QUERY BASED INFORMATION EXTRACTION SYSTEM USING DEEP LEARNING

Rakesh Bhimappa Mali

ITC Infotech India Limited Organization

Abstract

Deep learning has emerged as a new area of machine learning research. It tries to mimic the human brain, which is capable of processing and learning from the complex input data and solving different kinds of complicated tasks. It has been successfully applied to several fields such as images, sounds, text and motion.

The techniques developed from deep learning research have already been impacting the research of natural language processing. Automatically processing natural language inputs and producing language outputs is a key component of Artificial General Intelligence. This implementation focuses on extraction of information from the trained model based on the intent of input query.

Requirements

➢ Torch Deep Learning framework

➢ Python 2.7

Modules:

- numpy
- h5py
- argparse
- sys
- re
- codecs
- copy
- operator
- json
- glob, os
- csv
- re

➢ Luarocks/Lua 5.2

Modules:

- nn
- optim
- cunn
- cutorch
- hdf5
- rnn
- nngraph
- csv
- pretty-nn

Implementation

Implementation of the proposed model involves three distinct stages:

1. Preprocessing phase:

Processing of data corpus and translation of textual data into model understandable tensors

2. Training phase:

Structuring of model and training the same with preprocessed data.

3. Interactive phase:

Loading of trained model for establishing an interactive session with end user.

Implementation

1. Data Preprocessing (Python)

➢ Command-line options for data preprocessing:

- -vocabsize : indicates the maximum size of vocabulary.
- -dir : specifies the directory path of data corpus

➢ Preprocessing steps:

1. Segmentation of words in the data corpus and assigning an unique identifier to each token.

2. Saving the word to index dictionary as a CSV file.

3. Processing each data file to find its attributes. Attributes include the following:
   - the count of lines in the file
   - the length of the longest line in the file

4. Processing the content of each file and segregating data into the following tensors:
   - all_stories - tensor containing all stories of the file.
   - all_questions - tensor containing all questions present in the file.
   - all_answers - tensor containing answers corresponding to questions present in the file.
   - all_linenos - tensor involving all line numbers of the file.

5. Saving the preprocessed data tensors into different hdf5 files based on the task number.

   Example:

   - qa01.hdf5:
     - ✓ train data
       - all_stories
       - all_questions
       - all_linenos
       - all_answers
     - ✓ test data
       - all_stories
       - all_questions
       - all_linenos
       - all_answers

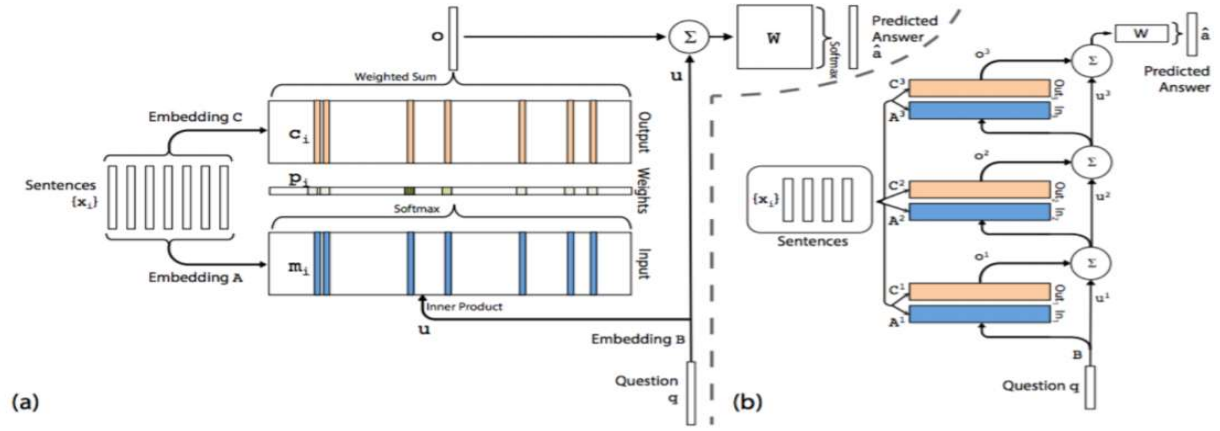
Implementation

2. Model Training (Lua)

➢ Command-line options:

- -datafile: specifies the preprocessed data file for model training.
- -eta: learning rate of the model.
- -max_grad_norm: maximum normalization for RNN models.
- -grad_norm: specifies the type of gradient renormalization.
- -N: specifies the number of epochs for training.
- -D0: number of outputs for lookup layer of nn.
- -debug: specifies the model for debugging.
- -unit_test: used for testing the output of each intermediate layer.
- -k: number of hops for MemN2N model.
- -max_history: specifies the number of previous lines to be considered as "context".
- -max_line_len: specifies the maximum line length of data for training.
- -pe: enables positional encoding.
- -te: enables temporal encoding.
- -save: a boolean option used to save the trained model.
- -saveminacc: minimum accuracy for saving a model.
- -cuda: a boolean option for enabling/disabling CUDA.

➢ Network Architecture:



Sentence selection: The network uses a memory of fixed size ‘m’, which is measured in terms of the number of encoded sentences. We zero-pad stories shorter than m sentences. For cases where the story is longer than ‘m’ sentences, the last ‘m’ sentences from the story are stored in the memory and earlier memories are simply discarded.

Input representation: Once the story is converted to the memory size using padding, the input to the model is a set of sentences x_1, $x_2 \cdot \cdot \cdot x_m$ and a question. To encode the story, we use an embedding look up matrix ‘A’. The encoded query vector is created from the query using a separate embedding matrix ‘B’ and position encoding. Using the story and query representations, we compute the importance of each memory slot by taking a softmax over the dot product of query with each memory slot: p_i = softmax(u^T m_i).

Output representation: In addition to input vector above, we also generate an output vector ‘c_i’ for each sentence by using an embedding matrix ‘C’ and a temporal encoding matrix. The final output ‘o’ from the memory module is then computed as $o = \sum_i p_i c_i$.

Multiple hops: When using multiple hops, we stack the memory layers as shown in part (b) of the above figure. The final output of the network is then used to generate the final prediction: after ‘k’ hops, we have $a = \text{softmax}(W(u^{k+1})) = \text{softmax}(W(\text{ReLU}(H\, o^k + u^k)))$, , where $H \in R^{dxd}$ is a linear mapping.

Implementation

3. Interactive phase (Lua)

➢ Command-line options:

- -modelfile: specifies the path of saved model.
- -datafile: specifies the preprocessed data file to be loaded.
- -D0: number of outputs for lookup layer of nn.
- -max_history: specifies the number of previous lines to be considered as “context”.
- -max_line_len: specifies the maximum line length of data for training.
- -cuda: a boolean option for enabling/disabling CUDA.

➢ Steps:

- Load the processed data corpus.
- Load the saved binary model file.
- Input user query.
- Convert query to word to index vector.
- Invoke forward pass as in training phase.

Results

```
naveen@blrpragws14819:~/Documents/MemN2N$ th story.lua -modelfile new_qa20.hdf5.99.9.memnn

Welcome to ITC Policies Search Engine

How may I help you?

reimbursement for uk?
reimbursement per day in uk is gbp 45


How may I help you?

reimbursement for denmark?
reimbursement per day in denmark is euro 100


How may I help you?

please provide information about travel policies for family.
travel of spouse and two dependent children up to the age of 21 years will be allowed under five conditions
if the project is over six months and if the concerned manager is on a valid work permit
if the project of less than six months duration has extended over six months
the passage for family travel will be paid by the company
no separate accommodation for family is assured
individual room in guest house or hotel could be shared by the family
responsibility of obtaining the visa is that of family or the individual
company will reimburse the cost of a suitable medical insurance policy for the family
for deputations extending six months individual is eligible to only one to and fro fare for the family


How may I help you?

what is long term deputation?
long term deputation is for business travel having stay greater than six months but less than 18 months


How may I help you?

maximum reimbursement in gbp?
reimbursement is subject to a maximum of usd 250 euro 191 gbp 162


How may I help you?

Types of foreign travel?
type 1 is non-billable business travel
type 2 is billable business travel
type 3 is deputation
type 4 is long term deputation
type 5 is secondment


How may I help you?

Thank you

Thank you for using the portal!
```

Analysis

➢ Model evaluation in four different categories:
1) Underfitting – Validation and training error high and almost equal
2) Overfitting – Validation error is high, training error low
3) Good fit – Validation error low, slightly higher than the training error
4) Unknown fit - Validation error low, training error 'high'

➢ Monitoring Validation Loss vs. Training Loss:
- If overfitting, decrease network size, or to increase dropout
- If underfitting, increase the size of model.

➢ Ensure the number of parameters is close to the input dataset size in order to avoid overfitting and underfitting.

Conclusion

The implementation involves a neural network model with an external memory and a recurrent attention mechanism for reading the memory. The model can be successfully trained via backpropagation on diverse tasks from question answering to language modeling.

The proposed model is comparable with conventional LSTMs and RNNs on language modeling tasks.

Increasing the number of memory hops will result in higher performance.

Future Scope

- Attention: The attention model may be improved if we replace the softmax probabilities with an attention model that treats sentences independently instead of normalizing as a mutually exclusive probability distribution.
- The accuracy of the model can be increased by increasing the number of memory hops and by training with different hyperparameter settings.
- Support for different input data formats:
  The implementation takes the input in the form of text files. It can be modified to read data directly from different input formats such as doc, PDF, html, etc.
- Classification of queries:
  The model can be extended by adding an additional network for classification of queries before answering an input query.

References

- Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston and Rob Fergus. 2015. End-To-End Memory Networks. Advances in Neural Information Processing Systems (NIPS) 28.
- Jason Weston, Sumit Chopra and Antone Bordes. 2015. Memory Networks. arXiv:1410.3916.
- Merity, Steve. ”Question Answering on the Facebook BAbi Dataset Using Recurrent Neural Networks and 175 Lines of Python Keras.”
- Harvard MemN2N GitHub repository. https://github.com/harvardnlp/MemN2N