Solving the Prerequisites: Improving Question Answering on the bAbI Dataset

Vincent Su, (John Miller , Jack Zhu) [1]

Introduction

The aim of this project is to make progress towards building a machine learning agent that understands natural language and can perform basic reasoning. Towards this nebulous goal, we focus on question answering: Can an agent answer a query based on a given set of natural language facts?

We combine LSTM sentence embedding models with an attention mechanism and obtain good results on the Facebook bAbI dataset [1], outperforming [2] on 1 task and achieving similar performance on several others.

1 Dataset

The Facebook bAbI dataset is a synthetic dataset of 20 toy question-answering tasks. Each task targets a specific skill that a general reasoning agent would be expected to have, such as answering yes/no questions or performing deduction over multiple sentences.

Each of the 20 tasks consists of 1,000 training examples and 1,000 test examples. The tasks are generated from a simulation of characters and objects interacting in a small, closed world, which produces ground text describing the scene and question/answer pairs. Supervision is provided in the form of answers for each of the questions and the location of relevant sentences in the input required to answer the question. For development, we randomly partition the 1,000 training examples for each task into 900 example training sets and 100 example development sets.

1.1 Example

Each training example consists of a sequence of supporting facts, a question, an answer, and labels indicating the sentences relevant to answering the question:

```
1 Mary moved to the bathroom.
2 Sandra journeyed to the bedroom.
3 Mary got the football there.
4 John went to the kitchen.
5 Mary went back to the kitchen.
6 Mary went back to the garden.
7 Where is the football?  garden 3 6
```

Support sentences are numbered (sequentially), and questions are labeled with the correct answer as well as the index of the relevant sentences. This structure is repeated for each different tasks.

Formally, each example is represented by a 5-tuple (q, f, a, r), where q denotes the question, $f = \{f^1, f^2, \ldots, f^k\}$ is a list of supporting sentences, a denotes the answer, and $r = \{r_1, \ldots, r_n\}$ denotes the set of relevant sentences. Further, f_i^j denotes the i-th word in the j-th supporting sentence, and q_i denotes the i-th word in the question.

2 Related Work

Question answering is a long-studied topic in natural language processing and machine learning. There are a variety of approaches to the problem ranging from constructing and reasoning over knowledge bases [3], inferring and reasoning over latent logical forms [4], to using neural networks [5].

Several authors have studied using neural networks and attention for question answering. Our work and conceptual framework is inspired by the memory networks of [1], though our implementation of the attention mechanism and inference modules is different. The work of Kumar et. al [6] applies attention and recurrent neural networks including the bAbi tasks and outperforms [1] on several of the tasks.

[1]Partners in CS221 but not in CS229

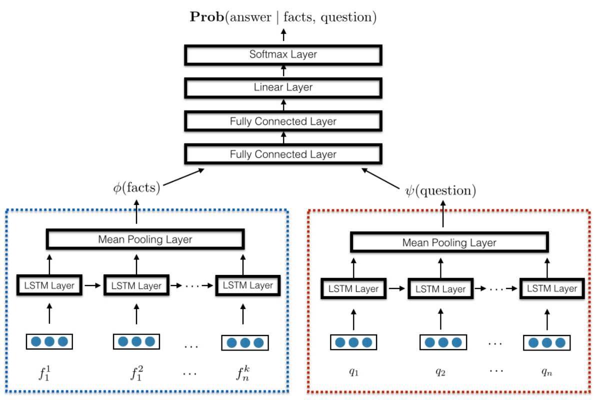

Figure 1: LSTM Sentence Embedding Model

3 Approach

Each of our models can conceptually be broken down into 3 parts. The first two parts are representation modules: one module converts the supporting facts into a vector $\phi(f)$, and another module converts the question into a vector $\psi(q)$. The third module is a reasoning module: it takes as input $\phi(f)$ and $\psi(q)$ and outputs a distribution $\mathbf{Prob}(a \mid q, f)$ over words in the vocabulary, V. We describe several implementations of this framework below.

3.1 LSTM Sentence Embedding Model

To better capture the semantic structure of the supporting facts and questions, the second model uses sentence representations generated from a Long Short-Term Memory network (LSTM) [7].

LSTM's are recurrent neural networks that can be trained to compose vector representations of sentences from word embeddings [10]. In our models, at time t, the LSTM takes as input a new word w_t and updates its hidden state according to

$$h_t = \mathcal{H}(w_t, h_{t-1}, c_{t-1}),$$

where $\mathcal{H}$ is a variant of the LSTM [7] with update equations:

$$\begin{aligned}
i_t &= \sigma\left(W_{xi}w_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right)\\
ft &= \sigma\left(W_{xf}w_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right)\\
c_t &= f_t c_{t-1} + i_t \tanh\left(W_{xc}w_t + W_{hc}h_{t-1} + b_c\right)\\
o_t &= \sigma\left(W_{xo}w_t + W_{ho}h_{t-1} + b_o\right)\\
h_t &= o_t \tanh(c_t).
\end{aligned}$$

The vectors i_t, f_t, o_t and c_t are called the "input gate", "forget gate", "output gate" and cell activation vectors, respectively, and σ is the sigmoid non-linearity. There are all the same size as the hidden vector h.

To construct $\phi(f)$, we first embed all of the supporting facts word-by-word using the LSTM model. Then, $\phi(f)$ is constructed by averaging the intermediate hidden states using a "mean-pooling layer"

$$\phi(f) = \frac{1}{T}\sum_{t=1}^{T} h_t,$$

where T is the number of words in the supporting facts. Constructing $\psi(q)$ uses an identical procedure.

Given $\phi(f)$ and $\psi(q)$, we compute $\mathbf{Prob}(a \mid q, f)$ by concatenating $\phi(f)$ and $\psi(q)$ and using this as input to a feed-forward neural network with 2 hidden layers. The network uses relu non-linearities and outputs a categorical distribution over all $|V|$ words in the vocabulary. This model is depicted graphically in Figure 1.

Objective. The model is trained by minimizing the regularized negative log-likelihood

$$J(\Theta) = \sum_{(q,f,a,r)\in\mathcal{D}_{\text{train}}} -\log \mathbf{Prob}(a \mid q, f; \Theta) + \lambda\|\Theta\|_2^2$$

where Θ is the parameters of the model.

3.2 Attention Mechanism

One of the weaknesses of the LSTM sentence embedding model is handling questions with large numbers of supporting facts, especially with the number of relevant sentences is small. In these cases, the LSTM model likely experiences difficulty with preserving information across time-steps and learning long-term dependencies.

To address this issue, we extend the LSTM embedding model with an attention mechanism. The attention mechanism is implemented as a LSTM that, at each time step, takes as input the question and the current supporting fact and predicts with the supporting fact is "relevant." Only the relevant sentences are then fed as input into the previously described LSTM sentence embedding model. The mechanism is trained using the relevant sentence supervision signal provided with each example.

More concretely, let $\Phi(s)$ denote the bag-of-words representation of sentence s. Consider an example (f, q, a, r), where $f = \{f^1, \ldots, f^k\}$. For $t = 1, \ldots, k$, we form

$$x_t = U \begin{bmatrix} \Phi(f^t) \\ \Phi(q) \end{bmatrix}$$

where $U \in \mathbf{R}^{D\times 2|V|}$ is a learned embedding matrix. At step t, the attention model takes as input x_t and outputs $\mathbf{Prob}(r_t \mid x_{1:t})$, an estimate if fact f^t is relevant.

During inference, any sentence f^t with $\mathbf{Prob}(r_t \mid x_{1:t}) > 0.5$ is deemed "relevant." If no sentence has $\mathbf{Prob}(r_t \mid x_{1:t}) > 0.5$, then we take the two sentences f^i, f^j with the highest probability under the model.

The model takes advantage of the relevant sentence labels and can be trained in a supervised fashion. We found that joint training of the attention mechanism and the question answering module yields the best results. Let f^* denote the set of sentences declared relevant for some example. The full model is trained by minimizing

$$J(\Theta) = \sum_{(q,f,a,r)\in\mathcal{D}} -\log \mathbf{Prob}(a \mid q, f^*; \Theta) + \left(\sum_{t=1}^{T_f} -\log \mathbf{Prob}(r_t \mid q, f; \Theta) \right) + \lambda\|\Theta\|_2^2.$$

3.3 Implementation and Training

All of our models were implemented from scratch using Theano[8][9].

In all of our experiments, we use Adagrad [11] to minimize $J(\Theta)$, which is in general non-convex. We initialize the word embeddings with pretrained-GloVe vectors of dimension 50 [12]. The other model parameters are all initialized with an i.i.d. Gaussian of variance 0.001 in every entry. We use a mini-batch of size 50, an initial learning rate of 0.01, $\lambda = 10^{-6}$, and by default, use 256 LSTM cells and hidden layers of dimension 128 in the feed-forward model.

4 Experiments

We evaluate the models introduced in the previous section on the synthetic tasks in the bAbi dataset. For each of these models, the reasoning module computes $\mathbf{Prob}(a \mid q, f)$ over single answers only since it is implemented as a feed-forward neural network rather than an RNN. Therefore, we only evaluate on 18 of the 20 tasks in bAbi since the remaining tasks, list (8) and path finding (19), require a sequence of answers in list form.

4.1 Evaluation Metric

All of the models are evaluated individually on each task using accuracy, defined to be the number of correct answers divided by the number of questions. Following [1], we consider the answer to each question to be a single word. This makes evaluating correctness of an answer unambiguous.

4.2 Results

Model accuracy on the test sets for the 20 tasks are reported in Table 1. The first three columns display our baseline bag-of-words model, our LSTM sentence embedding model, and the LSTM model with attention. For each task, we perform early stopping based on performance on the held out development set. We also include metrics from the highest-performing augmented Memory Network implementation in [1], and the results of a human oracle. Since our models are incompatible with tasks that require multiple answers, we designate those with a question mark.

As expected, our LSTM and LSTM + Attention models generally outperform the baseline on most tasks. Adding attention gives an additional performance boost on many tasks. Possible explanations for weakers performances are given in the next section. While our LSTM+Attention model performs worse the MemNet implementation in [1], we approach its accuracy in many tasks. In particular, our performance approaches within 3% of the MemNet on tasks 17, 18, 20, and even exceeding it by 2% on task 7. We believe that more careful cross-validation of model size and hyperparameter tuning can slightly improve our reported metrics.

Table 1: Preliminary Results on Development Set

	LSTM	LSTM with Attention	MemNet [1]
1 Single Supporting Fact	44	82	100
2 Two Supporting Facts	24	27	100
3 Three Supporting Facts	24	25	100
4 Two Arg. Relations	74	63	100
5 Three Arg. Relations	60	88	98
6 Yes/No Questions	62	82	100
7 Counting	77	**87**	85
8 Lists/Sets	?	?	91
9 Simple Negation	64	88	100
10 Indefinite Knowledge	44	76	98
11 Basic Coreference	60	82	100
12 Conjunction	53	75	100
13 Compound Coref.	89	90	100
14 Time Reasoning	39	67	99
15 Basic Deduction	52	32	100
16 Basic Induction	45	25	100
17 Positional Reasoning	58	63	65
18 Size Reasoning	91	92	95
19 Path Finding	?	?	36
20 Agent's Motivations	96	99	100

5 Discussion

5.1 Overview

In this section, we discuss and analyze the performance of our LSTM embedding models relative to the baseline bag-of-words model. Our model featured two improvements, an LSTM model which embeds the stories, and then an attention model. We find that the base LSTM model with word vectors gives significant boosts for most of the tasks over the bag-of-words.

The attention mechanism provided an additional boost on many of the tasks, but it also severely impacted performance on tasks which required more complex reasoning. This is potentially because the attention mechanism failed to mark enough important sentences as relevant, making it impossible for the reasoning module to draw correct inferences. We conclude that attention mechanisms have a lot of potential to help with natural language processing, but our particular implementation was not well suited for certain tasks. We provide a concrete example with discussion for our best task performance relative to [2]

Task 7 - Counting

```
Supporting Facts:
daniel took the milk there .
john moved to the hallway .
daniel left the milk .
daniel journeyed to the office .

Question:  how many objects is daniel carrying ?
True Answer:  none
Predicted Answer:  none
Predicted Relevant Facts:  daniel took the milk there .
daniel left the milk .
```

The LSTM's capacity for sequence learning is most clearly demonstrated in this task. Our LSTM model with attention boosts performance to 87%, which outperforms the stated result in the MemNet paper. The example above shows how the LSTM can correctly infer the effect of compounding sentences sequentially, whereas a Bag-of-Words classifier might simply have counted three instances of "daniel" in the story and predicted three as the answer.

6 Conclusion

We applied LSTM embedding models with attention to a classic AI task, question answering on the bAbI dataset. The attention mechanism greatly improved performance on many tasks. For questions which required more inferential reasoning, the attention model was often too greedy and fed the LSTM inference model insufficient information to correctly answer the question. Thus, we conclude that the performance boost from the attention mechanism is highly sensitive to its specific implementation. Our experimentation found that the LSTM-based attention mechanism was well-suited to tasks with a very sequential structure like counting (7), for which we report a better accuracy than the optimized MemNet.

References

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

[2] Weston, Jason, Chopra, Sumit, and Bordes, Antoine. Memory networks. CoRR, abs/1410.3916, 2014.

[3] A. Yates, M. Banko, M. Broadhead, M. J. Cafarella, O. Etzioni, and S. Soderland. Textrunner: Open information extraction on the web. In HLT-NAACL (Demonstrations), 2007.

[4] J. Berant, A. Chou, R. Frostig, P. Liang. Semantic parsing on Freebase from question-answer pairs. Empirical Methods in Natural Language Processing (EMNLP), 2013.

[5] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. Joint Learning of Words and Meaning Representations for Open-Text Semantic Parsing. AISTATS, 2012.

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

[7] Hochreiter, Sepp and Schmidhuber, Jügen. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[8] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. J. Goodfellow, A. Bergeron, N. Bouchard, and Y. Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

[9] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for Scientific Computing Conference (SciPy).

[10] K. Tai, R. Socher, and C. D. Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. ACL 2015.

[11] J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive sub- gradient methods for online learning and stochastic optimization. In Conference on Learning Theory (COLT).

[12] J. Pennington, R. Socher, and C. D. Manning. 2014. Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP).