Twitter Sentiment Analysis

Results & Observations

Sahil Jindal	Abhishek Gupta	Rohan Gyani	Mridul Ravi Jain
110020043	110040067	110040001	110040083

Assignment Specifications

- Give a neural network for recognizing the sentiments of tweets.
- Download tweets, do feature engineering on them.
- A naïve feature vector is the set of words in the tweets.
- Collect all the words in the tweets, sort them, remove duplicates.
- Each tweet will be represented by a 1/0 vector depending on the presence/absence of the word in the tweet.

Feature Engineering On Tweets

- We need to construct a feature vector large enough so that it can capture the essential sentiment of a tweet.

- A naïve feature vector is the set of all words in tweets. This has a large size.

- If our feature vector is too large, learning take long time and network learns unessential features which are not important for a sentiment.

- So, we consider only those words of the tweets that may constitute a sentiment and remove everything else.

Decreasing the Feature Vector Size

- Initially, total number of words in all tweets = 24680
- Convert all the words to lower case.
- Remove all Numbers.
- Remove special characters at start and at end of words such as

  ? , . [ ] ; _ ! = | - & ‘ # URLs
- Don't put certain common words, symbols, numbers in the vector of words such as:

  the, is, a, an, this, that, are my with, I’m, he, she, our, were, can, do, had….
- Finally we remove those words that occur only once in the entire set of words in the tweets because they are highly unlikely to contribute to a sentiment
- This results in a lot of unnecessary words being removed from our feature vector and finally we get a feature vector whose size is 1958

Methodology

- We used a neural network with 3 output neurons and 1958 input neurons. (No hidden layer)
- We performed 5-fold cross validation on the given tweet corpus.
- We divided the tweets into 5 partitions & used 4 partitions for training the neural network and the remaining one for measuring accuracy
- Coded in C++, it took approx. 6 min to converge to a solution for training and testing on a single partition.
- Learning rate was kept at 0.5, Momentum Factor = 0.

Results Obtained (5-fold)

- Set 1 : 56.25%
- Set 2 : 47.25%
- Set 3 : 48%
- Set 4 : 53.25%
- Set 5 : 55.8603%

- Here, Set ‘n’ corresponds to nth partition for testing and remaining for training.