2016-10-10
TF Study

Memory Networks

J. Weston, S. Chopra, A. Bordes.

ICLR 2015 (and arXiv:1410.3916).

CONTENTS

[1]Memory Network

General Framework

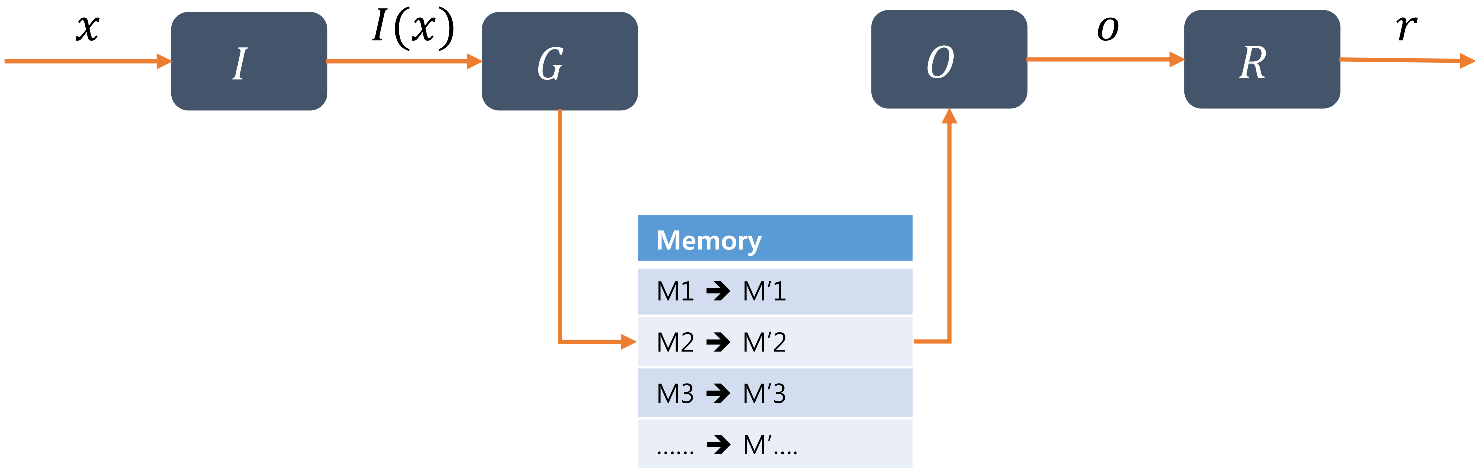
- Class of models that combine large memory with learning component that can read and write to it

- Consists of a memory $\boldsymbol{m}$($an\ array\ of\ objects\ indexed\ by\ \boldsymbol{m_i}$), and four components $\boldsymbol{I}, \boldsymbol{G}, \boldsymbol{O}$ and $\boldsymbol{R}$
  - **I:** (input feature map) convert incoming input to the internal feature representation.
  - **G:** (generalization) update memories given new input.
  - **O:** produce new output (in feature representation space) given the memories.
  - **R:** (response) convert output O into a response seen by the outside world.

- Flow of the model:
  1. Convert $\boldsymbol{x}$ to an internal feature representation $\boldsymbol{I}(\boldsymbol{x})$.
  2. Update memories $\boldsymbol{m_i}$ given the new input: $\boldsymbol{m_i} = \boldsymbol{G}(\boldsymbol{m_i}, \boldsymbol{I}(\boldsymbol{x}), \boldsymbol{m}), \forall i$.
  3. Compute output features $\boldsymbol{o}$ given the new input and the memory: $\boldsymbol{o} = \boldsymbol{O}(\boldsymbol{I}(\boldsymbol{x}), \boldsymbol{m})$.
  4. Finally, decode output features o to give the final response: $\boldsymbol{r} = \boldsymbol{R}(\boldsymbol{o})$

1) J. Weston, S. Chopra, A. Bordes. *Memory Networks.* ICLR 2015 (and arXiv:1410.3916).

- Flow of the model:
  1. Convert $\boldsymbol{x}$ to an internal feature representation $\boldsymbol{I}(\boldsymbol{x})$.
  2. Update memories $\boldsymbol{m_i}$ given the new input: $\boldsymbol{m_i} = \boldsymbol{G}(\boldsymbol{m_i}, \boldsymbol{I}(\boldsymbol{x}), \boldsymbol{m}), \forall i$.
  3. Compute output features $\boldsymbol{o}$ given the new input and the memory: $\boldsymbol{o} = \boldsymbol{O}(\boldsymbol{I}(\boldsymbol{x}), \boldsymbol{m})$.
  4. Finally, decode output features o to give the final response: $\boldsymbol{r} = \boldsymbol{R}(\boldsymbol{o})$



01 Memory Network
Component Description

- $\boldsymbol{I}$ component:
  - Pre-processing step : convert to feature vector
- $\boldsymbol{G}$ component:
  - Simplest form is just store $\boldsymbol{I}(\boldsymbol{x})$ in a slot in the memory , $\boldsymbol{m}_{\boldsymbol{H}(\boldsymbol{x})} = \boldsymbol{I}(\boldsymbol{x})$, $\boldsymbol{H}$: $\boldsymbol{slot\ choosing\ function}$, other earlier memory remain untouched
  - Sophisticated variants of G could go back and update earlier stored memories. If memory becomes full, 'forgetting' procedure could be implemented by H to choose which memory is replaced.
- $\boldsymbol{O}$ & $\boldsymbol{R}$ component:
  - $\boldsymbol{O}$: responsible for reading from memory and perform inference.
  - $\boldsymbol{R}$: produces the final response given O, could be RNN that is conditioned on the output of O , etc.

- Basic model
  - Assumption :
    - statement of a fact & questions to be answered = Text
    - Old memories are not updated, ***G*** only stored new information in next available memory slot
  - ***O*** component
    - Produces output features by finding k supporting memories given x
    - For $k = 2$, supporting memories are as follows:

$$o_1 = O_1(x, m) = \underset{i=1,\ldots,N}{argmax}\, s_o(x, m_i) \quad : s_o\,(function\ that\ scores\ the\ match\ between\ sentence\ x\ and\ m_i)$$

$$o_2 = O_2(x, m) = \underset{i=1,\ldots,N}{argmax}\, s_o([x, m_{o_1}], m_i) \quad : score\ function\ considering\ 1st\ order\ memory$$

  - ***R*** component
    - Produces textual response r , simplest model is just return m_{o_k}
    - Can use RNN for generating text, in this paper limit for single word

$$r = \underset{w \in W}{argmax}\, s_R([x, m_{o_1}, \ldots m_{o_k}], w) \quad : W(\ set\ of\ all\ words\ in\ the\ dictionary)$$

02 Implementation for Text

Basic Model Scoring Function

- Scoring function:
  - $\boldsymbol{s(x,y) = \Phi_x(x)^T U^T U \Phi_y(y)}$
  - $\boldsymbol{U}$: $\mathrm{n} \times D\ matrix$, $\boldsymbol{D}$: $number\ of\ features$, $\boldsymbol{n}$: $embedding\ dimension$
  - $\boldsymbol{\Phi}$: mapping text to D-dimensional feature space (BOW model)
  - $\boldsymbol{O}$ component uses $\boldsymbol{U_O}$, $\boldsymbol{R}$ component uses $\boldsymbol{U_R}$
  - Use $\boldsymbol{D = 3|W|}$ for experiment
- Example:
  - $\boldsymbol{x = \text{"}Where\ is\ the\ milk\ now\text{"}}$
  - $m_{o_1} = \text{"}Joe\ left\ the\ milk\text{"}$, $m_{o_2} = \text{"}Joe\ travelled\ to\ the\ office\text{"}$
  - $\boldsymbol{r = \text{"}office\text{"}}$

Figure 1: Example "story" statements, questions and answers generated by a simple simulation. Answering the question about the location of the milk requires comprehension of the actions "picked up" and "left". The questions also require comprehension of the time elements of the story, e.g., to answer "where was Joe before the office?".

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.
Joe travelled to the office. Joe left the milk. Joe went to the bathroom.
Where is the milk now? A: office
Where is Joe? A: bathroom
Where was Joe before the office? A: kitchen

- Training
  - Fully supervised setting:
    - Inputs and responses, and the supporting sentences are labeled.
    - During training we know the best choice of max functions
    - Margin ranking loss and stochastic gradient descent
    - For a given question $\boldsymbol{x}$ with true response $\boldsymbol{r}$ and supporting sentences $\boldsymbol{m_{o_1}}$ and $\boldsymbol{m_{o_2}}$, minimize over model parameter $\boldsymbol{U_O}$, $\boldsymbol{U_R}$
    - $\bar{\boldsymbol{f}}, \bar{\boldsymbol{f}}'$ *and* $\bar{\boldsymbol{r}}$ are other choices than correct labels, $\boldsymbol{\gamma}$ *is margin* , for every step of SGD sample those rather than compute whole sum

$$\sum_{\bar{f} \neq \mathbf{m}_{o_1}} \max(0, \gamma - s_O(x, \mathbf{m}_{o_1}) + s_O(x, \bar{f})) +$$

$$\sum_{\bar{f}' \neq \mathbf{m}_{o_2}} \max(0, \gamma - s_O([x, \mathbf{m}_{o_1}], \mathbf{m}_{o_2}]) + s_O([x, \mathbf{m}_{o_1}], \bar{f}'])) +$$

$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], r) + s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], \bar{r}]))$$

02 Implementation for Text

More sophisticated setup

- Input is not a sentence but sequence of word:
  - Words are not already segmented as statements and questions
  - Add Segmentation function

    $\text{seg(c)} = W_{seg}^{T} U_S \, \Phi_{seg}(c)$ $\left(W_{seg}: vector\ the\ parameters\ of\ linear\ classifier\ in\ embedding\ space \right)$

    c : sequence of input words represented as bag of words
  - If $seg(c) > \ \gamma\ (margin)$ ➔ sequence **c** is recognized as a segment

- Efficient memory via hashing:
  - If set of stored memories is very large ➔ spend too much time to score all of memories
  - Hashing word ➔ for given sentence hash it into all the word buckets corresponding to its word , input and hashed bucket need to share a word
  - Clustering word embeddings ➔ after training embedding matrix $\boldsymbol{U_O}$, run K-means to cluster word and give K bucket.

- Considering write time:
  - Take into account when a memory slot was written to. Different to time information described in the text of statements.
  - Learn a function on triples

$$s_{O_t}(x, y, y') = \Phi_x(x)^T U_{O_t}{}^T U_{O_t}(\Phi_y(y) - \Phi_y(y') + \Phi_t(x, y, y'))$$

  - $\Phi_t(x, y, y')$: three new feature which take on the value 0 or 1
  - $S_O(x, y, y') > 0$, model prefers y over y' , and if $S_O(x, y, y') < 0$, it prefes y'

Algorithm 1 O_t replacement to arg max when using write time features

```
function O_t(q, m)
    t ← 1
    for i = 2, ..., N do
        if s_{O_t}(q, m_i, m_t) > 0 then
            t ← i
        end if
    end for
    return t
end function
```

02 Implementation for Text

More sophisticated setup

- Previously unseen words:
  - Use language modeling ➔ given a neighboring words , predict what the word should be, and assume the new word is similar to that
  - Use bag of word approach, left context & right context are mapped to new features
  - 3|W| ➔ 5|W| dimension

- Exact matches and unseen words:
  - Score a pair of x,y in another way : $\boldsymbol{\Phi_x(x)^T U^T U \Phi_y(y)} + \boldsymbol{\lambda \Phi_x(x)^T \Phi_y(y)}$
  - Extend the feature representation D with matching features, one per word. ➔ D = 8|W|
  - Matching feature indicates if a word occurs in both x and y

- Dataset:
  - QA Dataset introduced in [2]Fader et al (2013)
  - 14M Statements, stored as (subject, relation, object) ex:) (milne, authored, winnie-the-pooh)

- Model:
  - 128 dimension for embedding, no fine tuning
  - k = 1 supporting memory
  - Exact matching feature is used
  - Time, unseen word modeling were not used.

- Time consumption problems:
  - Look up is linear in the size of the memory (14M) ➔ slow
  - Use two method for faster calculation.

- Evaluation:
  - re-ranking the top returned candidate answers & measuring F1 score over the test set. Labels are annotated by human.

2) Fader, Anthony, Zettlemoyer, Luke, and Etzioni, Oren. Paraphrase-driven learning for open question answering. In *ACL*,pp1608-1618,2013.

- Result:

Table 1: Results on the large-scale QA task of (Fader et al., 2013).

Method	F1
(Fader et al., 2013)	0.54
(Bordes et al., 2014b)	0.73
MemNN (embedding only)	0.72
MemNN (with BoW features)	0.82

Table 2: Memory hashing results on the large-scale QA task of (Fader et al., 2013).

Method	Embedding F1	Embedding + BoW F1	Candidates (speedup)
MemNN (no hashing)	0.72	0.82	14M (0x)
MemNN (word hash)	0.63	0.68	13k (1000x)
MemNN (cluster hash)	0.71	0.80	177k (80x)

- Simulated World QA:
  - Simple simulation of 4 characters, 3 objects and 5 rooms
  - Characters moving around, picking up and dropping objects.
  - QA for Simple stories.

- Dataset:
  - 7k statements and 3k questions
  - Statements are joined together again with a simple grammar

- Complexity control:
  - Setting a limit on the number of time steps in the past the entity was last mentioned.

- Generating text for simulation

- Tasks within the simulation are restricted to question answering tasks about the location of people and object
  - Ex) “What did John just do?” , “Where is John now?”

- Built simple automated grammar:
  - To produce more natural looking text
  - Each verb is assigned a set of synonyms.
  - Each object and actor can have a set of replacement synonyms.

- Executing Actions and Asking Questions:
  - For simple story, build model to answer questions

Statements) *Joe go kitchen; Fred go kitchen; Joe get milk; Joe go office; Joe drop milk; Joe go bathroom*
Q) Where milk ?
Where Joe?
Where Joe before office?

- Result:

Table 3: Test accuracy on the simulation QA task.

	Difficulty 1			Difficulty 5	
Method	actor w/o before	actor	actor+object	actor	actor+object
RNN	100%	60.9%	27.9%	23.8%	17.8%
LSTM	100%	64.8%	49.1%	35.2%	29.0%
MemNN $k = 1$	97.8%	31.0%	24.0%	21.9%	18.5%
MemNN $k = 1$ (+time)	99.9%	60.2%	42.5%	60.8%	44.4%
MemNN $k = 2$ (+time)	100%	100%	100%	100%	99.9%

- MemNN on simulated dataset for test previously unseen words:
  - For 'Lord of the Ring' context, test MemNN's performance for previously unseen words.
  - QA for Simple stories.

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

  - MemNNs can discover simple linguistic pattern.
  - Successfully generalize the meaning of their instantiations using unknown words to perform 2-stage inference.

Video : https://www.facebook.com/Engineering/videos/10153098860532200/

03 Experiments

Combining simulated data and large-scale QA

- Simulated QA learns how to infer stories, and large-scale QA learns real-world general information:
- Make an ensemble model of the two.

Figure 4: An example dialogue with a MemNN system trained on both the simulation data and the large-scale QA data. The system is able to (attempt to) answer questions about general world knowledge and about specific story-based statements in the dialogue.

Fred went to the kitchen. Fred picked up the milk. Fred travelled to the office.
Where is the milk ? A: office
Where does milk come from ? A: milk come from cow
What is a cow a type of ? A: cow be female of cattle
Where are cattle found ? A: cattle farm become widespread in brazil
What does milk taste like ? A: milk taste like milk
What does milk go well with ? A: milk go with coffee
Where was Fred before the office ? A: kitchen

04 Future work

Conclusion

- Develop MemNNs for text further, evaluating them on harder QA problem.:
- More complex simulation data could also be constructed in order to bridge that gap.
  - Requiring coreference, involving more verbs and nouns, sentences with more structure and requiring more temporal and casual understanding.
- More sophisticated architecture should also be explored to deal with complex tasks.
  - Using more sophisticated memory management via **G**
  - Sophisticated sentence representation (this paper used bag of word models)
- Current model needs to be fully supervised. Weakly supervised settings should be adapted.

05 Reference

Related papers

- Jason Weston. *Memory Networks for Language Understanding*, ICML Tutorial 2016
- J. Weston, S. Chopra, A. Bordes. *Memory Networks.* ICLR 2015 (and arXiv:1410.3916).
- Weston, Jason, Bengio, Samy, and Usunier, Nicolas. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pp. 2764–2770. AAAI Press, 2011.

Presentation

Thanks for Watching