DATA MINING

Topic : Sentiment Analysis and Emojification

- Saprem Shah(201401107)

1. Problem Definition

The **problem** in **sentiment analysis** is classifying the polarity of a given **text** at the document,sentence, or feature/aspect level . whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive,negative, or neutral. It is popular topic in Text mining.

2. Motivation

- Sentiment analysis is extremely useful in social media monitoring as it allows us to gain an overview of the wider public opinion behind certain topics.

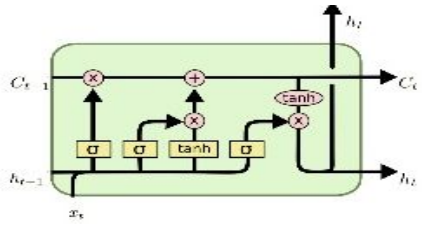
The applications of sentiment analysis are broad and powerful.

- ➔ Shifts in sentiment on social media have been shown to correlate with shifts in the stock market.
- ➔ The ability to extract insights from social data is a practice that is being widely adopted by organisations across the world.
- ➔ Adjust marketing strategy
- ➔ Develop better products and improve customer satisfaction.

3. Work

The author has present a neural network techniques to capture sentiment analysis in short text based on recurrent neural network LSTM and pre-trained word vectors word2vec. Experiments using the IMDB sentiment analysis dataset show that the method is competitive with state-of-the-art to previous reported results.
We can capture more information about the syntactic and semantic of word by using pre-trained word vectors. Hence, the proposed model has the prospective to overcome several flaws in traditional methods e.g., bag-of-words and n-gram models where order and information about word is vanished
Combination of convolutional neural network with recurrent layer is also direction worth to explore.

LSTMs (Modified RNN cell):



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma(W_o [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

Tohoku University, Sosuke

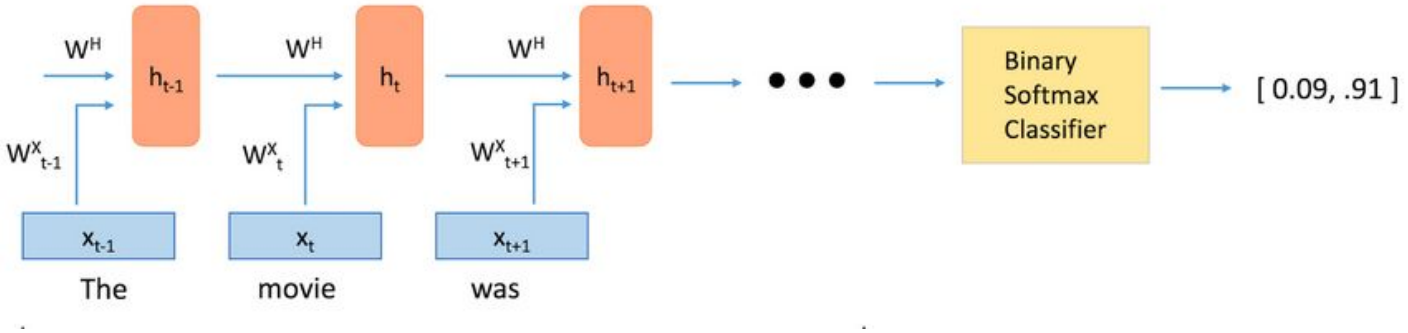
Long short-term memory (**LSTM**) units (or blocks) are a building unit for layers of a recurrent neural network (**RNN**).
LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

4. Exp work

Dataset:
The dataset is compiled from a collection of 50,000 reviews from IMDB on the condition there are no more than 30 reviews per movie. The numbers of positive and negative reviews are equal. Negative reviews have scores less or equal than 4 out of 10 while a positive review have score greater or equal than 7 out of 10. Neutral reviews are not included. The 50,000 reviews are divided evenly into the training and test set.

Project's main technique:
In this project I have implemented the algorithm for Sentiment Analysis for **IMDB movie dataset**-"**LSTMs using pre-trained GloVe vectors** " in Tensorflow.
Glove vectors are similar to word2Vec model of Google but on smaller scale.

Model:



Steps:
1) Training a word vector generation model (such as Word2Vec) or loading pre-trained word vectors
2) Creating an ID's matrix for our training set (We'll discuss this a bit later)
3) RNN (With LSTM units) model creation
4) Training
5) Testing

Word2Vec
Words can't be used as data since operations like dot product, backpropagation can't be applied. Since we have input as sentence, we need to convert it into matrix of scalar values.
The vector representation of a word is known as a word embedding. In order to create these word embeddings, we'll use a model that's commonly referred to as "Word2Vec"(or GloVe as used in the project). Words with similar contexts will be placed close together in the vector space.

HyperParameters (taken while training data):
- Learning Rate - 0.001
- Optimizer - Adam optimizer
- Number of LSTM units - 64
- Batch size - 24
- Iterations - 100,000

Test Results:

```
secondInputText = "That movie was the best one I have ever seen."
secondInputMatrix = getSentenceMatrix(secondInputText)
predictedSentiment = sess.run(prediction, {input_data: secondInputMatrix})[0]
if (predictedSentiment[0] > predictedSentiment[1]):
    print "Positive Sentiment"
else:
    print "Negative Sentiment"
```

```
Positive Sentiment
```

```
thirdInputText = "Justic League was the not as good as expected."
thirdInputMatrix = getSentenceMatrix(thirdInputText)
predictedSentiment = sess.run(prediction, {input_data: thirdInputMatrix})[0]
if (predictedSentiment[0] > predictedSentiment[1]):
    print "Positive Sentiment"
else:
    print "Negative Sentiment"
```

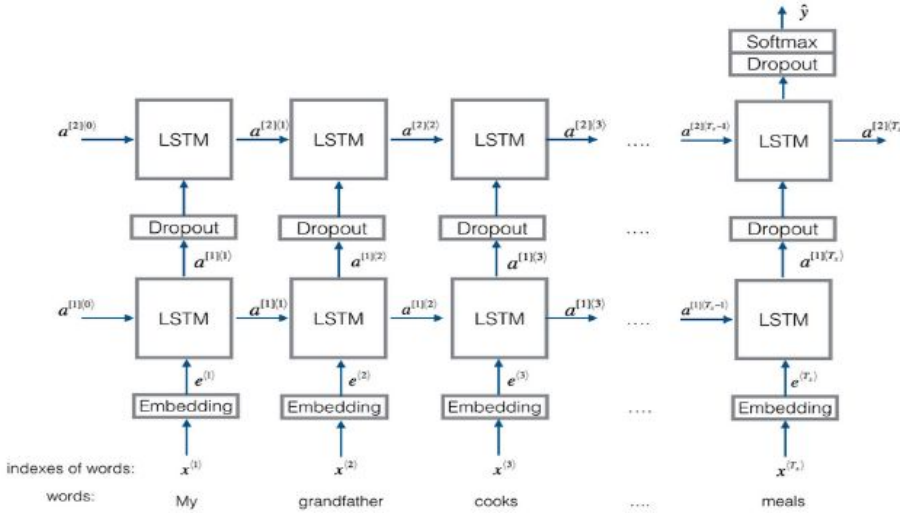
```
Negative Sentiment
```

Techniques tried and their accuracy:
Further using in-build IMDB dataset in Keras I have experimented various models such as **MLP** (86.86%), **Simple LSTM** (87.22%), **LSTM with Dropout** (84.08%), **LSTM with Recurrent Dropout**(78.94%) and **LSTM with CNN** (88.28%) and compare the results.

5. Further experiment using Sentiment analysis

Have you ever wanted to make your text messages more expressive?
Emojify : An emojifier based on Sentiment Analysis (of small dataset) which based on the input sentence gives us it's appropriate emoji from predefined 5 emojis taken into consideration.
Eg: Input : Congratulations on the promotion!
Output: Congratulations on the promotion! 😄

Model:



Test Results:

```
]: x_test = np.array(['I like to eat '])
   X_test_indices = sentences_to_indices(x_test, word_to_index, maxLen)
   print(x_test[0] +' '+  label_to_emoji(np.argmax(model.predict(X_test_indices))))

   I like to eat
```

6. Conclusion

The LSTMs based model by using pre-trained Word2Vec (or GloVe) - word vectors for constructing the input sentence into its vector representation performs quite well for the given (and other) datasets.

Previous works on Sentiment Analysis have been done using (neural as well as non-neural network models) Naive-Bayes, SVM, Recurrent Tensor Networks. Further new methods based on Attention based Neural Networks, DeepForest and FastText have been tried out. Bidirectional RNN is also a great method to try with great results.

The results of Naive Bayes and SVM has been very competitive to Neural Network based models. However, it is important to note that with different dataset different models give different results. If much larger dataset such as Yelp is used Neural Network based models clearly out-performs the Naive-Bayes and SVM approach.

Here, I observed that the results of RNN out-performs MLP model (although bit comparable) with LSTMs + CNN based model giving accuracy of 88.28% which can be increased further by tweaking the hyperparameters.

7. References

[1]https://www.researchgate.net/publication/320086745_SENTIMENT_ANALYSIS_WITH_RECURRENT_NEURAL_NETWORK_AND_UNSUPERVISED_NEURAL_LANGUAGE_MODEL

[2]https://blog.paralleldots.com/data-science/breakthrough-research-papers-and-models-for-sentiment-analysis/

[3]https://ieeexplore.ieee.org/document/8001979/

[4]http://colah.github.io/posts/2015-08-Understanding-LSTMs/