# Assignment 2: Sentiment Classification

Abhisek Panda
150026

March 2018

## 1 How to run the code?

python train.py –algo=A –rep=B –use_weights=C

where A can be anything from (NB, LR, SVM, MLP, LSTM) and B can be anything from (BBoW, NTF, tfidf, avg_W2V, avg_GLoVE, doc_vec, sen_vec). Set use_weights=True for term frequency weighted average of word2vec or glove vectors.

Another utility script, runner.sh is also provided which runs all possible combinations and saves output to a file named output_data.txt; however please note that it will take a lot of time to complete execution.

## 2 Implementation Notes

The following external libraries were used for the assignment:

- nltk and gensim for word processing
- scikit-learn for implementing all algorithms except LSTM
- Keras with tensorflow backend for LSTMs
- Other standard libraries like numpy

All the models except LSTM were trained on CPU. The LSTM was trained on the GPU servers provided by IITK. Representations were saved onto the file-system to avoid repeated computational intensive calculations.

## 3 Note on Hyperparameters

Most of the hyperparameters were set to default values offered by the libraries. Some of the custom set parameters are:

- In feed forward neural network, 2 hidden layers were used, with 200 and 64 hidden units.

- In LSTM, 128 LSTM units were used in the recurrent layer. Also, dropout was set to 0.2 to prevent over-fitting.
- Word2Vec and GLoVE embedding sizes were fixed to 300.
- The learnt embeddings for doc2vec and sentence2vec were 100 dimensional(sizes kept low for smaller training time). The model was trained for 10 epochs to obtain the embeddings. Higher accuracy is expected if trained for more time.

# 4 Results

| | Naive Bayes | Logistic Regression | SVM | Feed Forward NN | LSTM |
|---|---|---|---|---|---|
| BBoW | 83.44 | **86.97** | 84.86 | 85.81 | - |
| Normalized BoW | 84.89 | 80.68 | 85.12 | **85.80** | - |
| tfidf word vectors | 83.33 | **88.44** | 87.34 | 84.93 | - |
| Word2Vec | NA | 80.93 | 83.74 | 83.96 | **84.62** |
| Weighted Word2Vec | NA | **83.04** | 80.86 | 80.99 | - |
| GLoVE | NA | 82.19 | 82.26 | 82.32 | **84.91** |
| Weighted GLoVE | NA | **80.67** | 80.51 | 79.50 | - |
| Sentence Vectors | NA | 59.14 | 59.52 | **60.36** | - |
| Document Vectors | NA | 61.86 | 61.93 | **64.55** | - |

Table 1: The best algorithms for a fixed representation are highlighted in bold. - means that the combination was not tried in this work.

# 5 Observations:

- Logistic regression performed best in most of the cases, although the accuracies were quite close for all algorithms. This may be attributed to an easy decision boundary between the positive and negative labels.
- LSTMs, which are well known for boosting accuracy in NLP related tasks do not offer any major accuracy boost, although they certainly perform well. This may be due to the fact that the training corpus was quite small and reviews also were mostly very brief.
- Document vectors and sentence vectors give very poor performance compared to other representations. Although they were trained for less number of epochs due to computational constraints, small training size is also responsible for this reduction in performance.