Article Classification using Hierarchical Attention Networks
Project Report
ECE-GY-9123 Deep Learning

Sahil Makwane (sm9127), Sakshi Mishra (sm9268)

1 Problem Statement

Large volumes of scattered and uncategorized articles present on the internet can cause difficulty in accessing desired information. This issue engenders the need of classification of articles in groups with tags so that the user can quickly retrieve articles they wish for. Classification of text can be done manually but it is an arduous task. Thus, we introduce machine learning techniques for text classification so that text classification can be done quickly for large amounts of textual data.

2 Literature Survey

Articles are text documents that can be considered as a sequence of sequence. Meaning, sequences of words makeup a sentence, and sequences of sentences make up an article. There are unsupervised techniques for text classification such as Bag of Words, TF-IDF and N-grams which represent the text in terms of frequency of occurrence of words. However, these do not account for the contexts of words and sentences with much accuracy.

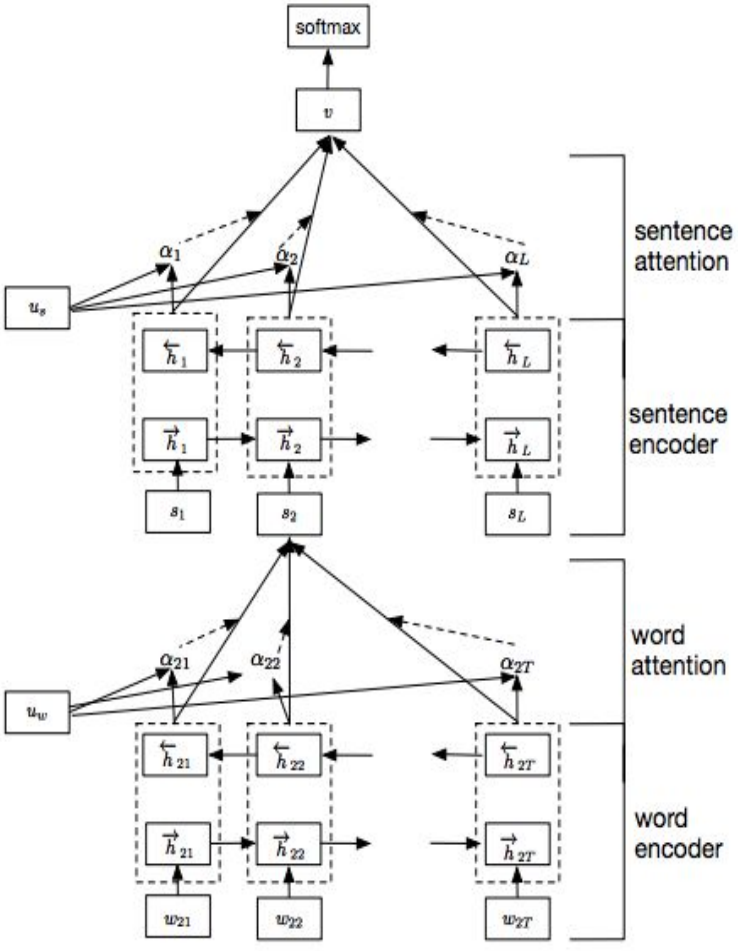
On the other hand, Hierarchical Attention Networks or HANs consider the hierarchical structure of the document. They also account for the context of the words and sentences that appear in the document. For instance, the context of the word 'book ' in the sentences, "I read a book" and "I need to book flight tickets" are different. HANs also takes into consideration that not all words and sentences equally contribute towards the representation of the article.

We shall be implementing attention mechanism both on word level and sentence level. The attention mechanism sets weights on words and sentences depending upon their context. The network uses word encodings of each sentence and then applies attention mechanism on the words. This results in a sentence

vector. Now these sentence vectors are encoded, and attention mechanism is applied to each sentence vector. The network congregates such sentence vectors to represent the document.

3 Model

First the text in our datasets will be preprocessed by lemmatizing and tokenization techniques. This preprocessed data is further passed as in input to the HAN model. Input is a 2D Tensor with dimensions in terms of (samples, steps, features). The output is a 2D tensor with dimensions in terms of (samples, features).



Hierarchical Attention Network

The Hierarchical Attention Networks model consists of the following parts:

1. **Embedding Layer**: This layer shall convert words into a vector of real numbers.
2. **Word Encoder Layer**: To get a rich description of words, bi-directional GRU as used at the word level.
3. **Word Attention layer**: To get the important information from each word in a particular sentence, attention mechanism is applied at the word level.
4. **Sentence Encoder Layer**: To get a rich description of sentences, bi-directional GRU as used at the sentence level.
5. **Word Attention layer**: To get the important information from a sentence, attention mechanism is applied at the sentence level.
6. **Fully Connected layer**: cascaded with a softmax layer will predict the probabilities for the document for each category.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 100)]             0
_________________________________________________________________
embedding (Embedding)        (None, 100, 100)          8056700
_________________________________________________________________
bidirectional (Bidirectional (None, 100, 200)          121200
_________________________________________________________________
time_distributed (TimeDistri (None, 100, 200)          40200
_________________________________________________________________
att_layer (AttLayer)         (None, 200)               20200
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 402
=================================================================
Total params: 8,238,702
Trainable params: 8,238,702
Non-trainable params: 0
```

Our Model

4 Datasets

Following datasets will used to implement the model:

1. IMDB Movie Reviews: Has labeled data of 50,000 IMDB movie used for sentiment analysis. For reviews having rating <5 the sentiment score is set to 0 and for rating >=7 sentiment score is 1.

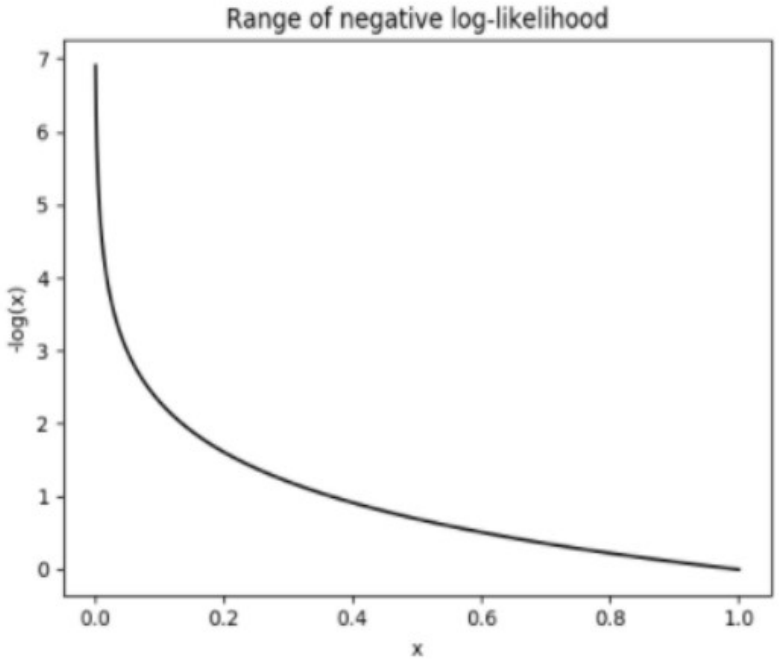
2. Yelp Reviews: Consists of yelp reviews for different businesses. It has ratings from 1 to 5, 5 being the highest.

3. News Category Dataset: Consists of 18,000 newsgroup documents from 20 different newsgroups.

5 Training

The maximum sentence length has been set to 100. Maximum number of sentences and words a review can have is set to 15 and 20000 respectively. First we preprocessed the data by removal of stop words, tokenisation and lemmatisation. For vectorisation we have used GloVe at size 100 to learn the word embeddings. Glove stands for Global Vectors for Word Representations is an algorithm to vectorise words with suitable representation.

With validation split set as 0.2 we split the dataset for train, validate and test. We used a Bidirectional GRU for contextual information about a word. For classification purpose we have used cross-entropy loss function. We have also used the Negative Log Likelihood Loss function in another notebook to draw a comparison. The NLL assigns higher loss to results having smaller prediction values, thus eliminating the inputs related to those results, whereas inputs with higher probabilities are assigned smaller losses.



Range of negative log-likelihood

The loss function reaches when input is 0, and reaches 0 when input is 1.

We have taken a 2 dimensional input of sentences and text IMDB dataset. The input dataset consists of sentiments according to which reviews are categorised. To develop our model we used the "TimeDistributed" function from Keras to develop the Hierarchical input layers.The 'Attention Layer' using a custom Keras layer which is implemented on Theano backend. A dense layer has also been implemented whose input is the output of the GRU. Then the output from the dense layer is then fed into the attention layer. We trained the model with 10 epochs.

6 Results

We worked on several implementations where we compared the models and also developed our own model:

1. **Notebook 1:** https://drive.google.com/file/d/10B1uxyuEZwqBgBrA_D_bqdTgsLLkwGDm/view?usp=sharing
2. **Notebook 2:** https://drive.google.com/file/d/1HR0Eca6HFlkr1RehuqpdnamBeTZSupxg/view?usp=sharing
3. **Our Notebook:** https://github.com/SMak06/HAN-doc-classification

Challenges

1. The implementations mostly feature deprecated libraries and methods that need to be upgraded to the latest standards in order to run the model without errors.
2. The implementations have not been assisted with comments, hence often render errors that cannot be easily diagnosed by other users.
3. We faced the issue of increasing the accuracy of the models.
4. We incurred a high loss for another model we tried implementing.

Comparison			
Notebook	Vectorizer	Loss	Valuation Accuracy
Notebook 1	word2Vec	1.24	65.93%
Notebook 2	GloVe	484.77	10.10%
Our Notebook	GloVe	0.0083	83.20%

7 Conclusion

We sought out to perform a comparative study for the different Hierarchical Attention Network models. Through the course of our project we decided to take upon the major implementations of https://www.cs.cmu.edu/ ./hovy/papers/16HLT-hierarchical-attention-networks.pdf. We tried to replicate the intended results

of the respective notebooks but ended up upgrading them or making changes in order to make them work. We thus present the results we obtained in the result section. Among the array of implementations available online, many failed to function due to the lack of documentation or proper code. We thus formed our own HAN model for document classification. We used pre-trained word vectors from GloVe (global vectors for word representation). On multiple runs our methods obtained us a validation accuracy around 83%. Thus, we conclude that our comparative study has been a success where we were not only able to peruse several HAN implementations but were also successful in building a model which produces relatively higher accurate results.

8 References

1. https://www.cs.cmu.edu/ ./hovy/papers/16HLT-hierarchical-attention-networks.pdf
2. https://nlp.stanford.edu/projects/glove/
3. https://github.com/richliao/textClassifier
4. https://humboldt-wi.github.io/blog/research/information_systems_1819/group5_han/
5. https://www.kaggle.com/c/word2vec-nlp-tutorial/download/labeledTrainData.tsv
6. https://www.github.com/pandeykartikey/Hierarchical-Attention-Network
7. https://www.kaggle.com/yelp-dataset/yelp-dataset
8. https://www.kaggle.com/crawford/20-newsgroups
9. https://github.com/arunarn2/HierarchicalAttentionNetworks
10. https://www.medium.com/analytics-vidhya/hierarchical-attention-networks-d220318cf87e
11. http://www.github.com/tqtg/hierarchical-attention-networks