STAT 946 Deep Learning

Project Title: QA with Memory Networks

Project Number: 4

Group Members:

Surname, First Name	Student ID	Your Department
Blouw, Peter	20388020	PHIL
Gosmann, Jan	20547828	SYDE

One of our group members is taking STAT 841 (Classification)?

- ☑ No
- ☐ Yes

Your project falls into one of the following categories. Check the boxes which describe your project the best.

1. ☐ **Kaggle project.** Our project is a Kaggle competition.
   - Our rank in the competition is ... .
   - The best Kaggle score in this competition is ... ..., and our score is ....
2. ☑ **New algorithm.** We developed a new algorithm and demonstrated (theoretically and/or empirically) why our technique is better (or worse) than other algorithms. (Note: A negative result does not lose marks, as long as you followed proper theoretical and/or experimental techniques).
3. ☐ **Application.** We applied known algorithm(s) to some domain.
   - ☐ We applied the algorithm(s) to our own research problem.
     - ☐ We used an existing implementation of the algorithm(s).
     - ☐ We implemented the algorithm(s) ourself.
   - ☐ We tried to reproduce results of someone else's paper.
     - ☐ We used an existing implementation of the algorithm(s).
     - ☐ We implemented the algorithm(s) ourself.

Our most significant contributions are (List at most three):

(a) A new memory network architecture tested on a realistic QA dataset.

(b) A statistical analysis of the impact of different architecture components.

1 Introduction

A considerable amount of recent machine learning research has been directed towards the development of models capable of understanding natural language in the same way that humans do. In one application of this research, a class of models called *memory networks* have been used to perform sophisticated question-answering tasks [2, 6, 8]. Early results are impressive, with memory networks achieving near-perfect accuracy on a widely used dataset of 20 tasks released by Facebook [7].

Unfortunately, this dataset has a number of limitations. It consists of questions paired with short stories generated from a vocabulary of only a few dozen words. Moreover, the sentences in each story are all syntactically quite simple. Given these limitations, the goal of our project is to develop a new type of memory network that can achieve good performance on a more realistic dataset. Specifically, we use MCTest, which consists of 660 short stories that are each paired with a set of four multiple choice questions [5]. To accommodate the complexity of the stories in MCTest, we implement a memory network that makes use of simple forms of linguistic preprocessing such as coreference resolution and semantic role labelling.

In what follows, we first provide a brief summary of both MCTest and the technical details of memory networks. After highlighting a few known problems associated with achieving good performance on MCTest using memory networks [2], we motivate our proposed solutions to these problems. We then present our experimental results, which indicate that our proposed solutions are largely unsuccessful. We conclude with a discussion of potential reasons for this lack of success.

2 QA with the MCTest Dataset

The MCTest dataset is designed to provide a benchmark for research on the machine comprehension of text [5]. Each story in the dataset is between 150–300 words and is understandable to a young child. All questions are multiple choice. An example of a story and a question are shown in Figure 1 below.

We chose this dataset for two reasons. First, it provides a good test of the applicability of memory networks to question-answering tasks that are not overly simplistic. Second, there is at least one example of prior work to which we can compare our results [2].

3 Memory Networks

Memory networks are a class of machine learning models that perform inference while exploiting a possibly unbounded memory. A typical memory network consists of four modules, each of which computes a potentially learned function. The *Input* module translates a linguistic expression into a feature representation. The *Generalization* module then updates the network's memory using the features provided by the input module. When a question is posed to the network, the *Output* module generates a new feature representation based on both the question and the current state

Paul woke up at 8. He was very happy because today he got to go to his favourite thing, the fair. Paul's mother Beth was taking him to the fair. After finishing breakfast at 9, Paul got in the car with his mom. At 10 they got to Jim's house to pick him up. Jim was Paul's best friend. Then at 11, they picked up Beth's boyfriend Hank. After driving for one more hour they all finally got to the fair at 12. They had all been looking forward to this for a very long time. Beth was a bit annoyed by having to drive so much to get here, but she loved her son very much so the trouble was okay. Everyone had a great time, most of all, Paul. Gail's favourite ride was Ferris. Hank's favourite ride was the Ghoster. It was very scary. Paul's favourite ride was the same as Hank's.

What was Paul's favourite ride? A) Hank B) car C) Ferris D) Ghoster

Figure 1: An example story from MCTest and one multiple choice question.

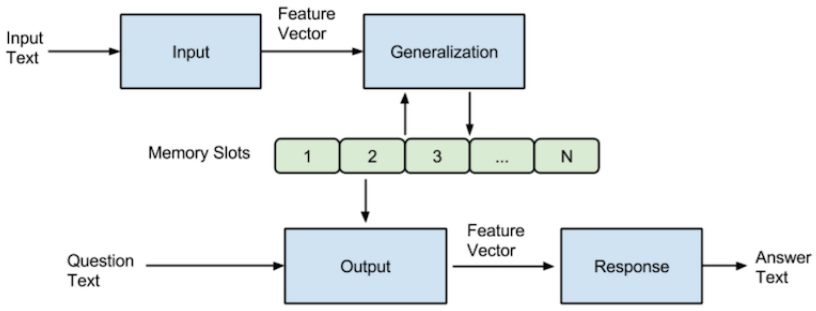


Figure 2: Memory network architecture (reproduced from Kapashi and Shah, 2015).

of the memory. Finally, the *Response* module predicts an answer to the question using the feature representation produced by the output module. Figure 2 illustrates the relationships between these modules in more detail.

3.1 Basic Implementation

In the simple implementation of a memory network described by Weston et al. [8], each module that computes a learned function is trained in a supervised manner. For instance, the output module is directly trained to select k supporting statements from the memory when a question is provided as input (i.e. the correct supporting statements are labelled in the training data). k is typically set to 2.

To select supporting statements, the output module scores each candidate statement stored in the memory against the question and chooses the highest scoring candidate:

$$s = \underset{i=1..n}{\operatorname{argmax}} S_O(q, m_i) \tag{1}$$

where n is the number of statements stored in memory, and m_i is a candidate statement. Additional supporting statements are selected in the same way, while taking into account the previously chosen statements. The scoring function S_O is an embedding model that maps two items into a vector space and returns their dot product:

$$S_O(x, y) = \phi_x(x)^T U^T U \phi_y(y) \tag{2}$$

where ϕ_X and ϕ_y are functions that map linguistic expressions to binary bag-of-words vectors. U is a learned embedding matrix.

Following this selection process, the question and supporting statements are then presented as input to the response module. Candidate responses (i.e. individual words in the model's vocabulary) are scored and a winner is chosen using an argmax expression of the same form as (1). The scoring function used to compute this argmax is defined by a new embedding model of the same form as (2). In all cases, training is performed using a hinge loss defined with respect to a margin between the scores assigned to correct and incorrect candidates.

3.2 End-to-End Implementation

To avoid the need for direct supervision of each module, Sukhbaater et al. [6] propose a memory network that can be trained solely from question-answer pairs. Their idea is to avoid using an argmax to select a discrete set of supporting facts by producing a continuous weighting of the memory entries:

$$p_i = \text{Softmax}(u^T m_i) \tag{3}$$

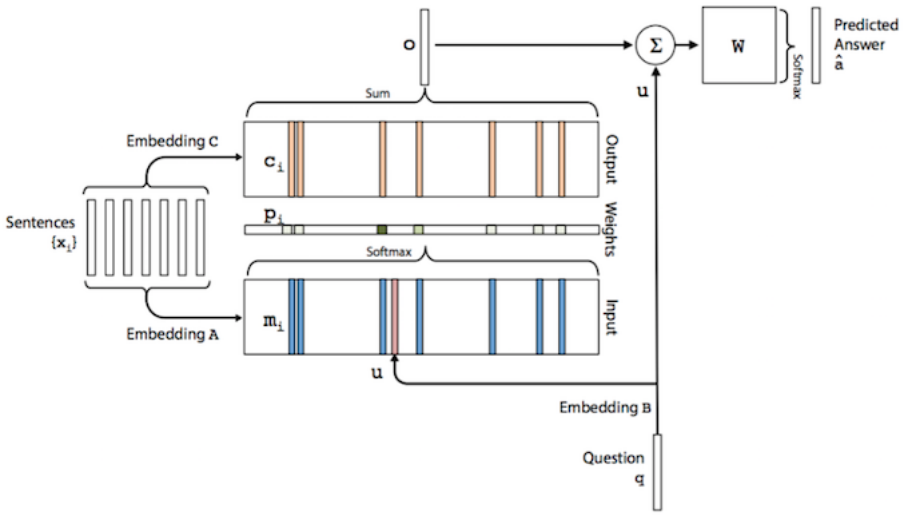


Figure 3: Sukhbaatar et al's end-to-end memory network

where u is the question embedding and m_i are the memory items. This weighting is then used to produce an output vector o with the memory embedding matrix C given by

$$o = \sum_i p_i c_i. \tag{4}$$

Finally, a prediction is obtained with the embedding matrix W of the response layer as

$$\hat{a} = \text{Softmax}(W(o + u)). \tag{5}$$

Because the softmax function is smooth, gradients can be backpropogated from the output layer to the input layer, and the entire model can be trained using a cross-entropy loss function.

4 Prior Results

To our knowledge, the only previous application of a memory network to MCTest is the work of Kapashi and Shah [2]. They use the end-to-end memory network just described to obtain the results presented in Figure 4. Since chance performance on MCTest is approximately 25%, these results suggest that the network drastically overfits the training data. Kapashi and Shah also suggest that the poor test error might be due to the fact that the stories in MCTest frequently use pronouns, which a memory network cannot easily learn to associate with the appropriate referent.

Dataset	WMemNN (Extended)	
	Train	Test
MC160	92.9%	36%
MC500	98.3%	34.2%

Figure 4: MCTest Results using an end-to-end memory network. Reproduced from Kapashi and Shah (2015). MC160 and MC500 respectively contain 160 and 500 stories split up into train and test sets.

In the next section, we describe extensions to the end-to-end memory network that are motivated by these possible explanations of Kapashi and Shah's poor results.

5 A New End-to-End Memory Network

Our extensions to the model architecture described by Sukhbaatar et al. are primarily motivated by the idea that performance might be improved if the embeddings of the memory items are designed to incorporate domain-general linguistic features. We hypothesize that using such features will enable the model to avoid learning task-specific embeddings for the training data that fail to generalize to the test data. Furthermore, by fixing the memory embedding matrices, the model only learns the input and output embedding matrices, which reduces the overall number of parameters (which in turn should reduce the likelihood of overfitting).

Below, we discuss a six techniques for improving the memory embeddings: holographic reduced representations (HRRs), word2vec features, timetags, coreference resolution, role-labelling, and pre-initialized embedding matrices. HRRs and word2vec are two method for specifying the memory embedding which then can be combined with any combination of the other methods.

5.1 Holographic Reduced Representations (HRRs)

HRRs are method for representing compositional structures with distributed vector representations [4]. To use HRRs, we assign a random high-dimensional unit vector to each unique word in the training and test sets, while requiring that vectors for all words be close to orthogonal (i.e. the cosine of the angle between two vectors is never greater than 0.15). To represent a sentence, we sum all of the vectors for the words occurring within it. In our baseline experiments, we encode each memory item as a sum of HRRs in this way, and learn the input and output embedding matrices.

5.2 Word2vec

As described by Mikolov et al. [3], word2vec is a set of algorithms for learning distributed representations of words from text corpora. We use pretrained word2vec vectors for each word in our model's vocabulary in place of the HRRs. To embed a memory item, we simply the sum the vectors for each word in the item as before. The rationale or doing this is that the spatial relationships present amongst the word2vec vectors will support better generalization [2]. For example, if the model is trained to correctly answer a question concerning cars, then we can expect some generalization to occur if the test set contains a question about taxis, given that the word2vec vectors for "car" and "taxi" are similar.

5.3 Time Features

For some questions the temporal order of facts in the memory might be important. For this reason we added time tags indicating the recency of each memory item. To generate the time tag for the i-th memory item we first generate a vector

$$\hat{t}_i = \mathcal{F}^{-1}(\mathcal{F}(v) \odot \mathcal{F}(w)^s) \tag{6}$$

where v and w are all zero vectors, except $v_1 = 1$ and $w_2 = 1$, $\mathcal{F}$ is the discrete Fourier transform, s is a shift hyper-parameter, and all operations in the Fourier space are element-wise (i.e., $\odot$ is the element-wise product).

Intuitively this gives a vector with a one at the first position for the most recent memory and it gets slowly shifted through the vector components as memories get older. Note that the formulation in Fourier space allows non-integer or partial shifts, thus allowing adjacent time tags to have a non-zero similarity as measured by the cosine. The hyper-parameter s controls the amount of shift (or inversely similarity) between adjacent time tags. After $1/s$ shifts the similarity will be 0.

For the use with the memory network we use a random orthonormal projection matrix P to obtain a distributed time tag representation t_i from the $\hat{t}_i$ as

$$t_i = P\hat{t}_i. \tag{7}$$

When implementing timetags in our experiments, we concatenate the HRR encoding of a memory item with a timetag.

5.4 Coreference resolution

Co-reference resolution is the problem matching words such as pronouns (e.g. "she","it") to the objects they refer to (e.g. "Alice","the dog"). Because many of the stories in MCTest contain sentences that refer to a central character repeatedly through pronouns, performing co-reference resolution might allow the model to learn to associate individuals mentioned in a question with particular memory items even if the individual is only indirectly mentioned in the memory item. Implementationally, we used the Stanford CoreNLP library to link pronouns to their referents. We then wrote a simple parser that replaced each pronoun with its corresponding referent. For example, if the word "she" refers to Alice, we simply replace "she" with "Alice".

5.5 Roles

Most of the stories in MCTest describe activities involving a few central characters. However, because the character names are not often repeated across stories, it is difficult for the model to transfer learning about one character to another, similar character. To solve this problem, we performed named entity recognition on each story using the NLTK library and replaced each character name with a generic "AGENT" representation or semantic role (we use numbered role representations for the different characters in a story). We hoped that using a common representation for main characters and the like in the stories would support the transfer of learning from the training set to the test set.

5.6 Preinitialized question and output embedding

During training the memory network has to learn embeddings for the questions and output that is to some degree similar to the embedding used for the memory. Thus, the performance might be increased by initializing those learned matrices with the HRR or word2vec embeddings instead of random weights.

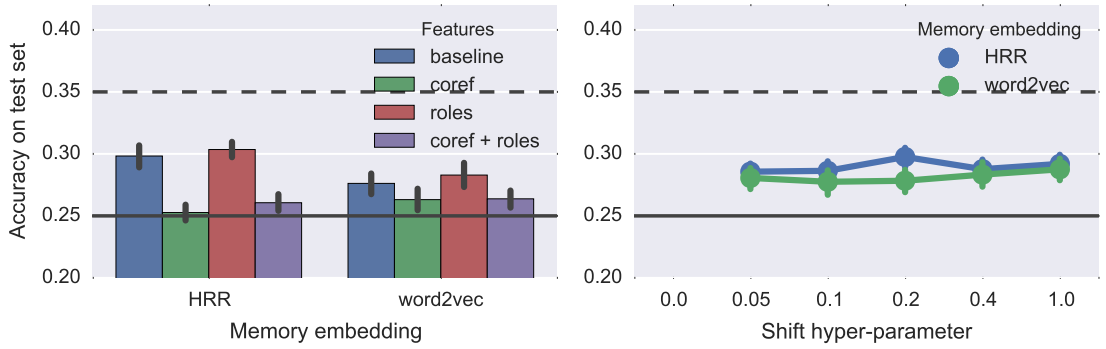
6 Experiments

As a starting point, we first re-implemented the basic memory network, and tested it on the first two tasks in FB20. We achieved results that are comparable but slightly worse than those reported in Weston et al. [7]. However, given that MCTest requires the use of a weakly supervised memory network, we implemented this basic model as an exercise and did not pursue improved results.

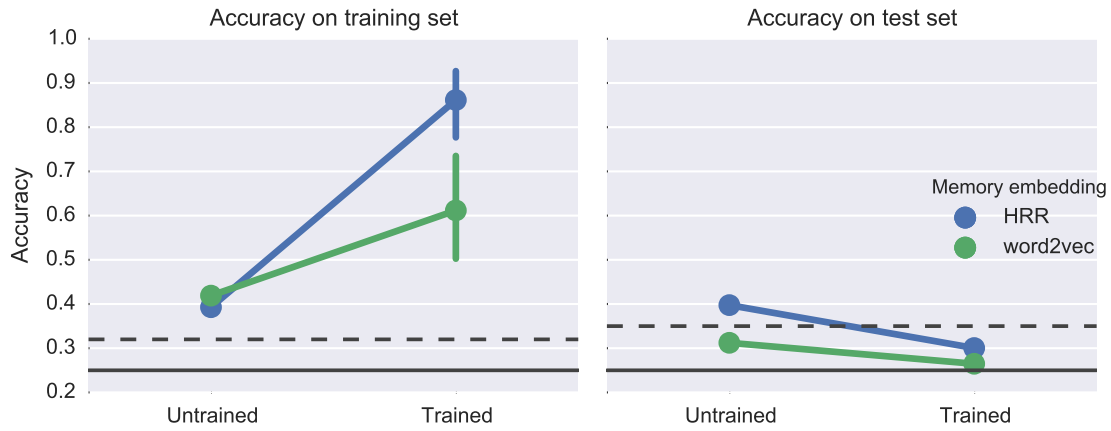
Next, we implemented an end-to-end memory network with each of the enhancements discussed in the previous section. We used SharcNet to run experiments on the MCTest160 data set in which different combinations of the enhancements are applied. For each trial in an experiment, we perform 20 epochs of training on every story in the training set using gradient descent via backpropogation. Each experiment consists of thirty trials. We report the results in Figure 5. As the final accuracy depends on the initial random model weights and random HRR vectors if used, we indicate 95% confidence intervals.

The results indicate that the coreference resolution leads to worse results. Role labeling might improve results slightly, but due to the size of the confidence intervals no reliable statement can be made. Time features do not lead to an improvement. HRRs give slightly better results than word2vec embedding. In neither case was a mean accuracy of 35% exceeded, which is the lower bound for a statistically significant result that differs from chance (cp. [1]).

Interestingly, using embeddings initialized with the HRR representations gives an accuracy of about 40% on the test set before training which is statistically significant with $p < 0.01$ (see 5b).



(a) Random initialization of learned embedding matrices



(b) HRR and word2vec initialization of learned embedding matrices

Figure 5: Results on the MCTest160 data set. Error bars denote 95% confidence intervals. The solid horizontal line denotes theoretical chance level (for infinite sample size), the dashed horizontal line denotes the required accuracy for reaching statistical significance in the difference to chance level given the actual sample size (obtained from [1]). (a) shows results for random initialization of the output and question embedding matrices. The left plot gives the accuracy on the test set when using different model features (baseline: no additional features, coref: using coreference resolution, roles: using role labeling). The right plot shows the accuracy when using time tags for different choices of the shift hyper-parameter s. Note that the x-axis is not scaled linearly. (b) shows the accuracy on the training and test set before and after training when initializing the output and question embedding matrices based on HRR or word2vec representations.

However, after training this accuracy is decreased while the accuracy on the training set increased. This indicates that the learning leads to overfitting.

From the results it appears that HRRs with role labeling and time tags with a shift parameter of 0.2 produce the best results (other than not training at all with preinitialized embeddings). We ran another 30 trials with this feature combination on the MCTest500 dataset and reached a mean accuracy of 30.6% (confidence interval from 29.4% to 31.2%). By using no training and preinitializing all of the embeddings with HRRs, we obtained 36.4% accuracy (confidence interval from 35.7% to 37.1%) on the same dataset, thereby exceeding the previously reported results in [2].

7 Discussion

Overall, we attempted to improve the transfer of learning from the training stories to the testing stories in our dataset in a variety of ways. None of our efforts resulted in a significant improvement. There are at least two possible explanations for this fact. First, even with our improvements, it could still be the case that the material in the training and test sets are too distinct to allow for successful learning using our methods. The fact that there a large number of words that are only seen in test set supports this hypothesis. Second, because the query embedding is used directly to predict a response as per (5), it is possible the model is simply learning to associate question embeddings and with the embeddings of the correct response. If there is enough variability amongst the choices and questions in the training set, the model could simply be "memorizing" which questions go with which answers while bypassing the selection of memory items. This is supported by the fact that the memory network is obviously overfitting the data. The best results could be obtained with no training at all and the generalization performance would decrease through the training. We hope to investigate these explanations and improve upon our current results in future work.

References

[1] Etienne Combrisson and Karim Jerbi. Exceeding chance level by chance: The caveat of theoretical chance levels in brain signal classification and statistical assessment of decoding accuracy. *Journal of neuroscience methods*, 2015.

[2] D. Kapashi and P. Shah. Answering reading comprehension using memory networks. Unpublished manuscript, Stanford University, 2015.

[3] T. Mikolov, I. Sutskever, L. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 1–9, 2013.

[4] T. Plate. *Holographic reduced representations*. CSLI Publications, Stanford, CA, 2003.

[5] M. Richardson, C. Burges, and E. Renshaw. Mctest: A challenge dataset for open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, 2013.

[6] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-to-end memory networks. In *Advances in neural information processing systems*, pages 1–11, 2015.

[7] J. Weston, A. Bordes, S. Chopra, T. Mikolov, and A. Rush. Towards ai-complete question answering. *ArXiv*, pages 1–11, 2015.

[8] J. Weston, S. Chopra, and A. Bordes. Memory networks. In *International Conference on Learning Representations*, pages 1–15, 2015.