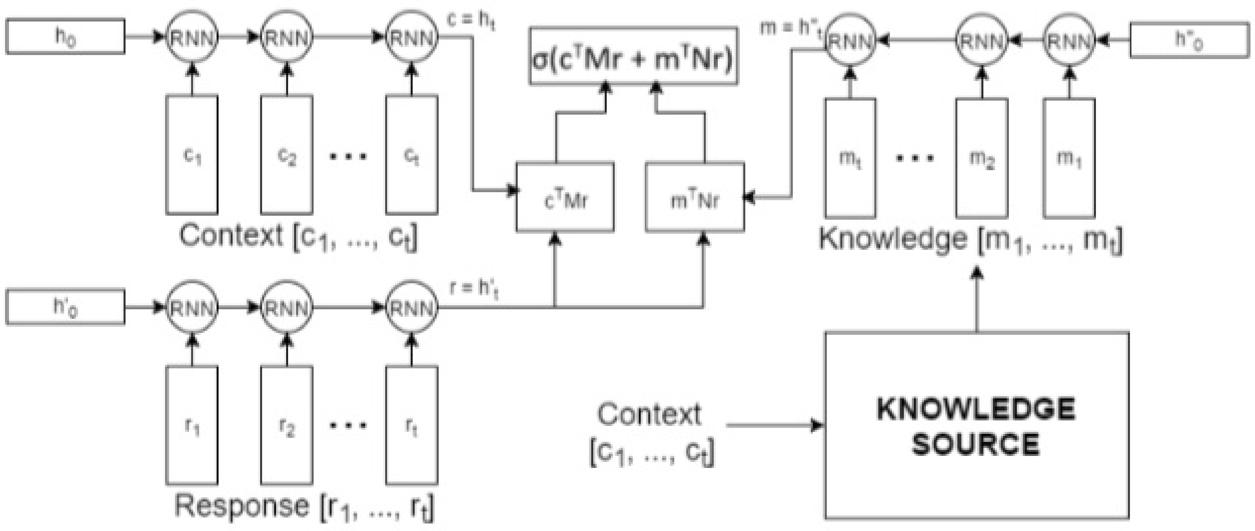
Implementation of Incorporating Unstructured Textual Knowledge



Dependencies:

- Python 2.7
- Tensorflow (only for word2vec)
- numpy
- nltk
- pandas
- Lasagne
- cPickle

Launch script from script folder/

1. Transform ubuntu-corpus / manpages words into embeddings

- Download training dataset ubuntu-ranking-dataset-creator
- Save it as output.csv in the data folder
- Make sure man_tokenized_sentences.csv is in the data folder
- Run `python script/create_ubuntu_dataset.py`

2. Generate datasets for the model

- Run `python/gen_data.py`

Except *dataset.p*, all files in *data/* can be deleted after this step. *dataset.p* contains all training and testing examples (including the knowledge)

3. Test the model

- Run `python main.py --data ../data/dataset.p --training_random [yf] --pre_train_DE [yf] --untied [yf]`
  - When **training_random** is set to true, the model is trained on randomly generated data. **Allow checking the model implementation rapidly**

    - When **pre_train_DE** is set to true, no external knowledge is used (N=0)
    - When **untied** is set to true, the external knowledge encoder is composed of a separate LSTM layer, not bidirectional, with a final dense layer.

Side notes concerning trueAi assignement:

Transform ubuntu-corpus / manpages words into embeddings

- I used a small part of ubuntu-dataset for the training of the neural network (30.000/1.000.000 elements). I just wanted to assess my model on some data (other than random) and not on all the ubuntu corpus.
  I also create a small dataset containing ~10 manPages as in the form described in the paper.
  Every words in the ubuntu corpus (size_of_vocabulary=10000 most common words) are transformed into embeddings (size_embeddings=28) with word2vec.
  I used tensorflow for word2vec.

Generate dataset

- Afterward, I used the 30.000 elements and split them into 24000 training examples and 6000 testing examples.
  I did not use any validation set.
  I implement the database containing the two hash tables (entity

and relation), and the algorithm to retrieve the best document. However because my manpages database contained only a few elements, it is of no relevance. Best document is retrieved randomly (see code in hash_tables.py l.155).
If only I had more time with my school/research projects, I would have used a bigger man page database. Nevermind, the code is here, I just need to implement a script for retrieving man pages and parse it.

Model

- I implemented almost all specificities mentioned in the paper. I just don’t use recall@k to check the correctness of the neural network.
  I used LSTM cells instead of classical RNN.
  I implemented a random batch generator, so as to make it easier to test the model, without executing the first two scripts.
  It is possible to
    - train the neural network on only context-response (in the test, on labeled data, I get some improvement of the loss error)
    - create two neural networks (one for the context response encoder, and one for the knowledge encoder)
    - learn weight for the tied neural network or both untied neural networks.
      The code is quite long, but there are some comments.
      No regularization, no batch normalization.
      There are others parameters for the script that can be

found in the source code (main.py l.503)

Further improvement

- retrieve man pages
- train with larger training set
- add regularization, dropout
- train with other knowledge database…

Conclusion

- It was a nice and interesting project. The paper’s idea is really worth it. I have the feeling that incorporating external knowledge is a key to improve dialogue systems. However the paper’s method is very simple, and I hope there will be work in that direction to improve the incorporation of external prior knowledge (Actually, I found some more advanced works on this idea, such as this one http://videolectures.net/deeplearning2016_wen_network_based/)

  I hope I was able to show you my strong motivation to work with the trueAI team, to solve next challenge in dialogue systems.

  I also think to get back to the project once the school semester will be over and improve the dataset.
  Feel free to contact me back to talk about the implementation if you have any questions.