Capstone Project

Machine Learning Engineer Nanodegree

Pavan C S

June 20th, 2017

Definition

Project Overview

Natural Language Processing is field in computer science which deals with interaction between humans and computers, by so making computer understand human languages. History of NLP goes as back as 1950. In the year, Alan Turing published an article titled "Computing Machinery and Intelligence" which proposed what is now called the Turing test as a criterion of intelligence. For a truly intelligent system, one of the important aspects is to understand human language and reply back as humans do. With this in mind, historically many work has been done till date, like, English to Russian translation in 1954 to today's IBM Watson, Apple Siri, Microsoft Cortana etc.

One of the sub problems in NLP is open domain question answering system. Given textual information to computer/agent, how accurately it can answer by understanding the context of the question. In this proposal, under open domain question answering system, proposing a project to build a supervised machine learning model to predict the answer in given paragraph/text.

I am looking forward to take up this particular project because of my personal interest in NLP which is one of the key aspects to develop human like AI and huge progress made in recent days. Another motivation for taking up this project is, deep learning foundation nanodegree which I am currently enrolled. Interested in solving the problem using techniques in deep learning like RNN, LSTM which are specifically tuned for this kinda problem.

Problem Statement

The ability to read text (Reading Comprehension) and then answer questions, is a challenging task for machines, requiring both understanding of natural language and knowledge about the

world. Here goal is create a question answering system, to be specific, an answer selecting model. This task involve following,

- Download **S**tanford **Qu**estion **A**nswering **D**ataset (SQuAD) dataset which is large enough to train an deep learning model
- Use Glove (word embedding) vector to get word vectors for SQuAD data vocabulary
- Train a deep learning model to select answer sentence in paragraph for a given query

Final application is expected to be an question answering system on top of comprehension

Metrics

Metrics for the model are exact match (EM)

Exact match (EM): these metric measures the percentage of predictions that match any one of the ground truth sentence.

In current problem, for a given question, I am trying to predict the correct answer sentence. Given 'n' number of question and 'm' number of correct answer sentence, above metrics measures % of answer sentence predicted correctly. It is similar to accuracy, but as author of the dataset is calling metrics as EM, I am also using the same.

To evaluate the solution, author of the dataset has provided two different test dataset. Dev set for developer to test the accuracy after training and a test dataset, which is not released for public to keep the integrity of the solutions. Will be evaluation on dev set provided.

Analysis

Data Exploration

Stanford Question Answering Dataset (SQuAD) is a new reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. With 100,000+ question-answer pairs on 500+ articles, SQuAD is significantly larger than previous reading comprehension datasets.

There are two dataset to train and evaluate model:

- train-v1.1.json
- dev-v1.1.json

Dataset is of two parts with following fields:

a. Paragraph
    - title_id: Wikipedia article title unique id
    - title: Wikipedia article title
    - paragraph_id: unique article's paragraph id
    - context: paragraph text
b. question and answer
    - question_id: unique question id
    - question: question text
    - answer_text: exact answer
    - answer_start: answer start index in corresponding paragraph
    - title_id: Wikipedia article title unique id
    - paragraph_id: unique article paragraph id

As data is in json format, it is parsed to create a data frame.

Exploratory Visualization

Following exploratory data analysis was performed to under dataset much better.

a. Few facts:
    - No. of articles in train data: 18896
    - No. of articles in test data: 2067
    - No. of question answer pair in train data: 87599
    - No. of question answer pair in test data: 34726
    - Vocabulary size: 83451

b. Type of question and distribution

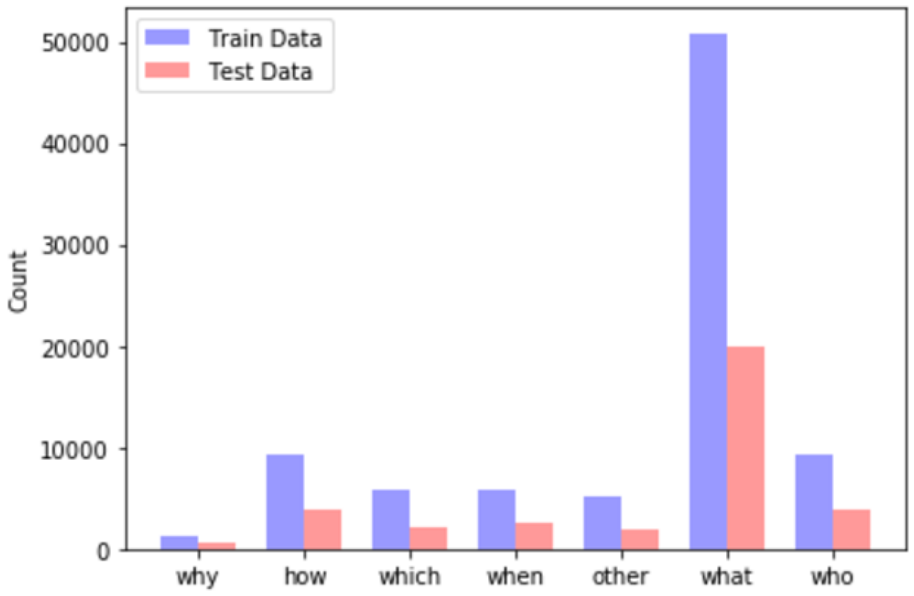

Fig a. Above figure shows different types of question distribution

Support code in analyse_data.ipynb

Algorithms and Techniques

Following algorithms and techniques are explored,

Recurrent Neural Networks (RNN) and Long Short Term Memory networks (LSTM): is a deep learning techniques which are specifically modeled for sequence-to-sequence processing. Specialty of RNN is that a neuron can pass on information it learnt to its successor like a loop.

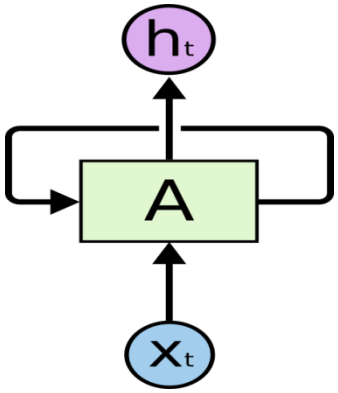

Fig 1: A typical RNN neuron.

Because of sequence-to-sequence property, RNN has huge impact in solving problem like speech recognition, language modeling, translation etc. LSTM is a special kind of RNN, which specifically designed for retaining for information for long term which is very helpful in language processing. Model is implemented using LSTM.

Glove: As neural network need numbers as input, words are represented as vectors using GloVe vectors. GloVe is pertained vector representation of words which captures word meaning.

Convolutional Neural Networks (CNN): is a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex. CNN can be convoluted over n dimension data with fixed dimension filters to create number of new layers which captures different patterns in given input.

k- Max Pooling: is a pooling layer which create new layer with 'k' number of values by taking maximum over input layer.

Dropout: is a regularization technique in deep learning to avoid overfitting. Based on dropout factor specified, number of neurons are randomly switched off to model to become more generalized.

Cosine similarity: captures how similar two vectors are.

Stochastic gradient descent (SGD): is astochastic approximation of the gradient descent optimization method for minimizing an objective function that is written as a sum of differentiable functions.

Hinge loss: is a type loss function which is defined as follows,

$$L = \max\{0, M - \text{cosine}(q, a+) + \text{cosine}(q, a-)\}$$

where a+ is a ground truth answer, a− is an incorrect answer randomly chosen from the entire answer space, and M is constant margin.

The following parameters can be tuned to optimize the model:

- Training parameters

- Training length (number of epochs)
- Batch size (how many sentences to look at once during a single training step)
- Solver type (what algorithm to use for learning)
- Learning rate (how fast to learn; this can be dynamic)
- Weight decay (prevents the model being dominated by a few “neurons”)
- Momentum (takes the previous learning step into account when calculating the next one)

❖ Neural network architecture

- Number of layers
- Layer types (LSTM, bi-LSTM , or pooling)
- Layer parameters

❖ Input data parameters

- Glove vector length
- Input question and answer max length

During training, both the training and the validation sets are loaded into the RAM. After that, random batches are selected to be loaded into the GPU memory for processing in cloud service ‘floydhub’.

Benchmark

Here want to define 3 Benchmark model,

a. Simple baseline model using sliding window: This model predicts answers with an accuracy of 20%.
b. Dataset author’s logistic regression model: This model predicts answers with an accuracy of 51%.
c. Human performance: Human’s answer prediction accuracy is of 86.8%

Please refer following link for author’s paper on dataset for benchmark model,

https://arxiv.org/pdf/1606.05250.pdf

Methodology

Data Preprocessing

Data preprocessing is divided into two steps,

a. Parse data: Parse data in json format into different files which can be converted to pandas dataframe. It is done for both train and test data. Files as follows
    - train_article.pkl, test_article.pkl : paragraph files
    - train_qas.pkl, test_qas.pkl: question answer files

Support code in parse_json.ipynb file

b. Prepare data: To prepare data from training model, following steps are performed.
    - Divide paragraphs in sentences
    - For each question, select top 'n' number of sentences based on question and sentence similarity.
    - Using answer start index, mark sentence as answer sentence or not
    - Do it for both for train and test data

Support code in prepare_data.ipynb

Implementation

Implementation can be divided into 2 stages:

a. Model training stage
b. Correct answer predicting stage

Model training stage:

Here following steps are performed,

1. Load prepared data(train and test data) and glove vectors into memory

2. From train data, choose question, correct sentence and in correct sentences for training
3. Select top ‘n’ sentence based on cosine similarity using glove, to improve the accuracy of prediction
4. Define parameter embedding glove vector dimension
5. Define parameter sentence and question max length
6. Define function for embedding words using glove vectors
7. Define function to pad question and sentence based on max length
8. Define the network architecture and training parameters
9. Define the loss function, accuracy
10. Define 10% of training dataset as validation set
11. Train the network, logging the validation/training loss and the validation accuracy
12. Plot the logged values
13. Save the trained network

Network Architecture:

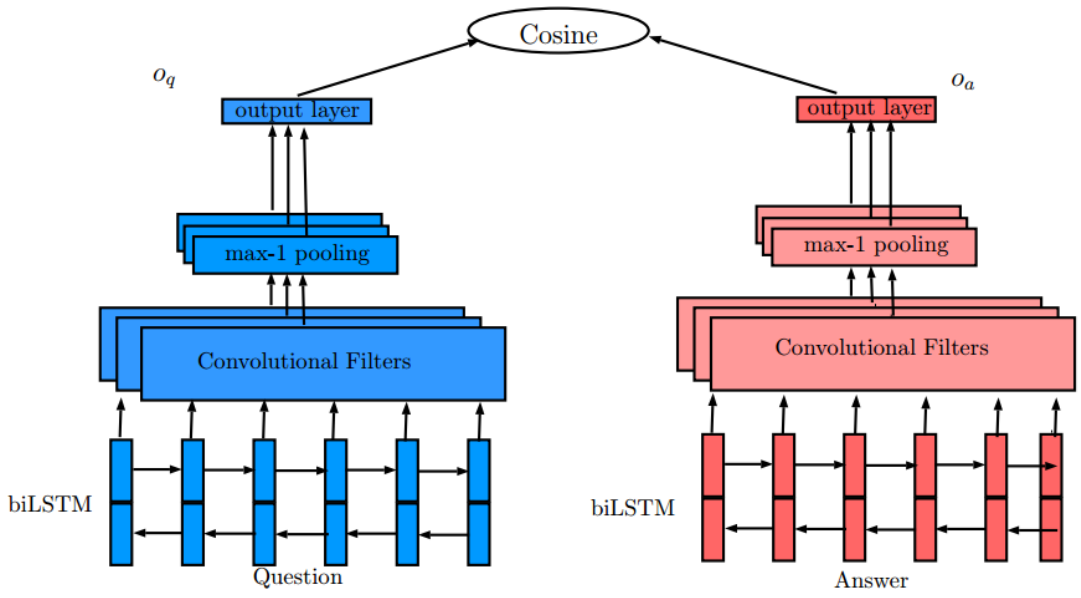

Fig b. above diagram shows deep neural network architecture/model implemented

Model is implemented with following steps,

1. Tokenized padded input data , question and a sentence, is embedded using glove vectors
2. Then passed question and sentence passed to shared bi Directional LSTM (biLSTM)layer
3. Then output is connected to Convolution Neural Networks (CNN) layer with filter size = 2
4. Then connected to max pooling layer with ‘k’ being ‘1’
5. The with activation function ‘tanh’, consine similarity is computed for both question and answer.
6. Loss function: To train model, loss function is defined as hinge loss to reward correct sentence and penalize wrong sentence

Correct answer predicting stage:

After training the model, to predict correct sentence given paragraph and question, following steps are performed.

1. Top 3 sentence based on cosine similarity are selected using glove vectors
2. Again new Consine similarity is computed between question and each sentence using trained model
3. Sentence with highest similarity is marked as correct answer

Refinement

To refine following steps are taken,

a. Architecture refinement:
    - Other architecture mention in paper are explored like architecture with CNN layer which underperformed
    - Instead of biLSTM, biGRU explored which under performed
    - Attention based model explored. Interestingly didn’t result in any improvement in accuracy
b. Hyper-Parameter refinement:

- biLSTM Dropout: dropout introduced to biLSTM layer to regularize
- biLSTM units: Number of units was finalized to 64 after experiment. Higher units didn't result in accuracy improvement
- CNN filter: Filter size was finalized to 2.Higher number didn't result in improvement
- CNN pooling units: Number of units was finalized to 3000. Higher number didn't result in improvement
- Optimizer : SGD with momentum. Adam didn't perfume will
- Learning Rate: After experiment, higher learning rate of 0.2 finalized
- Momentum: optimizer momentum was set to 0.9
- Decay : optimizer decay was set to 0.02

With this architecture and parameter, finally obtained accuracy of 58%

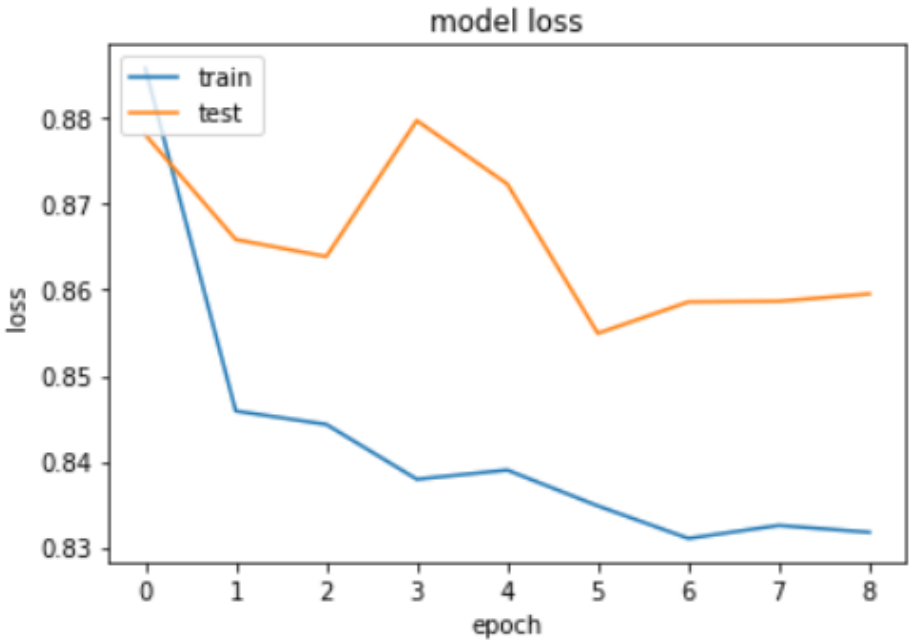

Fig b: above figure shows training/validation loss.

Results

Model Evaluation and Validation

During development, a validation set was used to evaluate the model.

The final architecture and hyper-parameters were chosen because they performed the best among the tried combinations.

Following are model details,

- Embedding vector size = 100
- biLSTM number of units = 64
- biLSTM dropout factor = 0.3
- CNN layer filter size = 2
- CNN layer units = 3000
- Max pooling activation = tanh
- Learning rate = 0.2 with decay
- Optimizer momentum = 0.9
- Optimizer decay = 0.02
- Number of epochs = 10

With these parameter, obtained accuracy of 58%.

To capture robustness of the model, following two parameters where observed,

- While training the model, 10% of data is kept as validation set. For each epoch, for both training and validation set, reducing loss is observed indicating model is learning.
- For each epoch, for test data (which is not seen by trained model), prediction accuracy is computed. Over the epochs, increased accuracy is observed as training loss reduced.

Justification

To understand successfully implementation of model it was important to check accuracy on test data. Model is tested on test dataset which is not seen by the model and then compared with benchmark mentioned above. With 58% accuracy, it is clearly beating simple baseline of 20% and baseline set by author of 50%. Which is pretty cool! Long way to beat baseline set by we humans ☺

Conclusion

Free-Form Visualization

Few predicted results are as shown below,

Paragraph:

```
Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title. The game was played on February 7, 2016, at Levi\'s Stadium in the San Francisco Bay Area at Santa Clara, California. As this was the 50th Super Bowl, the league emphasized the "golden anniversary" with various gold-themed initiatives, as well as temporarily suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so that the logo could prominently feature the Arabic numerals 50.
```

Question1:

Which NFL team represented the AFC at Super Bowl 50?

Predicted answer:

```
The American Football Conference (AFC) champion Denver Broncos defeated the National Football Conference (NFC) champion Carolina Panthers 24–10 to earn their third Super Bowl title.
```

Question2 :

```
'Where did Super Bowl 50 take place?'
```

Predicted answer:

```
The game was played on February 7, 2016, at Levi\'s Stadium in the San Francisco Bay Area at Santa Clara, California
```

Reflection

The process of the project can be summarized in following step,

1. To implement an answer selection problem a publicly available dataset was found which is large enough to train an deep learning model
2. Data was downloaded and processed
3. Start-of-paper were studying to different get deep learning architecture for problem stated
4. Model is implemented and trained on data
5. Finally model was tested on data to predict correct answer on test data

For me most difficult part was step 3 and 4 where going to paper and implanting the model. Initially faced lot of road blocks as deep learning and keras is new to me and took considerable time to implement. I have learnt a lot from this capstone project.

Also, What pretty cool about deep learning model here is, no paragraph/sentence structure information like POS tags, noun chunks etc where given. But still model was able to learn hidden patterns. That's one of the reason deep learning has huge implement on language modeling recently.

Improvement

Answer selection problem is still not a solved problem in academia. This is lot of scope in improvement. With respect to correct model implemented, could following improvement can be made.

1. Instead of predicting correct answer, predict exact word as answer
2. Try implementing even more advanced attention model to improve accuracy
3. Combine other techniques like reinforcement learning to improve the prediction using user feedback
4. Implement this as an QA application with better user interface