ECE-GY 9123 / Spring 2021 - **Deliverable 3: Project update**

Article Classification using Hierarchical Attention Network

Sahil Makwane (sm9127)
Sakshi Mishra (sm9268)

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.

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.

Dataset

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. News Category Dataset: Consists of 200,000 news headlines for the years 2012-2018.

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

Model

Primarily we worked on the following implementation:
https://drive.google.com/file/d/10B1uxyuEZwqBgBrA_D_bqdTgsLLkwGDm/view?usp=sharing

First we preprocessed the text in our dataset by lemmatizing and tokenization techniques.

This preprocessed data is further passed as in input to the HAN model. Input is a 3D Tensor with dimensions in terms of (samples, steps, features). The output is a 2D tensor with dimensions in terms of (samples, features).

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 is 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.

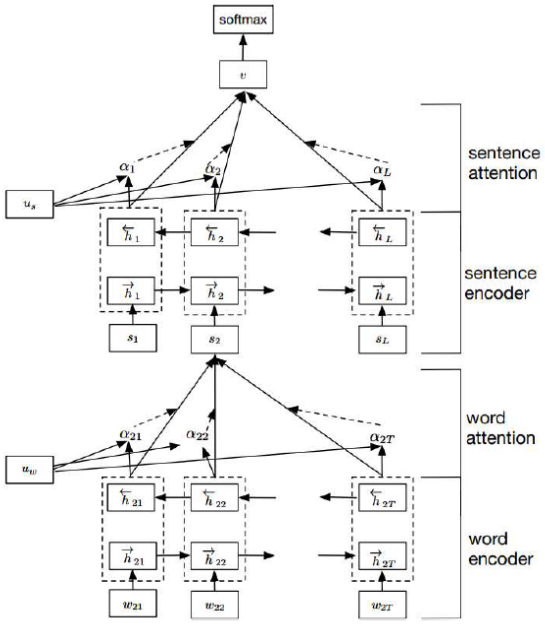

Figure: Hierarchical Attention Network

Source : https://www.cs.cmu.edu/~./hovy/papers/16HLT-hierarchical-attention-networks.pdf/

Training

First we preprocessed the data by removal of stop words, tokenisation and lemmatisation. For vectorisation we have used WORD2VEC at size 200 to learn the word embeddings. In the ratio of 8:1:1 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. 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.

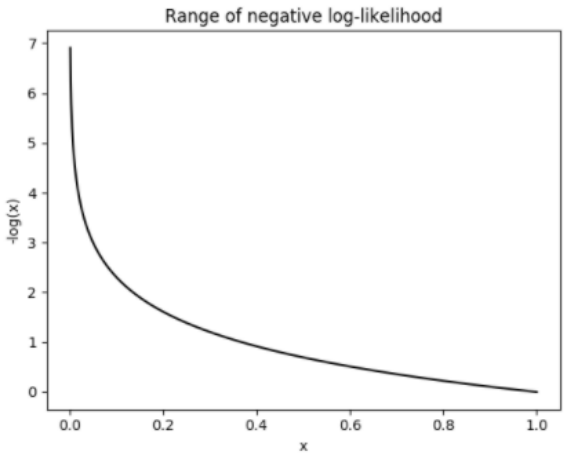

Figure: *The loss function reaches infinity when input is 0, and reaches 0 when input is 1.*

Source : https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood/

We started training the model with 9 epochs, and increased the number of epochs but there weren't significant improvements in the accuracy. The batch size is 64 and the learning rate is set at 1e-5. Maximum Sentence Length is set at 25.

Preliminary results

We are working on several implementations where we are comparing the models and their accuracy. As an intermediate result, here we have 3 different implementations.

1. We secured a testing accuracy of ~65% with our first implementation. Dataset - Yelp

   https://drive.google.com/file/d/10B1uxyuEZwqBgBrA_D_bqdTgsLLkwGDm/view?usp=sharing

2. This second implementation consists of an Embedding layer, Word Encoder with a word level bi-directional GRU, Word Attention, Sentence Encoder, Sentence Attention, Softmax. The model has 2 attention levels, one at word and another at sentence level which allows more or less attention to be paid to individual words/sentences when constructing document representation. Dataset - IMDB Movies Dataset from Kaggle, labeledTrainData.tsv - 25000 reviews with labels

   https://drive.google.com/file/d/1WNaJsMBf2LNDS_d62-a-OlSv6lNe1Yn4/view?usp=sharing

3. Another implementation but with a very high Loss

   https://drive.google.com/file/d/1HR0Eca6HFlkr1RehuqpdnamBeTZSupxg/view?usp=sharing

Final Project Concept

We shall be comparing several different implementations of HAN for classifying documents. We would be comparing the models, and their structure. Our end goal is to implement a model with a high level of accuracy over the different datasets. We think our final result would be a comparison of the different existing models along with our own implementation. Once we get significantly better accuracy, we need to run the model on other datasets as well.

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.

References

1. https://www.cs.cmu.edu/~./hovy/papers/16HLT-hierarchical-attention-networks.pdf
2. https://humboldt-wi.github.io/blog/research/information_systems_1819/group5_han/
3. https://www.cs.cmu.edu/~./hovy/papers/16HLT-hierarchical-attention-networks.pdf
4. https://devblogs.microsoft.com/cse/2018/03/06/sequence-intent-classification/