Bias Detection in News.md

Bias Detection in News

Team 29:

- Shivang Gupta (2019101117)
- Zishan Kazi (2019111031)

Problem Description

Bias Detection in news articles. Detect sentence level and article level bias in news domain.

Aim of the project is to find biased news articles. Different news sources may have their own views towards the society, politics and other topics. Furthermore, they need to attract readers to make their businesses profitable. This frequently leads to the potentially harmful reporting style resulting in biased news. Therefore most of the news articles howsoever unbiased they claim are biased in some way. So the task in the project is to detect neutral point of view (NPOV) violations at sentence level and article level in news articles.

Dataset

Data for PAN at SemEval 2019 Task 4: Hyperpartisan News Detection: https://zenodo.org/record/1489920

About Dataset

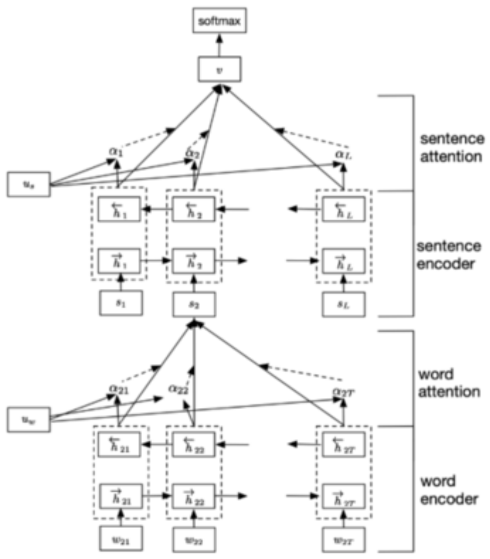
The data is split into multiple files. Articles are contained in files with names starting with "articles-". The ground-truth information is contained in file names starting with "ground-truth-".

The first part of the data (filename contains "bypublisher") is labeled by the overall bias of the publisher as provided by BuzzFeed journalists. It contains a total of **750,000 articles**, half of which (375,000) are hyperpartisan and half of which are not. The articles that are hyperpartisan, half of them (187,500) are on the left side of the political spectrum, half are on the right side.

This data is split into a **training set 80%** (600,000 articles) and a **validation set 20%** (150,000 articles), where no publisher that occurs in the training set also occurs in the validation set. Similarly, none of the publishers in those sets will occur in the test set.

The second part of the data contains only articles for which a consensus among the crowdsourcing workers existed. It contains a total of 645 articles. Of these, 238 (37%) are hyperpartisan and 407 (63%) are not, We will use a similar test set. Again, none of the publishers in this set will occur in the test set.

Overview



The big picture here is-:

- Words form sentences and sentences form documents.
- As we can see in this image we are representing a document as a single vector v, on which we apply softmax and then classify it.
- Reading the figure bottom to top:

  i. In the first layer we deal with words in a sentence.
  ii. In a sentence, for all the words we apply forward GRU and backward GRU and concatenate the output vectors from them. Then we linearly combine these states of the words to give the sentence in the above layer.
  iii. The same steps done for the words are done for the sentences in the second layer to get the document vector which is then further classified.

- What happens within a layer? Let's understand it for the first layer as both the layers do the same thing.

  i. After concatenating the output vectors from forward and backward GRU, for each word we get a vector.
  ii. We multiply these word vectors with the weight matrix, add bias term and apply an activation function `tanh`. For each word we get a new vector, with all these new vectors and u_w (context word), we find the softmax of each which gives us the `word attention` (α_{it}) or importance of word.
  iii. The word attention along with word vectors are linearly added to give a sentence. $s_{i} = \sum_{t}\alpha_{it}h_{it}$

Baseline

Hierarchical Attention Networks for Document Classification: https://www.cs.cmu.edu/~./hovy/papers/16HLT-hierarchical-attention-networks.pdf

We are using paper `Hierarchical Attention Networks for Document Classification` for implementation of baseline.

The basic idea of the model is -

The model has two distinctive characteristics -

1. It uses the hierarchical structure of documents.
2. It has two levels of attention mechanisms applied at the word- and sentence-level.

It consists of several parts: a word sequence encoder, a word-level attention layer, a sentence encoder and a sentence-level attention layer.

The basic idea of all these parts is -

- Encoder tries to get contextual word embedding of a sentence. Bi-directional LSTM or bi-directional GRU can be used for this.
- Word level attention layer can be used as an improvement, so that more weightage can be given to particular words that are important to the meaning of the sentence and aggregate the representation of those informative words to form a sentence vector. It can be implemented as -

$$
\begin{aligned}
u_{it} &= \tanh(W_w h_{it} + b_w) \\
\alpha_{it} &= \frac{\exp(u_{it}^\top u_w)}{\sum_t \exp(u_{it}^\top u_w)} \\
s_i &= \sum_t \alpha_{it} h_{it}.
\end{aligned}
$$

- Sentence encoder - Given the sentence vectors si, we can get a document vector in a similar way using a bidirectional GRU to encode the sentences:

$$
\begin{aligned}
\overrightarrow{h}_i &= \overrightarrow{\mathrm{GRU}}(s_i), i \in [1, L], \\
\overleftarrow{h}_i &= \overleftarrow{\mathrm{GRU}}(s_i), t \in [L, 1].
\end{aligned}
$$

- We use a sentence level context vector and use attention to get the importance of that sentence. It can be implemented in a similar way as word attention -

$$
\begin{aligned}
u_i &= \tanh(W_s h_i + b_s), \\
\alpha_i &= \frac{\exp(u_i^\top u_s)}{\sum_i \exp(u_i^\top u_s)}, \\
v &= \sum_i \alpha_i h_i,
\end{aligned}
$$

here v is the document vector that summarizes all the information of sentences in a document.

`So this the the basic implementation of this paper.We will also use additional inbuilt libraries like tensorflow and pytorch for training on our dataset.`

Baseline+

Possible Improvements -

- Cleaning of dataset by removing stop words like 'the' , 'is' etc.
- Replace bi-directional GRU with elmo or bert (for better contextual information).
- Fine-Tuning hyperparameters (like epochs, number of layers in neural net) to increase accuracy.