End-to-End Memory Networks as Language Models: Investigation and Implementation

William W. Jack — MIT 2017
6.806 Final Paper
wjack@mit.edu

Abstract

We present a TensorFlow based open source implementation of the end-to-end memory network model for language modelling developed by Sukhbaatar et al. [2]. We investigate various properties of these models including the number of recurrent memory hops used and the optimization method used. We provide both quantitative and qualitative results from these models, in particular examples of sentences generated using them, something not demonstrated in the Sukhbaatar et al. paper. Finally, we examine our models' performance and propose interesting topics for research moving forward.

Code: https://github.mit.edu/wjack/MemNetLanguageModel

1 Background

1.1 Motivation

Memory networks have demonstrated to be powerful frameworks for NLP tasks such as question answering [1]. In their November 2015 paper, End-To-End Memory Networks, Sukhbaatar et al. (Sukhbaatar) demonstrated a framework for implementing memory networks trainable in an end-to-end manner [2]. Furthermore, they show this frameworks success in question answering as well as language modelling, demonstrating it to be more successful than traditional RNN language models.

Sukhbaatar's analysis and investigation of memory networks as language models leaves many questions unanswered regarding their behavior, and the reasons behind Sukhbaatar's choice of particular aspects of the framework remain unexplained. For instance, Sukhbaatar does not demonstrate sentences generate by the memory network models, and does not explain why they chose to use stochastic gradient descent (SGD) optimization rather than more modern methods such as AdaGrad or ADAM.To investigate these aspects of memory networks, as well as to provide the community with an open implementation of an end-to-end memory network, we decided to implement a memory network in Google's new machine learning framework, TensorFlow.

1.2 Memory networks as question answering models

Sukhbaatars end-to-end memory networks are originally presented in the context of a question answering system, and subsequently adapted to question answering. In the question answering model individual sentences, x_i, containing contextual information, are embedded into two memory banks, each sentence occupying a memory space m_i. This embedding is performed with an embedding tensor A and a learned temporal encoding vector $T_A(i)$, such that $m_i = \sum_j Ax_{ij} + T_A(i)$. Each memory bank has a unique embedding tensor and temporal encoding vector. The embedding matrices and positional biases for each memory bank are unique. The question is then embedded, and its similarity with past memories is evaluated by taking its inner product with each memory in one of

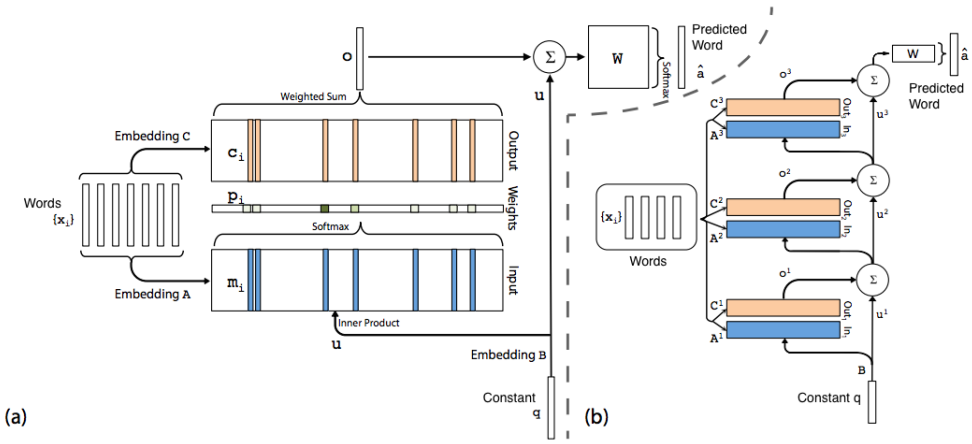

Figure 1: An end-to-end memory network as a language model. Graphics adapted from Sukhbaatar et al.

the banks. The resulting distribution is passed through a softmax, turning it into a probability distribution over the memories, and this distributions inner product is taken with the second memory bank. This second inner product represents each memory weighted by the relevancy of that memory to the question to be answered. These weighted memories are summed, producing an output vector in the embedding space. The output vector is summed with the embedded question, and passed through a tensor back into the original sentence space as an output representing the answer to the question. The authors found that this operation could be stacked and repeated, with the answer of one memory hop operation serving as the question for the next. They demonstrate great success with this model in question answering.

1.3 Memory networks as language models

We now turn our attention toward how to implement this model as a system for language modelling. Sukhbaatar accomplishes this by replacing each sentence with a single word, holding the question vector constant, and using RNN-like weight sharing in each memory hop. That is, the learned components are held constant through each recurrent memory hop. This framework is trainable as a language model by feeding N-1 words (represented as one hot vectors) from a corpus into memory, with the objective of predicting the next word. Sukhbaatar found that recurrently stacking the memory hop operation improves language model performance as well. It is important to note that without the temporal encoding parameters mentioned earlier, TA, this model will act as a bag-of-words model.

Figure 1 shows the memory network as a language model, using a graphic adapted from Sukhbaatar's paper.

2 Implementation

We investigate four implementations of such a model, a one-hop memory network, a three hop memory network trained using SGD, a three hop memory network using ADAM, and a six hop memory network using SGD, similar to the model presented in Sukhbaatars paper that achieved the best results.

2.1 Details

The models were implemented in TensorFlow and trained on Amazon EC2 g2.2xlarge instances. Significant troubles were encountered when training larger models on these instances. During training, the gradients and values of the learned parameters would repeatedly and erratically rise to NaN or infinity at various points in training. Assuming this to be an error in our implementation leading

to an exploding gradient, we spent approximately a week attempting to debug and eliminate this problem using techniques including gradient clipping, gradient normalization, lower learning rates, and different optimization methods – nothing seemed to work. As other TensorFlow users began to experience similar issues with larger models on g2.2xlarge instances, it became clear that there was a compatibility issue with these instances GPUs and TensorFlow. We modified the code to train the model on a CPU, and it succeeded immediately. Due to this CPU limitation, the training time of the model increased dramatically, and we were not able to train models nearly to the extent that Sukhbaatar was able to. Nonetheless, we believe that valuable insight into memory networks as language models was garnered from the work accomplished. All models were trained on the first 512kb of the Text8 dataset, and evaluated on the second 512kb. Text8 is a dataset consisting of the first 100MB of Wikipedias text. The combined vocabulary of the first two segments of text8 used is 22661 words. All models used gradient normalization to an $l2$ norm of 50. Meaning that the $l2$ norm of the gradient of all parameters was measured, and if larger than 50, the gradient was scaled down linearly to have an $l2$ norm 50.

2.2 One hop memory network

The first model we examine is a simple one hop memory network. We found that this model was capable of training far faster than the more complex models discussed later. The model was trained using Adagrad with an initial learning rate of .1.

After two epochs of training, this simple one hop model achieves a train perplexity of 369.54, and test perplexity of 1870.63 respectively. After one epoch of training, the model gave a train perplexity of 385.36 and a test perplexity of 1689.064. It seems that this model has fully converged and is no longer learning. Below are examples of words generated using this model given the context of 30 words. Throughout this paper, those sentences labelled as ML were generated by selecting the most likely word as the next word, and those sentences labelled as RC were generated by randomly choosing a word from the generated probability distribution over all words that the model produces.

ML:

- **Context:** ribbon worms the sipuncula and several phyla that have a fan of cilia around the mouth called a lophophore these were traditionally grouped together as the lophophorates but it now

  Prediction: being found in the united states and the most of the united states and the most
- **Context:** luanda s o paulo de loanda port railhead major cities amboim porto amboim bailundo vila teixeira da silva benguela s o felipe de benguella port railhead ca la vila robert

  Prediction: stadler is a is a is a is a is a is a is a is a is

RC:

- **Context:** the british empire hungarian philosophers hungarian novelists hungarian writers khazar studies writers who committed suicide the atlantic ocean is earth s second largest ocean covering approximately one fifth of its

  Prediction: compass of usage minarchists says that is controversial doublespeak much sink represented rationally also reviewers widely
- **Context:** make bricks for mortar when laying the bricks and for plaster on the interior and exterior walls some ancient cultures used concrete for the plaster to avoid rain damage it

  Prediction: is also in order british states molossians anatolia mvd notoc candidly culture states nations writings sessile

2.3 Three hop memory networks: ADAM and SGD

We examined two implementations of three hop memory networks that were identical in every way except for the optimization method used – ADAM for one, and SGD for the other. ADAM is a mod-

ern optimization method presented in [3] that adapts the learning rate based on an estimate of the quotient of the gradient vectors mean and variance evaluated for each parameter across steps. The memory networks we study include a variety of types of parameters to be optimized, and one could intuit that having parameter-specific learning rates could be highly advantageous for optimizing both temporal encoding parameters and word-embedding parameters, for instance. Additionally, the authors of [3] hold that ADAM works well for sparse gradients and is a method where hyperparameters typically require little tuning. As such, the author thought it would be interesting to compare ADAM to the technique presented in Sukhbaatar's paper, SGD.

Two memory networks were implemented, the only difference between the two being the optimization method. The first used a SGD optimizer with learning rate .01, and the second used an ADAM optimizer with learning rate of .01. The networks were trained for 2.5 epochs each. Both networks decreased in test perplexity from epoch 1 to epoch 2, so it is clear that they were continuing to learn and improve at this point in training.

2.4 Three hop memory network: SGD

The SGD optimizer achieved a train perplexity of 4797.85, and a test perplexity of 5776.39

ML:

- **Context:** zero mi a great rift valley also extends along the ridge over most of its length the depth of water over the ridge is less than two seven zero zero

  Prediction: in in in in in in in in in in in in in in in

RC:

- **Context:** six metres one two eight eight one ft the greatest depth eight six zero five metres two eight two three two ft is in the puerto rico trench the width

  Prediction: rivers that a zero blood ruling in swan occupations part cataclysmic according a albedo in ism

2.5 Three hop memory network: ADAM

The ADAM optimizer achieved a train perplexity of 7969.99, and a test perplexity of 8519.64

ML:

- **Context:** "premise asserts if and only if rather than if similarly the converse of a statement can be validly assumed to be true so long as the if and only if"

  Prediction: be such in the the three and the the three three three three three three three

RC:

- **Context:** six metres one two eight eight one ft the greatest depth eight six zero five metres two eight two three two ft is in the puerto rico trench the width

  Prediction: rivers that a zero blood ruling in swan occupations part cataclysmic according a albedo in ism

2.6 Six hop memory network

We now examine a six hop memory network optimized using stochastic gradient descent. This realization of a memory network is most similar to the one presented by Sukhbaatar et al. in their paper.

After training for two epochs, this model achieved a train perplexity of 8009.86, and a test perplexity of 10508.944. The model decreased n test perplexity from epoch 1 to 2, so it is evident that it was still continuing to learn.

ML:

- **Context:** ”s opposite other examples of modest proposals modest proposals and other literary hoaxes report from iron mountain sokal affair miscegenation origin of the word dihydrogen monoxide jack thompson attorney has”

  Prediction: a a a a a a a a a a a a a a a”

RC:

- **Context:** continent shaking wars he did indeed maintain his aloof position of minding not the times but the eternities schopenhauer on women schopenhauer is also famous for his essay on women

  Prediction: repeatedly mammary widow milligray ships bernazzani littoral reign saw buzkashi equalization moves caribou layer filament snipe

3 Discussion

3.1 Comparison of models implemented

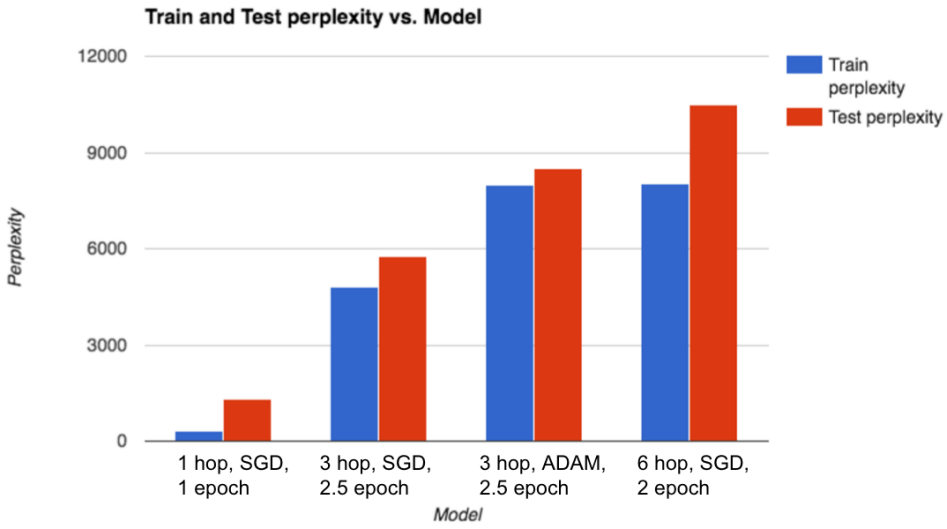

Figure 2: Train and test perplexity of the different models implemented

Due to the severe limitations arising from training these models on a CPU, only 2 epochs of training were possible for each model. Because all of the models are still in the process of learning rapidly and training has not converged to a constant train perplexity, it is difficult to draw absolute conclusions about which model performs the best. However, what we can tell is that smaller models do seem to converge faster, as evidenced by the one hop model fully converging after two epochs of training, whereas the three and six hop models are still learning after two epochs. Furthermore, we can see the efficacy of SGD optimization rather than ADAM through a comparison between the two three hop models. It seems that SGD trains the model far faster, achieving a test perplexity 32 % lower than the ADAM models after two batches. As neither model has converged, it is impossible to make conclusions about which optimization method leads to better post-convergence performance.

Additionally, one may note that the network optimized using SGD predicts the same word every time in the ML prediction, whereas the model optimized using ADAM predicts different, albeit common

words. This is evidence that the model optimized using ADAM has learned more in the way of temporal encoding parameters than the model optimized using SGD, which seems to be acting as something of a bag-of-words model.

3.2 Comparison with Sukhbaatar et al's results

It is difficult to compare our results with Sukhbaatars baselines, as their group trained each model for 50 epochs on a dataset 20 times larger than ours. Furthermore, they used 100 words as context to predict the next word, whereas we used 30 words. Additionally, they used a 500 dimensional embedding space, whereas we embedded into 300 dimensions. These compromises were made to speed up training. As expected, our models performed considerably worse than theirs. Sukhbaatars 6 hop model trained on text8 achieved a test perplexity of 147, far off from our 6 hop model, or our best performing model, the 1 hop memory network. (Figure 3)

Model	Author	Train performance	Test performance
RNN	Sukhbaatar	-	184
LSTM	Sukhbaatar	122	154
Memory net, 6 hops	Sukhbaatar	124	155
Memory net, 6 hops	Jack	8009.86	10508.94
Memory net, 3 hops, ADAM	Jack	7969.99	8519.64
Memory net, 3 hops, SGD	Jack	4797.85	5776.39
Memory net, 1 hop	Jack	369.54	1870.63

4 Conclusions

In conclusion, we have implemented a variety of end-to-end memory network models in TensorFlow and have gained valuable insight into how the choice of optimizer affects the training of a model, both quantitatively in terms of the perplexity it achieves, and qualitatively in the language it generates. We also give examples of sentences generated by end-to-end memory networks. Finally, we have provided to the community the first open source example of an end-to-end memory network written in TensorFlow. It is the hope of the author that this will accelerate research into this fascinating neural architecture.

Moving forward, the author aims to acquire a GPU that will enable further training of these models, and study the convergence behavior of models optimized with both ADAM and SGD. Additionally, the author would like to try feeding dense word vectors from word2vec into the model rather than one hot word vectors. The author hypothesizes this will decrease the size of the model, aid training, and eliminate one of the major tasks the current networks must perform during training, learning a dense representation of the words fed in.

5 Related work

5.1 Character level language model

We also implemented a character level language model using the end-to-end memory network framework. This network used 12 memory hops, gradient normalization to an $l2$ norm of 15, embedded character vectors into a 100 dimensional space, and used 400 characters as context for predicting the next. It was trained using SGD, with parameter updates applied after every training example. We found that this online learning method helped avoid local minima such as guessing a space for every character, while holding the probability distribution across all other characters constant. This model did not seem to learn anything beyond a bag of characters model, even when we tried optimizers such as ADAM. It always predicts the space character as the most likely next character, but did learn that certain letters such as e and a are more likely than other characters. The train perplexity it achieved was 21.40, the test perplexity it achieved was 18.10.

Acknowledgements

I would like to thank the staff of the 6.806/864 course, particularly Profs. Tommi Jaakkola and Regina Barzilay for fostering a phenomenal environment for learning and research in 6.806/864. I've yet to encounter an 'undergraduate' class at MIT that does this so well.

References

[1] Jason Weston, Sumit Chopra, Antoine Bordes: Memory Networks, 2014; [http://arxiv.org/abs/1410.3916 arXiv:1410.3916]

[2] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Rob Fergus: "End-To-End Memory Networks", 2015; [http://arxiv.org/pdf/1503.08895]

[3] Diederik Kingma, Jimmy Ba: "Adam: A Method for Stochastic Optimization", 2015; [http://arxiv.org/abs/1412.6980]