Document classification using Hierarchical Attention Networks

Dweepa Honnavalli 01FB16ECS138
Malaika Vijay 01FB16ECS189
Nandagopal N.V 01FB16ECS221

27 November 2019

1 Introduction

In this work we explore the use of word and sentence level attention using Bi-Directional Recurrent Neural Networks with GRU cells for multi-label classification of documents. In particular, we work on the problem of toxic comment classification over six classes of toxicity. The premise of this work is to use the intuition of relative word importance, and relative sentence importance to attend differently to different portions of a sentence, and document while performing the task of document classification.

We refer to the architecture put forth in the paper authored by Zichao Yang et al. [1] which is a Heirarchical Attention Network used for document classification.

2 The Need for Attention

All words in a sentence do not contribute to the overall meaning of the sentence equally. Some words are more telling of the sentence's meaning than others. For example, in a restaurant review - "the pasta was terrific! I would highly recommend the amazing food", the words "amazing" and"terrific" contribute significantly to the sentiment of the review. A human reader would implicitly weigh those words more heavily against others to decide the sentiment of the review. A similar reasoning of relative sentence importance is utilised to weigh the sentences that make up a document differently.

3 Dataset

We use the Wikipedia Toxic Comment Dataset that contains a 160,000 instances of comments on Wikipedia articles that are tagged as -

- Toxic
- Severe Toxic
- Obscene
- Threat
- Insult
- Identity Hate

Distribution of the classes are as shown in the table below. 1's represent that the review in question falls under that label of toxicity. 0's represent that the review is innocuous.

Distribution of Classes

Column	Number of 0s	Number of 1s
Toxic	144277	15294
Severe Toxic	157976	1595
Obscene	151122	8449
Threat	151694	478
Insult	151694	7877
Identity Hate	158166	1405

4 Methodology

4.1 Preprocessing

All reviews in the dataset are converted to lowercase, stripped of extra whitespace and non-ASCII characters. We do not remove stop words. Each review is decomposed into its component sentences, and each sentence is tokenized.

4.2 Construction of the Embedding Matrix

We consider a vocabulary size of 20,000 words, and extract the 20,000 most frequently occurring words to populate a word index. Each of these words is assigned a vector representation. We use the Glove pre-trained embeddings to initialise the embedding matrix. Each embedding spans 100 dimensions.

4.3 Model Architecture

The model under consideration in Fig 1 is the Hierarchical Attention Model. There are two levels of attention at play: Word Level and Sentence Level.
The Word level model consists of:

- An **Sentence Level Input Layer** that takes in words of a sentence one at a time.
- An **Embedding Layer** initialised with Glove embeddings.
- A **Bi-Directional RNN** with GRU cells.
- A **Word Level Attention** mechanism.

The Sentence level model consists of:

- An **Input Layer** that takes in an entire sentence.
- A **Bi-Directional RNN** with GRU cells.
- A **Sentence Level Attention** mechanism.

Dimensions of the output at each layer when passed through an input of 2 reviews, each review containing a maximum of 15 sentences and each sentence containing a maximum of 80 words is given in Fig 2.

4.3.1 Sentence Input Layer

This layer accepts instances of reviews. These reviews are formatted as sequences of word indices, where each sequence represents an index. Each input to the network is of dimension 15x80 i.e. (maximum number of sentences per review, maximum number of words per sentence). Each sentence of the review is then processed by the network one at a time, akin to a Keras TimeDistributed layer.

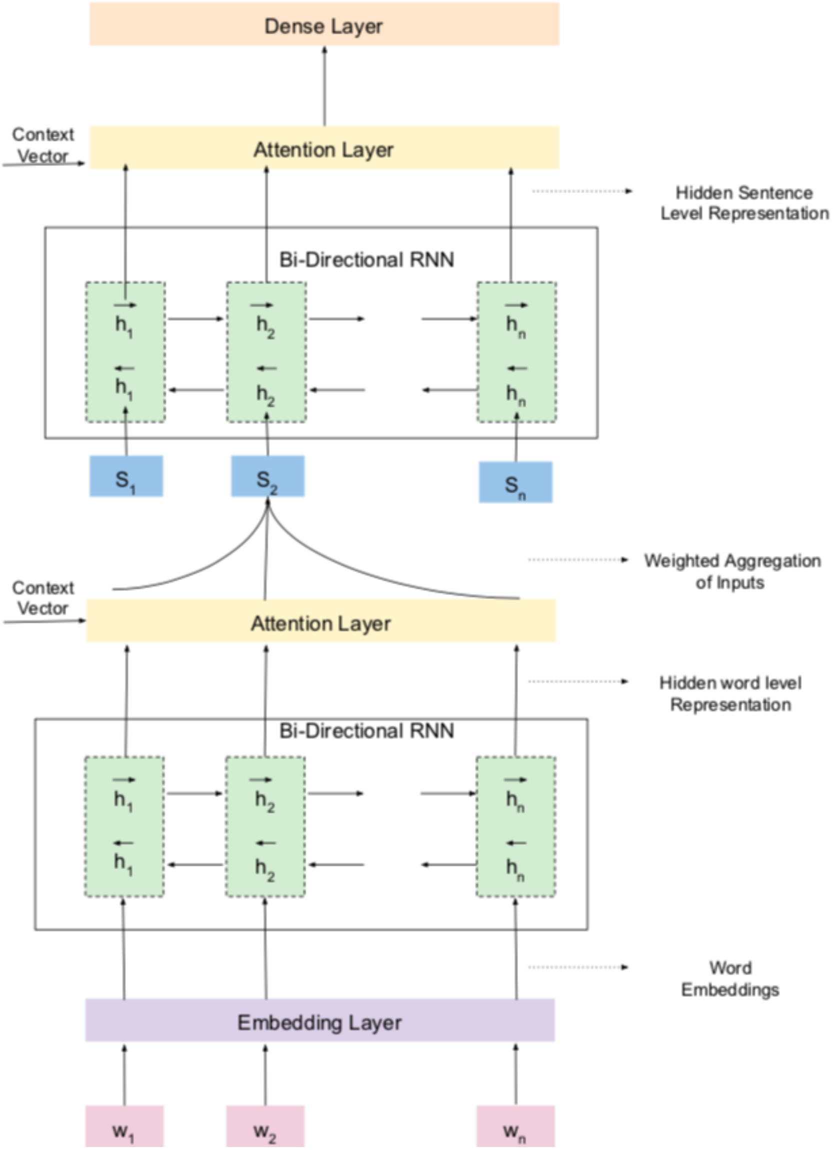


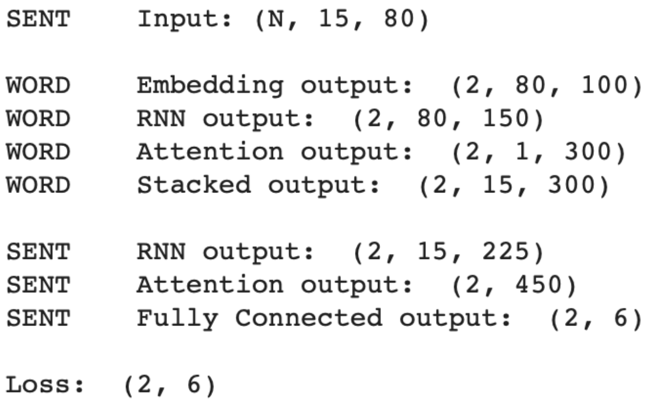
Figure 1: Architecture of the Network

4.3.2 Embedding Layer

For each sentence passed to the embedding layer, the embeddings of the tokens that make up the sentence are looked up from the embedding matrix. These tokens are replaced with their corresponding embeddings. The 1x80 dimension input is now converted into an 80x100 dimension output, where 100 is the dimension of each word embedding.

4.3.3 Word Level Bi-Directional RNN

Each sentence is processed independently, by passing each word of each sentence into a Bi-Directional RNN. Each of the networks (i.e. one that processes the input from word$_1$ to word$_n$ and another that processes the input from word$_n$ to word$_1$) use 150 GRU Cells. Each timestep corresponds to a single word being processed as input. The output from each GRU cell, which is a hidden representation of each word, is passed on to the word level attention mechanism. The 80x100 dimensional input



```
SENT    Input: (N, 15, 80)

WORD    Embedding output:  (2, 80, 100)
WORD    RNN output:  (2, 80, 150)
WORD    Attention output:  (2, 1, 300)
WORD    Stacked output:  (2, 15, 300)

SENT    RNN output:  (2, 15, 225)
SENT    Attention output:  (2, 450)
SENT    Fully Connected output:  (2, 6)

Loss:  (2, 6)
```

Figure 2: Output dimensions of each layer in the architecture

is now converted into two 80x150 dimensional output vectors, one for the forward network, and the other for the backward network.

GRU cells feature an update and reset gate. The update gate decides how much past information is retained, and how much new information is added. It is updated as follows -

$$z_t = \sigma\big(W_z x_t + U_z h_{t-1}) \tag{1}$$

where x_t is the input at time-step t, and h_{t-1} is the hidden state computed at the previous time-step.
The reset gate determines how much past information to discard. We compute

$$r_t = \sigma\big(W_r x_t + U_r h_{t-1}) \tag{2}$$

A new candidate hidden state is computed as -

$$\widetilde{h} = tanh(W_h x_t + r_t(U_h h_{t-1})) \tag{3}$$

Finally, the new hidden state is computed as -

$$h_t = z_t h_{t-1} + (1 - z_t)\widetilde{h}_t \tag{4}$$

4.3.4 Word Level Attention Mechanism

The representations of the words in the sentence output by the Bi-Directional RNN must now be assigned scores of importance.
Let $\overrightarrow{o_{it}}$ be $\overrightarrow{GRU}$ (x_{it}) i.e. the output of the GRU cell given word$_{\mathrm{t}}$ of sentence$_{\mathrm{i}}$, and $\overleftarrow{o_{it}}$ be $\overleftarrow{GRU}$ (x_{it}) i.e. the output of the GRU cell given word$_{\mathrm{t}}$ of sentence$_{\mathrm{i}}$ Finally o$_{\mathrm{it}}$ represents the concatenated outputs of the forward and backward networks. Attention scores are computed by a single perceptron that does the following computation -

$$u_{it} = tanh(W o_i + b) \tag{5}$$

The attention scores are computed as -

$$\alpha_{it} = \frac{exp(u_{it} u_w)}{\sum_t exp(u_{it} u_w)} \tag{6}$$

Where u_w is a universal context vector that is trained along with the rest of the network to give information on which word in the sentence is important.

Finally a new representation of the sentence is computed by summing over the weighted inputs to the attention layer, where the weights are the attention score.

4.3.5 Sentence Level Bi-Directional RNN

The representations of the sentences as an aggregation of the weighted words is then passed as input to another Bi-Directional RNN. This network serves the purpose of generating a contextual representation of each sentence in the network. These representations are then fed to another sentence level attention layer.

4.3.6 Sentence Level Attention

This layer assigns scores of importance to each sentence in the review. As in the word level attention mechanism, the scores are computed as a single perceptron. The math remains the same as in the word level attention mechanism.

4.3.7 Dense Layer

The final representation of the review as an aggregation of the vectorial representation of each sentence is then passed onto the final dense layer, with 6 neurons, one for each output class. We employ a sigmoid activation function in this layer.

4.4 Loss Function

We use binary cross entropy to estimate the loss. We employ mini-batch gradient descent. Binary cross entropy is calculated as follows:

$$H^p_{(q)} = \frac{1}{N}\sum_{i=1}^{N} y_i log(P_i) + (1 - y_i)(log(1 - P_i)) \tag{7}$$

5 Results

The model was trained on 2000 reviews in batches of 200. Results on 10,000 reviews are presented below

Train Accuracy	Validation Accuracy
96.3%	96.2%

References

[1] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. pages 1480–1489, 01 2016.