Develop and Validate an RNN Model to Teach Machines to Read and Comprehend

E4040.2019Fall.report
Junzhi Sheng js5300
Columbia University

Abstract

In this project, we propose a class of attention based deep neural networks and train them to read and comprehend articles. We first generate context-query-answer triples to solve the lack of labels in supervised learning, aiming at teaching the machine to give answer to queries after reading the articles from CNN dataset. However, since the sentences in CNN dataset is relatively long and the amount is comparatively large, it costs time to train our models and is hard to achieve a satisfying accuracy. We decide to apply data from SciQ and it performs better. In order to furtherly verify the benefit of Bi-directional LSTM and attentive layer on teaching machines to comprehend, we implemented an experiment of Multi-Choice Game in which we reached decent results.

1. Introduction

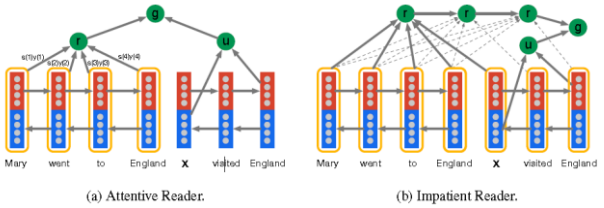
Teaching machines to read articles and comprehend them is a quite essensive path to build or design humanlike machines. Due to the lack of labels in supervised learning, machines do not know how to comprehend in a right way and we do not have a method to verify its capacity. The paper [2] introduced a method to generate context-query-answer triples as labels to supervise the learning period. Furthermore, applying Bi-LSTM and attentive layer are helpful tacic of helping machines analyse those text and give answers to questions.

In this project, we seek to verify the effectiveness of Bi-LSTM and attentive layer when we want to teach a machine to do reading based on different dataset. The first difficulty we faced is the large-scale data, which can cost us considerable time to load and train it. At the beginning of the project, we tried to use part of the text dataset from CNN which was employed in the original paper. However, since those sentences in the data are too long, we had trouble attaining satisfying output due to the gradient vanishing. We solved this by employing another dataset called SciQ which is smaller than CNN dara. We implemented a model using similar structure as the original paper and by appropriately decreasing the learning rate and using the gradient clip we successfully solved the convergence issue and then achieved a better result. However, it is still not result good enough to convince us the effectiveness of the model consists of Bi-LSTM and attentive layer due to the existence of overfitting. For further verification, we applied the same data on a brand new game called Multi-Choice Game. The model is trained to choose answers from 4 options. In this game, we achieved over 60% accuracy of both the best performed choice and the worst performed choice, which is convincing enough to state that attention model can be a beneficial tool for machines' comprehension.

2. Summary of the Original Paper

2.1 Methodology of the Original Paper

In order to solve the issue of lacking "labels" in this kind of supervised learning, they applied several strategies, such as be.0.1.V match, Permuted frame and Matching entity, to generate context-query-answer triples as labels from the original documents. Furtherly, they employed few document and query embedding models to learn from the articles, including different Symbolic Matching Models and Neural Network Models such as the Attentive Reader and Impatient Reader shown in Figure 1. Methodology shown in Figure 2.



(a) Attentive Reader. (b) Impatient Reader.

Figure 1: Document and query embedding models

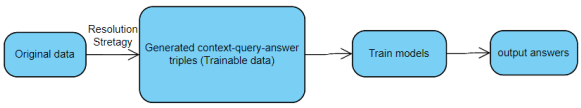


Figure 2: Methodology steps

2.2 Key Results of the Original Paper

	CNN		Daily Mail	
	valid	test	valid	test
Maximum frequency	30.5	33.2	25.6	25.5
Exclusive frequency	36.6	39.3	32.7	32.8
Frame-semantic model	36.3	40.2	35.5	35.5
Word distance model	50.5	50.9	56.4	55.5
Deep LSTM Reader	55.0	57.0	63.3	62.2
Uniform Reader	39.0	39.4	34.6	34.4
Attentive Reader	61.6	63.0	**70.5**	**69.0**
Impatient Reader	**61.8**	**63.8**	69.0	68.0

Figure 3: Results of the Original Paper

As shown in Figure 3, they achieved over 60% both validation and test accuracy when employing Attentive Reader and Impatient Reader on CNN documents.

3. DataSets Introduction

1. Original dataset:

This dataset contains two separate datasets which are CNN news and Daily Mail news with 39k and 69k in training, 3.8k in validation 3.1k in test. We don't use test data since we don't have more time. In each sample contains document (context), query, answer and entity mark. Here entity mark is like @entity1: New York, each word same as New York will use @entity1 to substitute in document. This is try to avoid model to only learn special word not correlation between document and query. The answer is all like entity mark and just one word. Thus we can predict answer is which entity mark without to predict the word in the whole dictionary.

2. SciQ dataset:

This dataset is 10k on training and 877 on validation. However, for document, query and answer only 5k can be used on training and 477 on validation. Same like original dataset input, but don't have entity mark instead answer can be any word inside corresponding document.

This dataset also can do Multiple Choice QA dataset, where the goal is to choose the correct option given any material available. For this purpose the output is 4 options probability.

3. Methodology

In this section, we will firstly introduce the objectives of our project in 3.1, followed by some technical difficulties we faced. Then we will go to problem formulation and design, which are solutions and methodology to overcome technical challenges above to achieve the goals.

3.1. Objectives and Technical Challenges

The main target of our project is to train our agent to be capable of reading and comprehending text documents. In a more detailed word, we want our agents to be able to give answers to questions after reading related articles. However, it is not very easy. We have been faced with some technical challenges: gradient vanishing due to the long input sentences, hard to find global min point due to really high dimensional cost function, hard to find correlation between document and query in attention layer alpha. Objectives challenges: large original dataset one is 38k and another is 69k, long training time need more than 10h to train 25 epoches only on 20k data.

3.2. Problem Formulation and Design

Objectives challenges:

1. large original dataset: we have two ways to deal with this problem one is use part of original dataset for example, we use 28k from first original dataset. Another is we use other dataset with near 10k.
2. long time training: use park of data to train and also use Adam as optimizer to accelerate gradient descent.

Technical challenges:

1. gradient vanishing: this problem only happens on first model. we use skip connection [4] in each rnn cell and reduce the batch size.
2. hard to converge: we reduce the learning rate to 0.0001 and use gradient clip by norm set value as 10.
3. hard to find correlation: we adjust the original paper attentive reader model structures. we didn't applied tanh activation when calculate alpha between document and query instead we use linear transformation also we didn't applied tanh activation before final predict instead we simple calculate linear dense layer.

1. Use engineering language and mathematical formulation;
2. Provide system drawings, block diagrams, and/or circuit schematics for your software or hardware design, as applicable to your project;
3. Include flow charts and pseudo code descriptions for the step-by-step discussion of your software design.

4. Implementation

In this section we will go through our implementation. First, our two-layer LSTM network is given in 4.1. Following is the algorithm applied. Second we will go through our attentive reader model given on 4.1. Then..

4.1. Deep Learning Network

First model

1. Two layer deep LSTM Reader.

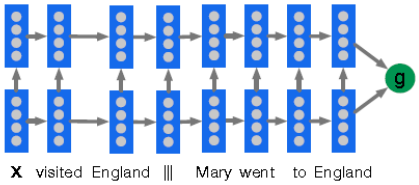


Figure 4: Two layer deep LSTM Reader

2. Training algorithm details:

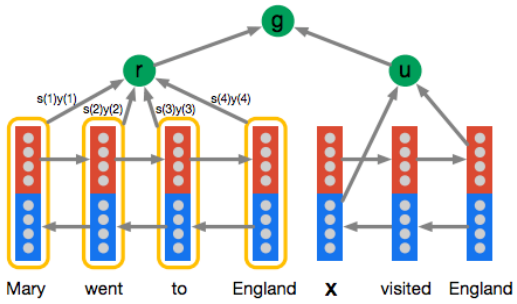
First, we build the word dictionary from training document data and training query data. Second we import one pre train word embedding matrix from stanford nlp. Third we build two layers forward direction lstm layers. Last output layer, we stack two rnn layers final output states and applied no activation dense layers as final output result.

3. Data used:

For this deep lstm model we applied original dataset but we only have time to use 28k over 38k for training and 2k over 3.9k for validation.

Second model

1. Attentive Reader.



(a) Attentive Reader.

2. Training algorithm details:

First we also build dictionary based on training document and query. Second also impot same pre train word embedding matrix. Third for document and query do one layer bidirectional lstm or gru encoder respectively. Fourth we build attention layer and find alpha. R is each t time document output times alpha. U is forward final output state stack with backward final output state. Last output layer is normal dense layer without activation.

3. Data used:

For this model we use two datasets. One is the original datasets we also use 28k over 38k for training and 2k over 3.9k for validation. Another one is SciQ datasets which same contains document, query, answer but each part is shorter than the original datasets. Also this dataset only have 5k training and 400 for validation we can use to train fast.

4.2. Software Design

Provide the following description and discussion:

First model:

1. Flow charts or flow charts, very often you should provide one top level flow chart, then additional flow charts for detailed lower level implementations.
2. Embedding layer: here input is one sequence which is concat with document and query, we set 0 as delimiter. we use pre trained word embedding from stanford nlp, in this model we use 300 dimensions as embedding layer. Also before encoder we applied one drop out layer.

   Encoder layer: here each layers are lstm cell use 128 units and dropout wrapper with only forward direction.

   Output layer: for two rnn layers, each has one final output state h, each h is [batch_size, units]. we stack those two final output states as [batch_size, 2*units]. After stack we applied dropout layer. Then we add a dense layer without activation function. The output dimension is [batch_size, num_label].

   Loss: we use softmax cross entropy loss function.

   Optimizer: we applied gradient clip by norm before optimizer and use clip norm value as 10.

   Hyperparameters: hidden unit: 128, dropout embedding rate: 0.1, dropout encoder rate: 0.2, learning rate: 0.001, optimizer: Adam, batch size: 32, epoches: 25.

Second model:

1. Embedding layer: here input is two sequence one is document another is query. we do the embedding respectively. Each we use pre trained word embedding from stanford nlp, in this model we use 100 dimensions as embedding layer. Also before encoder we applied one drop out layer to each one.

   Encoder document layer: here only one layer which is lstm or gru cell use 128 units and dropout wrapper with bidirectional direction.

   Encoder query layer: here only one layer which is lstm or gru cell use 128 units and dropout wrapper with bidirectional direction.

   Attention layer: first we calculate alpha, we stack query final output states and then applied dense layer as linear combination. Alpha in here is to find correlation between each word in document and query. Do softmax activation because we believe there is only piece of sequence in document should have high correlation with query. Then we use document encoder output times alpha as attention layer output.

   Output layer: we use alpha times document encoder final output. Then we add a dense layer with softmax activation function. The output dimension is [batch_size, num_label].

   Loss: we use softmax cross entropy loss function.

   Optimizer: we applied gradient clip by norm before optimizer and use clip norm value as 10. We use Adam, SGD, Rmsprop three options.

   Hyperparameters: hidden unit: 128, dropout embedding rate: 0.1, dropout encoder document

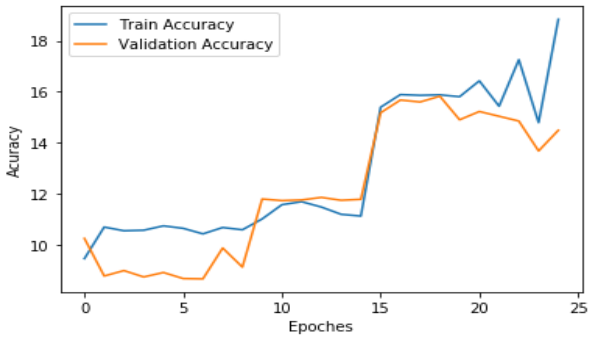
rate: 0.2, dropout encoder query rate: 0.1, learning rate: 0.001, optimizer: Adam, batch size: 32, epoches: 25.

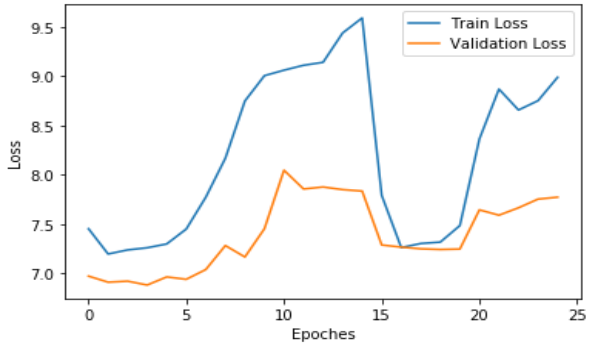
5. Results

5.1. Project Results

First model on cnn datasets :

1. Figures, plots:



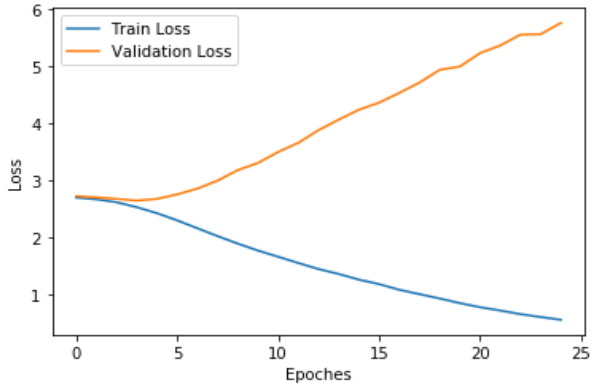


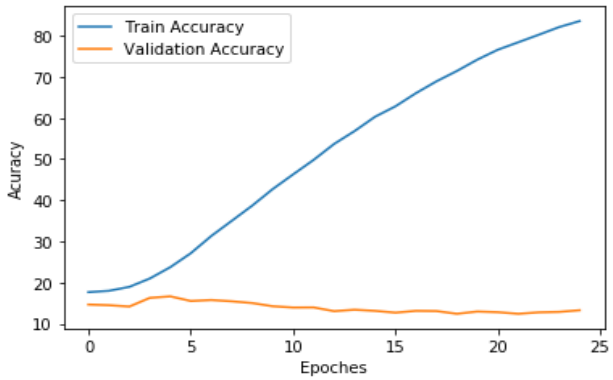
2. Description of results:

From above we can easily see train accuracy and validation accuracy keeps increasing. However, we believe train accuracy is still too low. Since 25 epochs takes near 10h to train, we decide to move to the second model and for no good train accuracy we will discuss in 5.3.

Second model on cnn datasets:

1. Figures, plots:



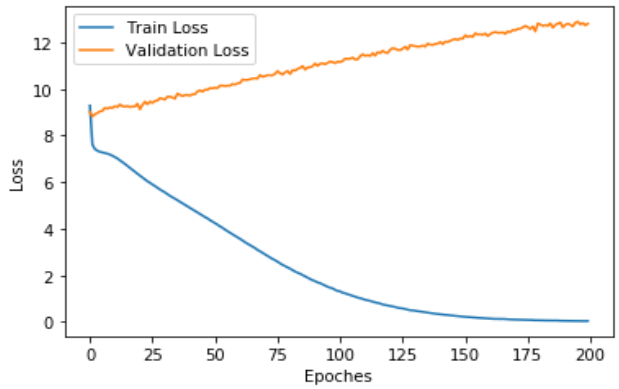


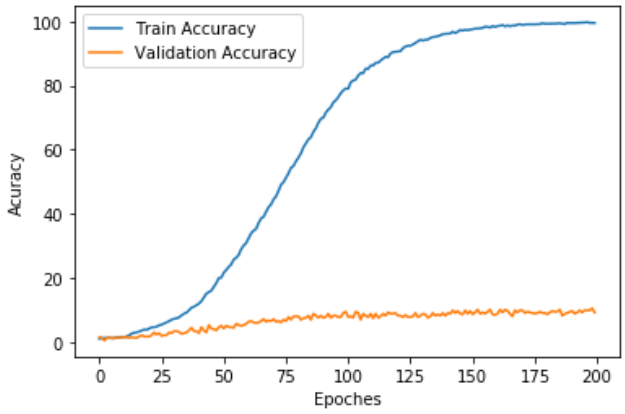
2. Description of results:

From above we can see training accuracy keeps increasing to above 80%. However, validation accuracy increase to highest at first 5 epoches then a little drop. We believe attentive model could learn more efficient and effective than deep lstm model. However, this model faces overfitting problem which we will discuss in 5.3.

Second model on SciQ datasets:

1. Figures, plots:





2. Description of results:

From above we could see training accuracy keeps increasing to near 99% and also validation accuracy keeps increasing but really slowly. We believe attentive model could really train effective and efficient even on different datasets. However, this model also faces overfitting but not serious than cnn datasets. At least this validation is keeping increasing. We will discuss in 5.3.

5.2. Comparison of Results

Provide detailed comparison between the results of the original paper and the students' project

Our best result:

CNN datasets	DeepLSTM Reader	Attentive Reader
Train	18.85	83.59
Valid	15.82	16.67

Compare with original:

1. DeepLSTM Reader:

Number of Train: we use 28k original: 38k + 69k
Number of Valid: we use 2k original: 3.9k
Number of dict: we ues 70k original: more than 100k
Validation accuracy: we get 15.82% original: 55%

2. Attentive Reader:

Number of data: we use 28k original: 38k + 69k
Number of Valid: we use 2k original: 3.9k
Number of dict: we ues 70k original: more than 100k
Validation accuracy: we get 16.67% original: 61.6%

From the above table we find attentive model is better than deep lstm model even on our part of datasets on both training process and validation process. We also do attentive reader on different dataset.

	Attentive on CNN	Attentive on SciQ
Train	83.59	99.79
Valid	16.67	10.48

Compare with different datasets:
Number of Train: CNN 28k SciQ: 5k
Number of Valid: CNN 2k SciQ: 477
Number of words in dict: CNN 70k SciQ: 34k

From the above table, we find both training accuracy is increasing and is really good. Although from table attentive on CNN validation looks better, this highest accuracy is get in first 5 epoch. In SciQ dataset attentive model is increasing to get highest validation accuracy which means is increasingly coverage We will discuss this in 5.3. .

5.3. Discussion of Insights Gained

Provide detailed discussion, regardless of the actual results: why are your results different, did you use smaller dataset, did you use different hyper-parameters, number of epochs different?

First discuss difference with original datasets.

DeepLSTM Reader:

1. difference: underfitting with train only 18.85%

why: first actually from the 5.1 first model figure, our training accuracy is still increasing but we really don't have enough time to train more epoches because we train 25 epoch it takes 8h. Second there do have some other problems with this model. The average length of input sequence is 777, even with skip connection [4] we believe it is hard to find good gradient. Third this model encoder sequence is concat of document and query but we think forward direction cannot find piece of import sequence in document which will influence query and this sequence is too long.

2. difference: overfitting with validation 15.82%

why: Even our result shows validation only lower than train accuracy few percent, but obvious this model will overfitting if we do more epoches. First reason, our training data only 28k, but original train data is 38k. Second reason, we use our part of training data build our dictionary which is 70k, but we find in one open source for original paper, they use original two datasets which are CNN and Daily Mail to build dictionary. Third in our validation data 2k, we find there is 2324 words which is 10.3% of our validation dataset (validation dataset contains 22532 unique words) is not in our training word dictionary. We think this also influence our results since lots of words never seen.

Attentive Reader:

1. difference: overfitting with validation 16.67%

why: First reason, our training data only 28k, but original train data is 38k. Second dictionary problem totally same as deep lstm model. Third optimizer hyperparameters problem, we try two combines which are sgd with 0.1 learning rate and adam with 0.001. The sgd even can't get good training accuracy. We think optimizer hyperparameters and other hyperparameters like dropout rate will influence our validation results. However, we don't have enough time to train since 25 epoches takes 10h to train.

Second discuss difference on same model but on different CNN datasets and SciQ datasets.

1. difference: in CNN dataset model validation is not converge but in SciQ dataset model validation is increasingly converge.

why: first, in SciQ dataset the average length of document is 76 and query is 14, however in CNN dataset the average length of document is 763 and query is 14. Thus we think for shorter sentence model will more easy to find which piece of sequence in document has high correlation with query. However, for longer sentence maybe

hyperparameters hidden units, dropout rate and learning rate will influence overfitting.

2. problem: in SciQ datasets, the validation converge too slow.

why: first, SciQ datasets is not really big with only 10k and after we clean some sample don't have document. There is around 5k dataset can be used for training and 477 for validation. Second, we find in SciQ validation dataset there is 23% of words are not in word dictionary which is built by training datasets. There are too many words not train in training process. This maybe the main reason to influence validation accuracy increase slowly.

6. Conclusion

1. DeepLSTM model can't find correlation between document and query and is easily get gradient vanish. This model is not good for teaching machine reading and comprehension.
2. Attentive Reader model can effective and efficient find correlation between document and query, perform better on input document sentence length not too long like more than 700.

Future Improvement:

1. Train on bigger size dataset hope to use all original datasets.
2. Build word dictionary hope to use all original dataset which are 39k + 69k.
3. Try more hyperparameters combination especially on tuning optimizer, learning rate, dropout rate.
4. Training on more epoches from 50 to 100.
5. Training options together in Multi-Choice game to take into consideration the relationship between options.

7. References

[1]https://github.com/orgs/cu-zk-courses-org/teams/jsxl

[2]Hermann, Karl Moritz, et al. "Teaching machines to read and comprehend." *Advances in neural information processing systems*. 2015.