Sentiment Analysis

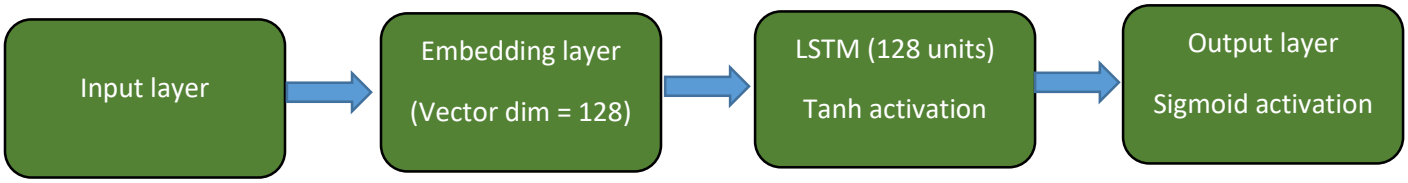
Submitted by : Preetam Keshari Nahak
116CS0205

1. Introduction

Sentiment analysis is an application of natural language processing in which we try to figure out emotions such as positive, negative or neutral nature from the textual data using various text analysis and mining techniques. Sentiment analysis helps business to know about what the customer thinks about their products, services by analyzing their reviews, comments from various platform about the product.

2. Thoery

A typical sentiment analysis solution can be implemented using basic machine learning algorithms. With basic classification models, we can address such problems and find a workable solution. However generic classifications models though perform quite good, they fail in some cases where there's need of understanding the context of the texts etc. In that case, deep learning models play handy role. Basically, RNN with LSTM cells are appropriate to solve such problems. Below we represent our model architecture to perform sentiment analysis problem.



[Fig 1. Proposed Model]

Now let's discuss briefly various concepts related to this architecture

- Word embeddings

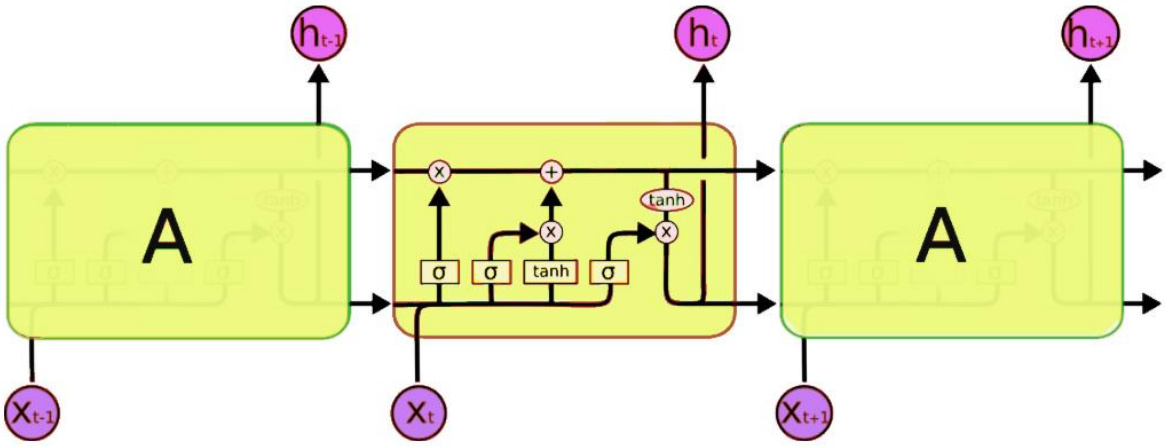
Objective of word embedding is to represent words as vector in a desired dimension space say D, which will preserve the similarities among words with similar context. There are two types of word2vec techniques COBW (Common bag of words model) and skip-gram model. Basically, word embeddings will be useful in order predict similar or closer words for a given word.

- LSTM

RNN works perfectly, when we deal with short term dependency of the words in a sentence. When it comes to long term dependency RNN fails to preserve information due to vanishing gradient problem. E.g Consider the following sentence :

"Ram, who was a great king of ayodhya and son of dasrath went in exile for 14 years."

Here the word "Ram" has a long term dependency with the phrase "went in exile for 14 years". So while training our RNN, our gradients vanish slowly as we move towards the expected subject "Ram". So in order to address such problems LSTM cells are used. The basic structure of a LSTM cell looks like :



[Fig 2. Internal structure of a LSTM cell]

A LSTM cell has basically 3 types of gates in it forget gate, output gate and update gate. Various equations governing these gates are as follows.

$$ft = (Wfxt + Ufht{-}1 + bf)$$

$$it = (Wi\,xt + Uiht{-}1 + bi\,)$$

$$ot = (Woxt + Uoht{-}1 + bo)$$

$$ct\,' = \sigma h(Wcxt + Ucht{-}1 + bc\,)$$

$$ct = ft * ct{-}1 + it * ct'$$

$$ht = ot * \sigma h\,(ct\,)$$

where,

xt : input vector to the LSTM unit

ft : forget gate's activation vector

it : input/update gate's activation vector

ot: output gate's activation vector

ht : hidden state vector also known as output vector of the LSTM unit

ct : cell state vector

ct' : cell input activation vector

W, U: Weight matrices and bias parameters for input and recurrent connections

- Input Layer :

Each sentence is given as an input to the Embedding layer which gives a sequence of embedding vectors, representing the words in the sentence.

- Output Layer

We can use sigmoid activation function for binary classification and softmax for multiclass classification.

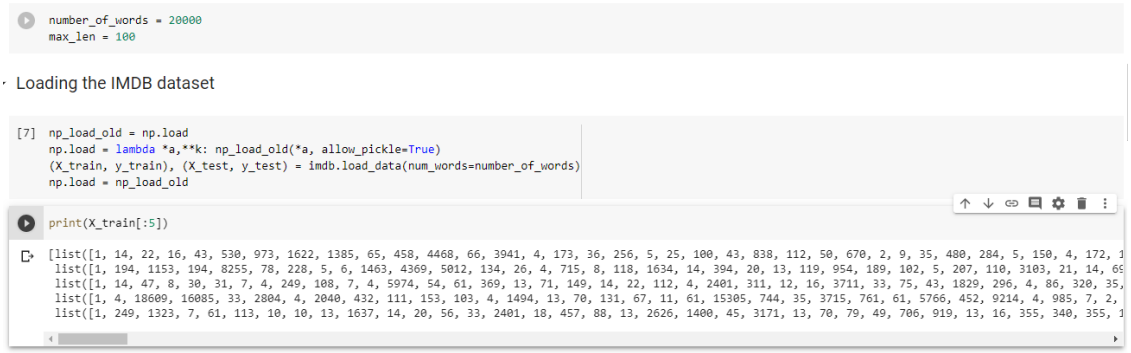
3. Methodology

3.1 : Dataset

We will use movie review dataset by IMDB which consists of 50k records with 25 positive ad 25k negative movie reviews. The data comes with keras library by default and it's already pre-processed so that our further computation becomes easier.

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive
5	Probably my all-time favorite movie, a story o...	positive
6	I sure would like to see a resurrection of a u...	positive
7	This show was an amazing, fresh & innovative i...	negative
8	Encouraged by the positive comments about this...	negative
9	If you like original gut wrenching laughter yo...	positive

[Fig 3. Actual Reviews]



[Fig 4. Preprocessed IMDB data]

3.2 Model

As mentioned in Fig 1, our model will have basically one embedding layer, LSTM layer and output layer. Embedding layer will prepare word embeddings of dimension 128. Input of reviews of maximum 100 words are fed to the embedding layer. Reviews with below 100 words are padded in order to prepare uniform input data for the embedding layer. The embedding layer generates 100 word embeddings of dimension 128 for each input movie review. This embeddings are fed to the LSTM layer and similarly to the output layer further.

```
[ ] model.summary()

Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 100, 128)          2560000
_________________________________________________________________
unified_lstm (UnifiedLSTM)   (None, 128)               131584
_________________________________________________________________
dense (Dense)                (None, 1)                 129
=================================================================
Total params: 2,691,713
Trainable params: 2,691,713
Non-trainable params: 0
_________________________________________________________________
```

[Fig 5. Model Summary]

3.3 Training and testing the model

We trained the model with three epochs with a batch size of 128 for 25000 reviews with 'rmsprop' optimizer and the trained model is tested with 25000 reviews. As the model show over fitting behavior after three epochs, we finalized to train the model with three epochs.

▾ Training the model

```
model.fit(X_train, y_train, epochs=3, batch_size=128)

Epoch 1/3
196/196 [==============================] - 6s 29ms/step - loss: 0.4572 - accuracy: 0.7899
Epoch 2/3
196/196 [==============================] - 6s 28ms/step - loss: 0.2847 - accuracy: 0.8858
Epoch 3/3
196/196 [==============================] - 5s 28ms/step - loss: 0.2307 - accuracy: 0.9100
<tensorflow.python.keras.callbacks.History at 0x7fd9286357b8>
```

[Fig 6. Training the model]

▾ Evaluating the model

```
[13] test_loss, test_acurracy = model.evaluate(X_test, y_test)

782/782 [==============================] - 4s 5ms/step - loss: 0.5195 - accuracy: 0.8264

print("Test accuracy: {}".format(test_acurracy))

Test accuracy: 0.8263999819755554
```

[Fig 7. Testing the model]

3.4 Results and accuracy

After testing was performed, it gave an accuracy of 0.8263999819755554

Conclusion

Here we demonstrated an approach to perform sentiment analysis with IMDB dataset which gave an accuracy of 82 percent (approx.) which can be improved with a better architecture and text preprocessing techniques.