Abstract

Sentiment analysis is defined as finding out sentiment of the person in the context.
Medium of sentiment could be any thing like text written by person and facial expression and a lot more.
In this report , I have considered text data as carrier of sentiment.

DataSet

I have used Stanford movie review dataset.
In which each user have expressed their sentiment in the form of the text review for the movie they have watched.
There are two classes of sentiment - negative and positive.

https://ai.stanford.edu/~amaas/data/sentiment/

Approach Overview

1. non sequence based(aka “Bag of Word”)
2. Sequence based

Bag of Word

Preprocessing steps -

1. tokenization (“this is project” = “this” , “is” , “project”).
2. Stop word removal stop are those words who are very common in written text . So,they bear no separability power to distinguish between different classes of texts. Example - “who” , “as” etc .
3. Lemmatization -
    a. In lemmatization we remove inflections from the root word and get the root word
    b. Examples - (walking , walk - walk , better - good) .

Feature selection -

For feature selection I have used two different approaches indepently . namely Tf-IDF and MI gain .

TF-IDF

In this we assign a score to each token(Tf-IDF) .higher the score is higher the the probability that given token is considered for final feature for classification.

IDF is defined as the inverse document frequency calculated as follows = log((files having token)/ total number of files).

TF is the term frequency of the term in the docs .

TF-IDF score is a product of above both.

MI score -
In this we calculate the MI score of each token and arrange them in increasing order and select the top few tokens that are featured.

Vectorization of Text -

For classification we have to vectorize text into vectors.
In this we have created a 500 unit length long feature vector. We have two vectors for each review, one the basis of MI score and other on the basis of Tf-IDF.

Each entry in the vector is the frequency of that token in review text.

Model -

I have used MLP .
First layer is 500 units with sigmoid activation.
Second layer is 400 units with sigmoid activation.
Third layer is 300 units with sigmoid activation.
Fourth layer is 2 units with softmax.

Optimizer is ADAM.

Result -

In all three datasets train , val , test result is 0.5 .

Conclusion -
Vectors are highly sparse and vectors don't have any semantic similarity.
So we are not getting any reasonable result and no learning at all.

Recurrent Network

Word embedding used are word2vec trained by negative sub sampling

Few commonalities among all artitecture -
Recurrent layer's activation is tanh , dropout = 0.2
Dense layer is sigmoid .
Optimizer - ADAM
Training batch rule - Batch stochastic Gradient descent
First accuracy is for train ,second for validation and last one is for test set.

Shallow Vanilla RNN -

Recurrent layer size is 64
Output layer is of 1 . with a sigmoidal classifier.
Actuaries .801 , 0.783 , 0.7901
Summary - Low representation power of shallow network

Deep Vanilla RNN -
R1 - 64 , 0.2
R2 - 64 , 0.2
D3 - 32
D4 - 1
A - .79 , 0.77 , 0.7791
Summary -
Suffering from vanishing gradient descent , resulted in slow learning .
Hidden states are rewritten every with little power to transfer hidden states into far future runs.
Cannot handle long-term dependency. Resulting in poor generalization.

Shallow RNN with GRU layers

G1 - 64,0.2
D2 - 1
A - 0.757 , 0.7544 , 0.756
Summary -
GRU units in recurrent layers try to handle information lost over longer sequences by introducing a forget gate.

Deep RNN with GRU layers
G1 - 64
G2- 64
D3 -32
D4 - 1
A - 0.7946 , 0.947 , 0.7923

Summary
it is more complex and tries to capture more information.

Shallow LSTM -
L1- 64
D1 - 32
D3 - 1
A - 0.85,0.86 , 0.85

Shllow LSTM works best.it can handle long term depenecy , vanishing gradient descent . memory lost over time.