TBIR Project Report part 1 – Text Based Question Answering

Model Description:

This implementations consists of 3 models

1. **Word2vec model**: Converts each word in training data to a vector of size 100. This model is trained of entire training data (Question answer pairs - train) using word2vec model from genism package. It generates a vocabulary of 1002 words including start, end, pad and unknown tags.

2. **Autoencoder**: This model has an encoder-decoder architecture implemented using keras. The first layer (encoder) consists of bi-directional LSTM cells that creates a summarization of the input question. The encoder's final hidden state that represents a summarization of the input question is fed to the next LSTM layer (decoder) as input. This layer tries to reconstruct the question using the summarization produced by the encoder. The output of each decoder cell is converted to a probability distribution using softmax activation function. Here, number of classes is equal to size of vocabulary. Thus each unique word or tag represents one class. During prediction, the class (word) with maximum probability is taken as the output for each cell of decoder.

a. Important features:

- Added sample weights to each word during training. These weights are similar to TF-IDF. For each word, its frequency in a given question and the number of questions in a training batch that contain that word is used to determine its weights. This reduces effects due to frequently occurring stop words like 'the', 'on' etc.
- Used a Bi-directional LSTM for encoder layer. This enabled both forward and backward predictions given a word.

b. Results:

- The model can correctly predict WH words, verbs, prepositions and stopwords
- It correctly detects the position of nouns, verbs, prepositions. (Although does not predict the exact words.
- The model is able to correctly predict length of the question, its start and end words, and position of "?" and "<Pad>" tags.
- Due to pre-trained embedding's it is able to predict semantically similar words E.g. In place of "left" or "corner" it predicts "right"
- E.g. in place of "what is on the door?" it predicts "what is on the wall?")

c. Training details: Trained on 20 epoch with a batch size 50 using cross entropy loss and Adam's optimizer with default hyper-parameter settings in Keras

3. **Question answering model:** This model is an extension of above autoencoder model. The summarized question from encoder output is fed to an LSTM layer along with the answer

sequences. Similar to the autoencoder, output of each decoder cell is converted to a probability distribution using softmax activation function.

a. Important features:

- Teacher Forcing: The autoencoder output is concatenated with answer sequences and fed as input to the model. (Examples of model input: QuestSummary-<start>, QuestSummary-AnsSeq1, QuestSummary-AnsSeq2 …..). During prediction, the model is given only QuestSummary and input for prediction of first word. Then using the concatenation of predicted word and questSummary as input the prediction for second word is obtained. This is continued for the entire answer sequence.

b. Results:

- Predicts numbers for answers requiring numbers, colors for answers requiring colors etc.
- Predicts multiple words sequences separated by "," for answers with multiple words.

c. Training details: Trained on 20 epoch with a batch size 50 using cross entropy loss and Adam's optimizer with default hyper-parameter settings in Keras

Improvements:

1. Implement WUP score to check the correctness of predicted answers
2. Make the code modular and reduce repetition of code across notebooks
3. Use weights in Question answering model to reduce the effect of frequently occurring words in answers
4. Train autoencoder and question answering model on validation data to fine tune hyperparameters.
5. Make the code efficient to reduce training time
6. Try simpler models to reduce number of parameters.

Running the code:

- Set Keras backend to tensorflow
- Run the files in following order:
  1. Word embedding model.IPYNB
  2. Autoencoder-QA-3.IPYNB
  3. Q&A on loaded enc-dec.IPYNB

The folder also contains saved models and training and test data.