E1-246 Natural Language Understanding
Project Report
Hierarchical Attention for Document Classification

Shivank Gupta (14638) {shivankgupta
Akash Mishra (14421) akashmishra
Tapan Bhardwaj (14704) tapanb}@iisc.ac.in

Abstract

In our project, we are implementing a hierarchical attention network for document classification. Every document has some kind of hierarchy such as collection of words make sentences and collection of these sentences creates a document. In our approach , the algorithm, which we have used, replicates this hierarchical structure of a document. In every document, there are only a few words or sentences which are helpful in classifying document. So we used attention mechanism applied at both word level and sentence level. By this approach our model will give more attention to more informative words whereas less attention to less informative words. Intuitively this should be a better approach for document classification which we will check by implementing this algorithm over IMDB and YELP dataset.

1 Introduction

Among the many tasks in natural language processing, text classification is a very fundamental task in which we assign labels to text. There are many approaches for doing so. In a traditional approach, we represent documents with sparse lexical features such as n-grams and then use a linear model on this representation. Another approach which uses deep learning such as constitutional neural network and recurrent neural network based on long-short-term memory to learn text representation. Although these neural network based approaches for text classification has been proven quite effective but they do not use any kind of attention in their approach. Since we know that not all part of a document is equally effective for the classification task so It's obvious that a better representation can be obtained by including this knowledge of document structure in model architecture. In our algorithm, we have used this same approach for document classification.

We used Hierarchical Attention Network in our approach which provides two basic insight about document structure. First, word level and sentence level hierarchical attention and second determining more important words and sentences in a given document. But the importance of words and sentences is highly contextual. That's why we used two level of attention mechanisms, one at the word level and one at the sentence level. For example, if a food review document has a word 'delicious', it means that there are high chances that the review is positive but same word 'delicious' does not have the same importance in a movie review document.

To evaluate the performance of this model, we look at two different dataset, YELP dataset and IMDB dataset and our model outperforms previous approaches by a significant margin.

2 Recurrent Neural Network(RNN) Units

2.1 Gated Recurrent Unit (GRU)

GRU is improved version of recurrent neural network. It was introduced by Cho, et al. in 2014 to solve the vanishing gradient problem of RNNs. GRU uses update gate and reset gate to decide that what information is important and should be passed to the output and what information is irrelevant and should be removed. Fig.1 shows the single unit of GRU.

To calculate the **update gate for time stamp t** we use the formula-

$$z_t = \sigma(W^{(z)} * x_t + U^{(z)} * h_{t-1}) \qquad (1)$$

This update gate helps the model to determine how

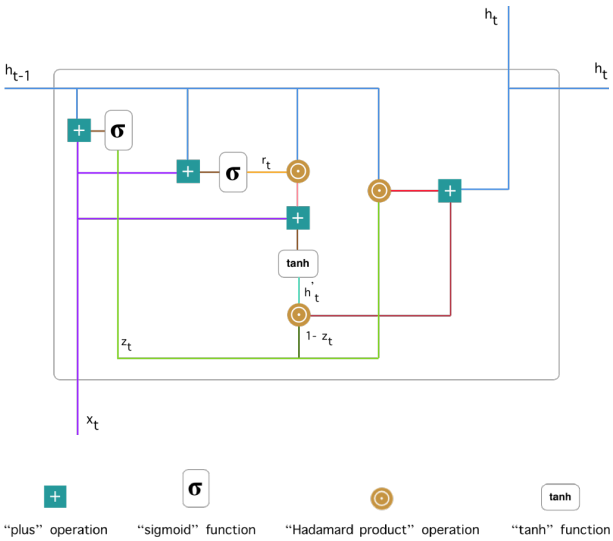


Figure 1: *GRU*

much of the past information(from previous time steps) should be saved.

To calculate the **reset gate for time stamp t** we use the formula-

$$r_t = \sigma(W^{(r)} * x_t + U^{(r)} * h_{t-1}) \quad (2)$$

This reset gate helps the model to determine how much of the past information(from previous time steps) should be removed.

With the help of these two gates, model calculates the final memory content as-

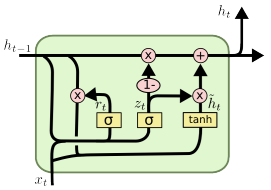
$$h_t^{'} = tanh(W x_t + r_t \circ U h_{t-1}) \quad (3)$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ h_t^{'} \quad (4)$$

2.2 Long Short-term Memory(LSTM)

LSTMs are a variations of RNN. It was introduced by Hochreiter & Schmidhuber in 1997 to avoid the long term dependency problem of RNN. Remembering information for a long period of time is their default behavior.

Instead of having a single neural network layer, the repeating module of LSTM has four neural network, interacting in a very special way. Because of which, same as GRU, LSTM also has the advantage of remembering important information and removing not so important information. Fig.2 shows a working diagram of a LSTM and equations related to it.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 2: *LSTM*

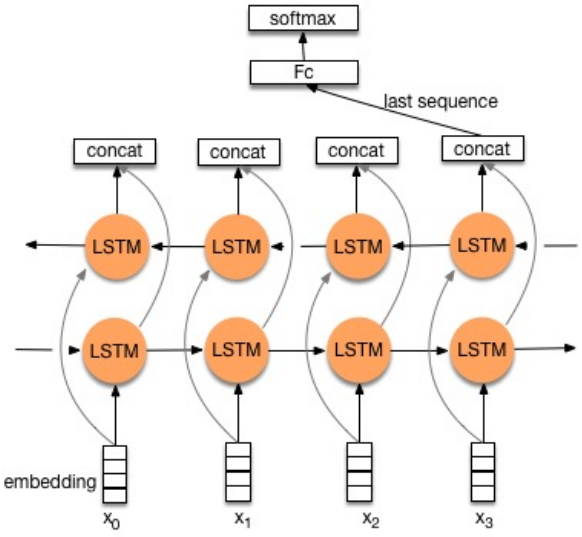


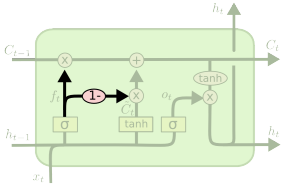
Figure 3: *Bidirectional LSTM*

2.3 Bidirectional LSTM

Bidirectional LSTMs are an extension of traditional LSTMs that can improve model performance on sequence classification problems. It was invented by Schuster and Paliwal in 1997. In problems where all timesteps of the input sequence are available, Bidirectional LSTMs train two instead of one LSTMs on the input sequence. The first on the input sequence as-is and the second on a reversed copy of the input sequence. This can provide additional context to the network and result in faster and even fuller learning on the problem.

2.4 Cached LSTM(CLSTM)

Although neural networks has been proved very useful in sentiment analysis but they still had one



$$C_t = f_t * C_{t-1} + (1 - f_t) * \tilde{C}_t$$

Figure 4: *Cached LSTM*

remaining drawback which is to model long text in document level sentiment classification under a recurrent architecture because of the deficiency of the memory unit. To remove this problem, we use CLSTM. It introduces a cache mechanism, which divides memory into several groups with different forgetting rate which helps the network to keep sentiment information better within a recurrent unit.

3 Hierarchical Attention

This model projects the raw document into a vector representation, on which we build a classifier to perform document classification.

3.1 Word Encoder

Given a sentence with words,we first embed the words to vectors through an embedding matrix. We use a bidirectional LSTM to get annotation of words by summarizing information from both directions for words and therefore incorporate the contextual information in the annotation.

We obtain an annotation for a given word by concatenating the forward hidden state and backward hidden state which summarizes the information of the whole sentence centered around the words.

$$x_{it} = W_e w_{it} \tag{5}$$

$$\vec{h_{it}} = \vec{LSTM}(x_{it}) \tag{6}$$

3.2 Word Attention

Not all words contribute equally to the representation of sentence meaning. Hence, we use attention mechanism to extract such words that are important to the meaning of the sentence and aggregate the representation of those informative words to form a sentence vector i.e. we first feed the forward annotation through a one layer MLP then we measure the importance of the word and get a normalized importance weight through a softmax function.

After that, we compute the sentence vector as a weighted sum of the word annotations based on the weights. The word context vector is randomly initialized and jointly learned during the training process.

$$u_{it} = tanh(W_w h_{it} + b_w) \tag{7}$$

$$\alpha_{it} = exp(u_{it}^T u_w)/\Sigma_t exp(u_{it}^T u_w) \tag{8}$$

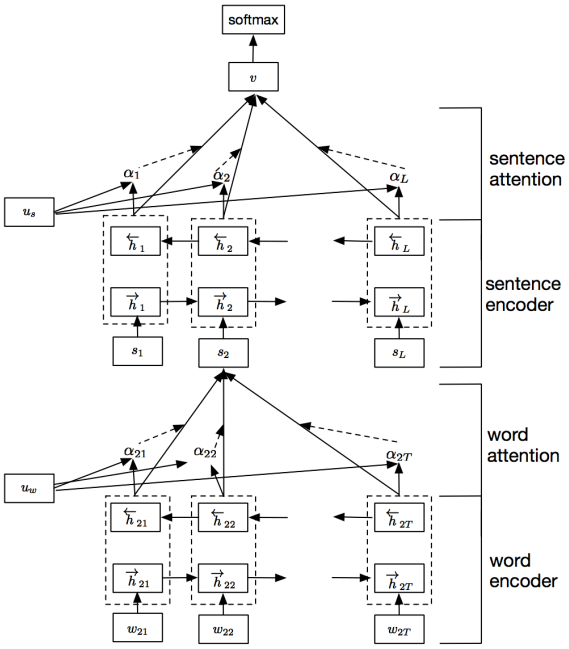


Figure 5: *Hierarchical Attention Model*

$$s_i = \Sigma_t \alpha_{it} h_{it} \tag{9}$$

3.3 Sentence Encoder

Given the sentence vector, we can get a document vector in similar way. We used a bidirectional LSTM to encode the sentences. We concatenate the forward hidden state and backward hidden state to get an annotation of the sentence. The hidden states of a particuler sentences summarizes the neighbour sentences also but still focus on that particuler sentence.

$$\vec{h_i} = \vec{LSTM}(s_i) \tag{10}$$

3.4 Sentence Attention

Not all sentences are equally useful in correctly classifying a document. Here, we again use attention mechanism and introduce a sentence level context vector and use this vector to measure the importance of the sentences. Now, we create a document vector that summarizes all the information of sentences in a document. Similarly, the sentence level context vector can be randomly initialized and jointly learned during training process.

$$u_i = tanh(W_s h_i + b_s) \tag{11}$$

$$\alpha_i = exp({u_i}^T u_s)/\Sigma_i exp({u_i}^T u_s) \tag{12}$$

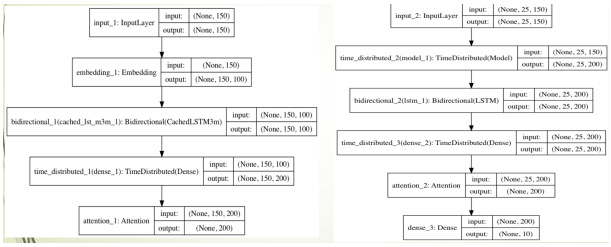


Figure 6: *Model description*

$$v = \Sigma_i \alpha_i h_i \tag{13}$$

3.5 Document Classification

The document vector generated after sentence attention is a high level representation of the document and can be used as feature for document classification using a softmax function.

$$p = softmax(W_c v + b_c) \tag{14}$$

We use negative log likelihood of the correct labels as training loss:

$$L = -\Sigma \log p_{d_j} \tag{15}$$

where j is the label of document d.Description of Model is given in figure 6.

4 Data Sets

We have used Yelp'13 and IMDB(movie review) dataset for our analysis.

- Yelp'13 dataset is obtained from the Yelp Dataset Challenge in 2013. It contain restaurants reviews ranging from 1 to 5.
- IMDB dataset contains the reviews of movies from 1 to 10. so it is a 10-class problem and its a huge dataset with over 3 lakhs reviews in training set while 10 percent of it in test dataset.

5 Experiments

We have experimented our model on two large datasets that are very popular in document classification. We have used 90-10 train-test split while we have not divided it into validation set as we were not able to tune the hyperparameter.

Yelp'13 Dataset

	LSTM	Cache LSTM
Accuracy	63%	65%
Loss	0.79	0.77

IMDB Dataset

	LSTM	Cache LSTM
Accuracy	38.5%	40%
Loss	1.55	1.49

Figure 7: *This figure shows accuracy and losses on different datasets for different bidirectional architectures*

5.1 Preprocessing

Since documents contain lot an arbitrary no. of sentences again contain arbitrary no. of words therefore it is necessary to preprocess the document into fixed length document so that we can feed it into the model. We have used pretrained 100-dimensional glove vector embeddings for words representation and we have fixed the size of each sentence to 100, in case of short sentences padding is done while in case of long sentences some part of it is trimmed of. We have also kept the size of document to have at max 25 sentences.

5.2 Results

In our analysis we have used two bidirectional architectures for word and sentence encoders.

- Bi-Directional LSTM
- Bi-Directional Cached LSTM

We have run our model on the above given datasets and obtain the results as shown in figures. We have run our model only for 5 epochs since we were short of resources therefore the results were not that much convincing, if we could have run the model for more no. of epochs then we could have obtain promising results.

6 Conclusion

- As Cache LSTM is capable of capturing long distance dependencies better than LSTM, It gives improved results as compared to normal LSTM on all the Datasets that we have experimented with.

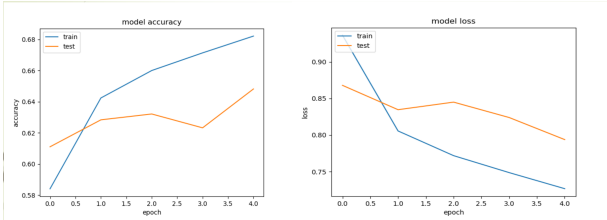


Figure 8: *This figure shows Accuracy and Loss on YELP'13 dataset with LSTM*

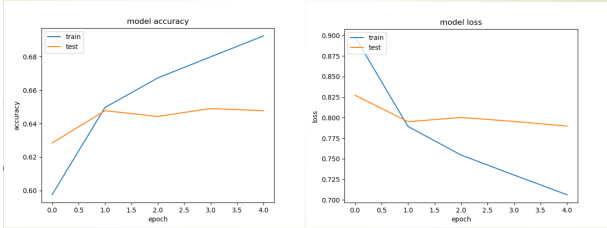


Figure 9: *This figure shows Accuracy and Loss on YELP'13 dataset with Cache LSTM*

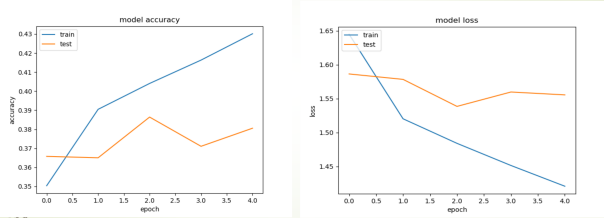


Figure 10: *This figure shows Accuracy and Loss on IMDB dataset with LSTM*

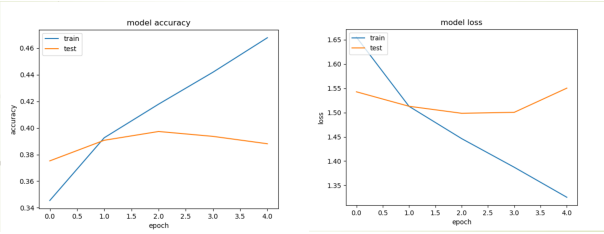


Figure 11: *This figure shows Accuracy and Loss on IMDB dataset with Cache LSTM.*

- However, learning of Cache LSTM is very slow as compared to normal LSTM. It is 5 times slow as compared to normal LSTM.

- Cache LSTM can be used in place of normal LSTM to produce state of art results if we have sustainable amount of computation power.

7 References

- Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, Eduard Hovy *Hierarchical Attention Networks for Document Classification* `http://www.cs.cmu.edu/~/hovy/papers/16HLT-hierarchical-attention-networks.pdf`

- Jiacheng Xu, Danlu Chen, Xipeng Qiu, Xuanjing Huang *Cached Long Short-Term Memory Neural Networks for Document-Level Sentiment Classification* `https://arxiv.org/pdf/1610.04989.pdf`

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio.2014 *Neural machine translation by jointly learning to align and translate* `https://arxiv.orgpdf/1409.0473.pdf`