John A. Paulson School of Engineering and Applied Sciences

Project Update - CS287

Nicolas Drizard & Virgile Audi
nicolas.drizard@g.harvard.edu   vaudi@g.harvard.edu

April 22, 2016

Introduction

The focus of this final project is about non-factoid question answering. To solve the problem on non-factoid QA, we chose to implement memory network type of models and in particular the dynamic memory network presented in [Kumar et al., 2015]. For this first project update, we mainly worked on pre-processing the data form the bAbi dataset, implement a count-base baseline as well as looking at a one hop memory weakly supervised memory network.

1 Pre-processing

The goal of testing the created models on the bAbi dataset is to be able to have a model that works on all 20 pre-defined tasks without having to fine tune the prediction step. This constrain impacted the choice of pre-processing we adopted for this project. We wanted to create a pre-processing script that given a set of tasks and their respective training and test sets, would output a unique format of input and output data.

The first step of the pre-processing was to create a mapping between words present in the training and test sets and an index, and vice-versa. A particularity of the bAbi dataset is that even though it is not a Multiple Choice QA dataset, the number of possible answers is not infinite. We therefore decided to create mappings that given a set of tasks to solve, would start by indexing the words that appear as answers followed by the other words present in training and test stories and questions.

Given the tasks and the mappings mentioned above, the pre-processing script returns four matrices. The first matrix contains the sentences of the different stories using a bag of words representation, as well as the index of the task it belongs to. In this matrix, we concatenate the stories from all the inputed tasks, and sentences are therefore globally indexed and the index of a sentence in a story is disregarded. Using a similar trick as the one used in homework 1, we introduced padding in order to get a standard sentence size equal to the length of the longest sentence in training and testing.

We used the same construction for the second matrix that corresponds to the bag of words representation of the questions. We also output a vector that contains the index of the answers of the question. Finally we created a fourth matrix that maps the questions to the relevant sentences in the stories matrix. Each row of this matrix contains:

- the index of the first sentence of the corresponding story in the stories matrix,
- the index of the last sentence of the corresponding story for this particular questions,
- the indices of the supporting facts in the stories matrix

The number of supporting facts might vary from task to task, and has a maximum of 3. If we are dealing with less than 3 facts, we fill the row with 0 entries.

2 Count-Base Baseline

2.1 Model Presentation

Question/Word embedding

???

Sentence Matching The count-model worked rather well just using the previous features. We felt never the less that implementing a first sentence matching step could help improve results. The goal of this step was to restrict the number of sentences in which we could find the answer to a given question. To do so, we used co-occurences of words in the questions with words in the sentences of the story. This is similar to the algorithm presented in the MCtest paper [Richardson and Renshaw, 2013]. This step allows to look for one or more supporting facts. If the desired number of facts is one, the function outputs the highest scoring sentence on account of word co-occurences. Consider the following example:

Mary picked up the football.
Mary went to the garden.
John walked to the garden.
Where is the football?

If we look for more than one supporting fact, then word counts would not help us answer the question, as the only scoring sentence (if we disregard stop words such as the) would be the first sentence. So the function we coded uses an iterative approach to narrow the search for supporting facts. It then looks for sentences that have co-occurences with the sentences that scored in the first question/sentence matching. The fact that "Mary" appears both in the first and second sentence will allow us to return the relevant facts to answering the question.

A potential downside to this method is that it will most of the time find more than three supporting facts which is the maximum amount of supporting facts needed to answer questions of any tasks. Nevertheless, this step would only be necessary for the count-base model. And since the use of this model was just to act as a baseline to the memory networks, we decided that the model performed sufficiently well to leave it to that.

2.2 Results

3 Memory Network

If we didn't have sufficient time to implement a first memory network, we wanted to present the first memory network we will implement. The model was introduced in [Sukhbaatar et al., 2015] and can be summarised in the following graphical representation:

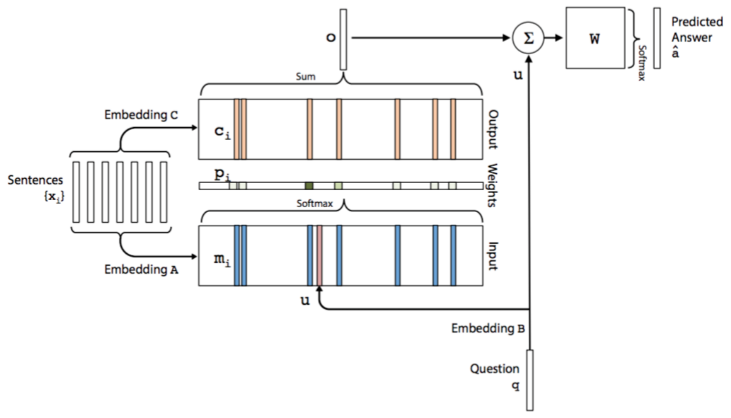

Figure 3.1: 1-Hop Memory Network

This model implements a single memory hop operation. This model takes for inputs the question and the sentences of the story. The sentences of the story are then embedded with two differents look-up tables. The first look-up table (which corresponds to the matrix A in the above diagram) will be used to store the sentences of the stories as input in memory. These embedded representation of the story's sentences will then be combined with the embedded representation of the question (using the matrix B) using a dot product. The result of this dot product is then passed through a softmax layer to give a probability distribution over which sentence is likely to give information about the answer. The memory output vector o results from a weighted sum of the output embeddings of the sentences (using matrix C) using the probability distribution p. We then apply a weight matrix to the sum of question embeddings and the output vector o followed by a softmax to predict the answer. We can summarize this process with the formula:

$$\hat{a} = \text{softmax}\,(W(o+u))$$

where: $o = \sum_{i=1} \text{softmax}(u^T m_i)$, with u being the embedded representation of the question and m_i the embedded represenation of the sentence i in the story.

The embedded representation of sentences and questions can either be obtained by summing the embeddings of their words or by concatenating them. We plan on testing the two approaches.

Finally, we will implement a temporallity feature by modifying the embedded representation of the sentences by adding a term depending on the index of the sentence in the story.

4 Future Steps

If we plan to code this model using the regular "nn" module, we would also like to use the "nn-graph" module. Indeed, once this model implemented, we would like to implement a variation by adding mutliple hops to memory. This model is also presented in [Sukhbaatar et al., 2015] and its structure could be too complex for the "nn" module. We will then tackle the dynamic memory network that we introduced in the proposal and can be found in [Kumar et al., 2015].

References

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

[Richardson and Renshaw, 2013] Richardson, B. and Renshaw, E. (2013). Mctest: A challenge dataset for the open-domain machine comprehension of text. *Conference on Empirical Methods in Natural Language Processing.*

[Sukhbaatar et al., 2015] Sukhbaatar, S., szlam, a., Weston, J., and Fergus, R. (2015). End-to-end memory networks. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 2440–2448. Curran Associates, Inc.