed using adadelta (Zeiler et al. 2012).
2.2.2 MULTILAYER DETAILS

The memory layers are stacked in the following way:

One layer (as described above) corresponds to one memory lookup. Many layers are stacked using the following equation in which the input to the layer above the previous is the sum of input \( u^k \) (times a linear mapping \( H \)) and output \( o^k \) (from layer \( k \)):

\[
    u^{k+1} = H u^k + o^k.
\]  

(16)

The linear mapping \( H \) is updated each iteration with other parameters such as \( W \). This results in the answer prediction at the top of the network being:

\[
    \hat{a} = \text{Softmax}( (o^K + u^K)W ).
\]

Also, The same input and output embeddings are used throughout different layers (such that \( A^k = A^{k+1} = \ldots \) and \( C^k = C^{k+1} = \ldots \)).

3 RELATED WORK

Other variations of memory networks have been proposed but they are either strongly supervised in that relevant supporting are fact explicitly labeled beforehand as being relevant (such as DMN) or they were not tested on multi-argument relations (such as Neural Reasoner and Neural Turing Machine (Graves et al., 2014)). Both DMN and Neural Reasoner used a GRU RNN (that is trained with the main loss/matching function of the memory network) to embed the input statements and questions (as opposed to our model which embed sentences using the skip-thoughts model). The original MemN2N (Sukhbaatar et al., 2015) used bag-of-words plus a positional encoder to embed the input statements and questions.

4 EXPERIMENTS

4.1 DETAILS OF TRAINING THE MEMORY NETWORK PORTION

Our models were trained with 1k training samples (from the bAbi dataset) for 20 epochs using adadelta and a learning rate of \( l = 0.000001 \). Weights were initialized to random values from a Gaussian distribution with \( \sigma = 0.1 \) and \( \mu = 0 \). A batch size of 10 is used (with cost averaged over batches). Memory arrays consist of only the most recent 110 sentences.

4.2 RESULTS

Our model (Skip-Thought Memory Network) is compared with strongly supervised models such as the very first (Step-by-step supervised) Memory Network and Dynamic Memory Network and weakly supervised end-to-end models such as Neural Reasoner.

|           | 5: Three Argument Relations |
|-----------|-----------------------------|
| **Step-by-step Supervision** |                |
| Memory Net-step     | 98.0%                      |
| Dynamic Memory Net  | 99.3%                      |
| **End-to-End**      |                |
| Memory Net-N2N      | 87.1%                      |
| Neural Reasoner     | -                          |
| Skip-Thought Memory Network | 41.7%                   |

Table 2: Results on bAbi5 three argument relations. The results of Memory Net step, Memory Net N2N, and Dynamic Memory Net, and Neural Reasoner are taken respectively from (Weston et al., 2014; Sukhbaatar et al., 2015; Kumar et al., 2015; Peng et al., 2015).

Our model failed to surpass the performance of existing end-to-end frameworks (most likely) because it was unable to converge completely. The cost function minimizes to approximately 3.0 and then starts to oscillate preventing it from decreasing any further. We think the cause of these convergence issues is that the weights of the skip-thought encoding are not influenced by the main
cross-entropy loss function between \( \hat{a} \) and true label \( a \). This could possibly be solved by tying the weights of the skip-thought encoder (after initial training on BookCorpus) to the weights \( W \) of the memory network.

5 CONCLUSIONS AND FUTURE WORK

We have proposed Skip-Thought Memory Networks, a framework that appends Skip-Thought Vectors to the existing MemN2N framework for the purpose of learning multi-argument multi-hop semantic relations for the purpose of QA.

The obvious future work would be to alter the framework in order to allow it to converge completely. To alleviate this, we suggest exploring methods for tying the weights of the skip-thought encoder (after initial training on BookCorpus) to the weights \( W \) of the memory network.

Code will be available at https://github.com/ethancaballero/Skip-Thought_Memory_Networks if anyone wants to try to improve on the model.

ACKNOWLEDGMENTS

We thank Patrick Ehlen and Ben Athiwaratkun for useful discussions.

References:

Cho, K., Merrienboer, B. van, Gulcehre, C., Bougares, C., Schwenk, H., and Bengio., Y. "Learning phrase representations using rn encoder-decoder for statistical machine translation". In Proceedings of EMNLP, pages 1724-1734, 2014.

Chung, J., Gihulcehre, C., Cho, K., and Bengio., Y. "Empirical evaluation of gated recurrent neural networks on sequence modeling". CoRR, arXiv preprint arXiv:1412.3555, 2014.

Graves, Alex, Wayne, Greg, and Danihelka, Ivo. "Neural Turing Machines." arXiv preprint arXiv:1410.5401 (2014).

Kumar, A., Irsoy,O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Gulrajani, I., and Socher, R. "Ask me anything: Dynamic memory networks for natural language processing". CoRR, arXiv preprint arXiv:1506.07285, 2015.

Kiros, Ryan, Zhu, Yukun, Salakhutdinov, Ruslan, Zemel, Richard S., Torralba, Antonio, Urtasun, Raquel, and Fidler, Sanja. "Skip-thought vectors". In NIPS, 2015. arXiv preprint arXiv:1506.06726

Peng, Baolin, Lu, Zhengdong, Li, Hang, Wong, Kam-Fai, "Towards Neural Network-based Reasoning". in NIPS RAM workshop arXiv preprint arXiv:1508.05508 (2015)

Sukhbaatar, Sainbayar, et al. "Weakly Supervised Memory Networks" arXiv preprint arXiv:1503.08895 (2015).

Weston, Jason, et al. "Towards AI-complete question answering: A set of prerequisite toy tasks." arXiv preprint arXiv:1502.05698 (2015).

Weston, Jason, Sumit Chopra, and Antoine Bordes. "Memory networks." arXiv preprint arXiv:1410.3916 (2014).

Zeiler, M. D. "Adadelta: an adaptive learning rate method". arXiv preprint arXiv:1212.5701, 2012.