the morning paper

a random walk through Computer Science research, by Adrian Colyer

Made delightfully fast by

MENU

Memory Networks

MARCH 10, 2016 ~ ADRIAN COLYER

Memory Networks Weston et al. 2015

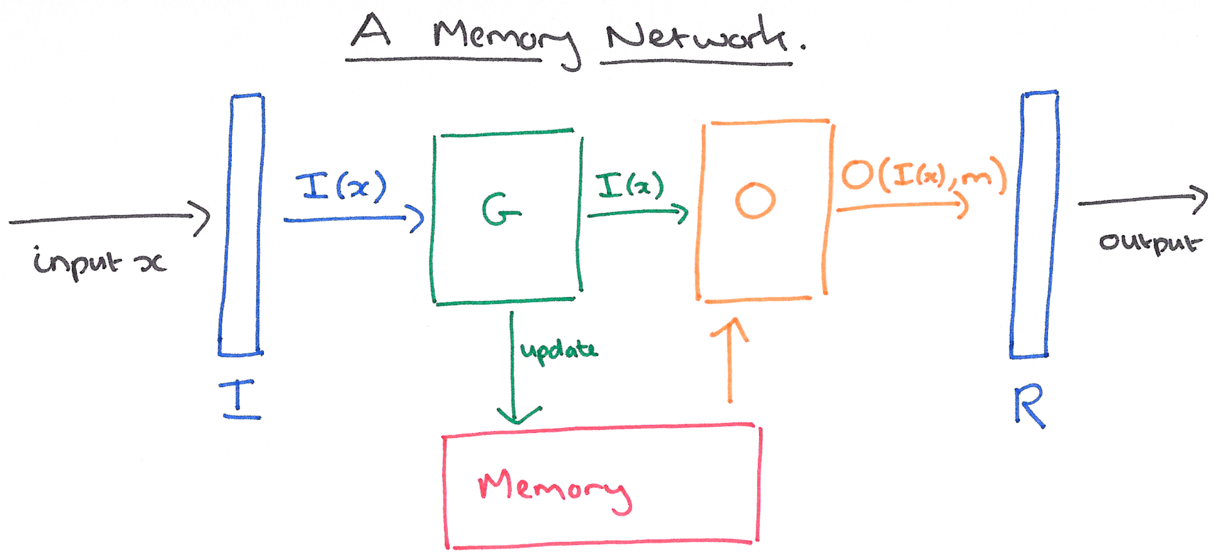
As with the Neural Turing Machine that we look at yesterday, this paper looks at extending machine learning models with a memory component. The Neural Turing Machine work was developed at Google by the DeepMind team, today's paper on Memory Networks was developed by the Facebook AI Research group. The two bodies of work appear to have been developed independently – an idea whose time has come perhaps?

> Our work was submitted to arxiv just before the Neural Turing Machine work of Graves et al. which is one of the most relevant related methods. Their method also proposes to perform (sequence) prediction using a "large, addressable memory" which can be read and written to. In their experiments, the memory size was limited to 128 locations, whereas we consider much larger storage (up to 14M sentences). The experimental setups are notably quite different also: whereas we focus on language and reasoning tasks, their paper focuses on problems of sorting, copying and recall. On the one hand their problems require considerably more complex models than the memory network described in Section 3. On the other hand, their problems have known algorithmic solutions, whereas(non-toy) language problems do not.

A memory network combines learning strategies from the machine learning literature with a memory component that can be read and written to. The model is trained to learn how to operate effectively with the memory component. The high-level view of a memory network is as follows:

- There is a memory, **m**, an indexed array of objects (e.g. vectors or arrays of strings).
- An input feature map **I**, which converts the incoming input to the internal feature representation
- A *generalization* component **G** which updates old memories given the new input. "We call this generalization as there is an opportunity for the network to compress and generalize its memories at this stage for some intended future use."
- An output feature map **O**, which produces a new output in the feature representation space given the new input *and the current memory state*.
- A *response* component **R** which converts the output into the response format desired – for example, a textual response or an action.

I,G,O and R can all potentially be learned components and make use of any ideas from the existing machine learning literature. They are connected as follows:



In question answering systems for example, the components may be instantiated as follows:

- **I** can make use of standard pre-processing such as parsing, coreference, and entity resolution. It could also encode the input into an internal feature representation by converting from text to a sparse or dense feature vector.
- The simplest form of **G** is to introduce a function *H* which maps the internal feature representation produced by I to an individual memory slot, and just updates the memory at *H(I(x))*.

> More sophisticated variants of G could go back and update earlier stored memories (potentially, all memories) based on the new evidence from the current input *x*. If the input is at the character or word level one could group

> inputs (i.e., by segmenting them into chunks) and store each chunk in a memory slot.

For a 'huge' memory (e.g. all of Wikipedia) the function *H* could be designed or trained to store memories by entity or topic. If memory becomes too full, a 'forgetting' function could also be implemented, but the authors had not yet explored this at the time the paper was written.

- **O** Reads from memory and performs inference to deduce the set of relevant memories needed to perform a good response.
- **R** would produce the actual wording of the question answer based on the memories found by O. For example, R could be an RNN conditioned on the output of O.

When the components I,G,O, & R are *neural networks*, the authors describe the resulting system as a Memory Neural Network (MemNN). They build a MemNN for QA (question answering) problems and compare it to RNNs (Recurrent Neural Network) and LSTMs (Long Short Term Memory RNNs) and find that it gives superior performance.

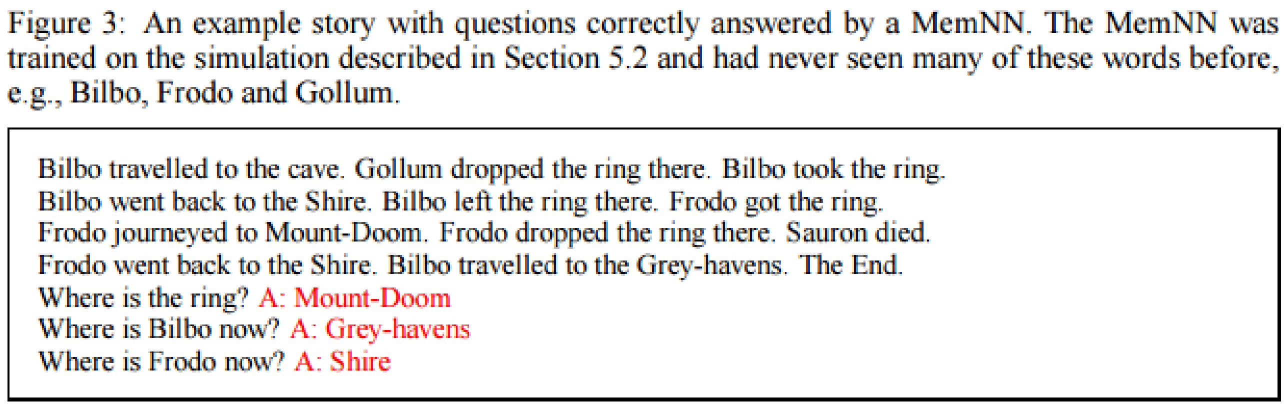


Figure 3: An example story with questions correctly answered by a MemNN. The MemNN was trained on the simulation described in Section 5.2 and had never seen many of these words before, e.g., Bilbo, Frodo and Gollum.

Bilbo travelled to the cave. Gollum dropped the ring there. Bilbo took the ring.
Bilbo went back to the Shire. Bilbo left the ring there. Frodo got the ring.
Frodo journeyed to Mount-Doom. Frodo dropped the ring there. Sauron died.
Frodo went back to the Shire. Bilbo travelled to the Grey-havens. The End.
Where is the ring? A: Mount-Doom
Where is Bilbo now? A: Grey-havens
Where is Frodo now? A: Shire

The most basic version of their MemNN works as follows. **I** is given a sentence at a time, and **G** simply stores the sentence in the next available memory slot (i.e., we assume there are more memory slots than sentences). All of the hard work is done in the **O** and **R** components. **O** is used to find up to *k* supporting memories. The first memory is found by scoring the match between the input sentence and each sentence in memory, and returning the one with the highest score. For $k > 1$, the *n*_th *memory ($1 < n \leq k$) is found by comparing both the original input sentence and all of the n-1 supporting memories found so far against the sentences in memory. The sentence with the highest matching score is returned. The output of **O** is a list of sentences containing the original input sentence and the _k* supporting sentences it has found . The simplest version of **R** can simply return the first supporting sentence, a more sophisticated version might use an RNN to perform sentence generation. A compromise is to limit textual responses to be a single word using a scoring function over all of the words in the dictionary and the sentences produced by **O**.

When the input to **I** is a stream of words, not already segmented as statements and questions, then a (to be learned) segmentation function is added. When it indicates it has found a segment, that sequence is written to memory and the process proceeds as above. With a lot of memories, scoring each one every time in **O** becomes prohibitively expensive. To address this the authors consider two schemes to reduce the number of sentences to be compared by putting sentences into buckets and only comparing an input sentence to memories in the same buckets: (i) hashing words and putting a sentence into all buckets corresponding to its words, and (ii) clustered word embeddings using K-means to cluster word vectors.

> We can extend our model to take into account when a memory slot was written to. This is not important when answering questions about fixed facts ("What is the capital of France?") but is important when answering questions about a story…

The authors had most success learning a model with an output scoring function based on triples (x,y,y') and three new feature variables that can take on the values 0 and 1, and represent whether x is older than y, x is older than y', and y is older than y' respectively.

> Even for humans who have read a lot of text, new words are continuously introduced. For example, the first time the word "Boromir" appears in Lord of The Rings (Tolkien, 1954). How should a machine learning model deal with this? Ideally it should work having seen only one example… for each word we see, we store a bag of words it has co-occurred with, one bag for the left context, and one for the right. Any unknown word can be represented with such features. Hence, we increase our feature representation D from 3|W| to 5|W| to model these contexts (|W| features for each bag). Our model learns to deal with new words during training using a kind of "dropout" technique: d% of the time we pretend we have not seen a word before, and hence do not have a n-dimensional embedding for that word, and represent it with the context instead.

POSTED IN UNCATEGORIZED

DEEP LEARNING FACEBOOK

< PREVIOUS
Neural Turing Machines

NEXT >
Strategic Dialogue Management via Deep Reinforcement Learning