# Sentiment Analysis of Short Text Content

**Kruthika Hassan**
Department of Applied Mathematics
University of Washington
Seattle, WA 98105
kruthika@uw.edu

## Abstract

In this paper, an honest attempt is made at understanding the effectiveness of Distributional Semantic Models, Recurrent Neural Networks and Long Sort Term Memory units in analyzing the semantics of short text content. The neural network is trained on a large corpus of twitter samples to correctly predict the sentiment and perform a multinomial classification. A comparative analysis of using a distributional semantic model and not using one emphasizes the role played by it.

## 1 Introduction and Overview

Semantics is the study of the meaning of words, constructions, and utterances. We can divide semantics into parts, the study of meaning of individual words (or *lexical semantics* and the study of how meanings of individual words are combined into the meaning of sentences (or even larger units) [1]. In this paper, the emphasis is on latter. With such an understanding of semantics, analyzing sentiment or emotional tone behind a piece of text or paragraph becomes apparent. For humans, this task is a toy example. But, to make it machine doable, a step by step algorithmic procedure needs to be adopted.

A first logical step involves obtaining a mathematical model for the semantics. This is effectively captured by the distributional semantics, which is the study of statistical patterns of human word usage to extract meaning from the text, which can be sentences, paragraphs or even documents. Distributional semantic models (DSM) are computational models that build contextual semantic representation from corpus data. The DSM has two important facets to it:

- Distributions are vectors in a multidimensional semantic space, i.e. objects with magnitude and direction.
- Semantic space has dimensions which correspond to possible contexts as gathered from a given corpus

The vectors thus obtained are known as Embeddings. Though there are different DSM techniques, the Global Vector embedding (GloVe) method is explored in this paper.

Having successfully obtained vector embeddings for given corpus, the next challenge is to use a classifier to successfully learn the sentiment behind text. Though there are several classification algorithms, in this paper, the focus is on Reccurent Neural Networks (RNN) and Long Short Term Memory (LSTM) units.

Rest of the paper is organized as follows; Section 2 sets the stage through problem definition and also a brief explanation about theory underlying the solution; Section 3 elucidates the algorithmic approach taken to construct the solution; Section **??**enumerates the experiment and results; Section 5 is on a short discussion and conclusion.

## 2 Problem Setup

A large twitter corpus, consisting of around $100,000$ tweets, where each tweet is labeled positive, negative, neutral or irrelevant, based on the sentiment towards a topic, is considered. The topics are a bunch of companies like Apple, Microsoft, Google, Amazon, etc. The problem on a bird's-eye level involves multinomial classification of the tweets based on the sentiment they hold towards a particular topic. It can be further broken down into three important sub problems:

1. Text pre-processing.
2. Obtaining vector embeddings for text corpus using GloVe method.
3. Training an RNN and LSTM for a classifier.

### 2.1 Theory

Before delving into solution of the problem, a short theory on GloVe and RNN,LSTM helps in better understanding of the problem and also the route considered to solve it.

#### 2.1.1 GloVe

We have previously understood in section 1, how morphemes, that is the smallest morphological unit can be represented as a vector using DSM. Now, assuming words form the smallest meaningful units in our corpus, it is imperative to know how similar they are in terms of meaning. Though Euclidean distance seems a logical choice for this, characterizing similarity between two words based on a single scalar may not be that effective. Words have more intricate relationships than can be represented by a single scalar value. A natural and simple candidate for an enlarged set of discriminative numbers is the vector difference between the two word vectors. GloVe is designed in order that such vector differences capture as much as possible the meaning specified by the juxtaposition of two words.

GloVe is essentially a log-bilinear model with a weighted least-squares objective. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. Owing to the fact that the logarithm of a ratio equals the difference of logarithms, this objective associates (the logarithm of) ratios of co-occurrence probabilities with vector differences in the word vector space. Because these ratios can encode some form of meaning, this information gets encoded as vector differences as well [2].

#### 2.1.2 RNN and LSTM

Long Short Term Memory Units are modules that you can place inside of reucrrent neural entworks. At a high level, they make sure that the hidden state vector is able to encapsulate information about long term dependencies in the text.

## 3 Algorithm Flow

The problem setup explained in Section 2 is tackled as three sub problems and Figure 1 shows a diagrammatic representation of algorithmic flow.

### 3.1 Text Preprocessing

A snapshot of the raw twitter corpus is shown at upper right corner in Figure 1. An important point to be considered when analyzing sentiment behind a piece of text is to transform emoticons, capital letters, characters that convey emotions into meaningful features. For example,

- Characters :-) , :-( is transformed into tags <happy face>, <sad face> respectively.
- If the text contains all caps "WONDERFUL", it may denote a strong sentiment towards a topic. Replacing it with a tag such as <all caps> would help in extracting information.

Once the text is preprocessed by converting meaningful characters into features, the data is ready to be broken into smaller units or words called "tokens".

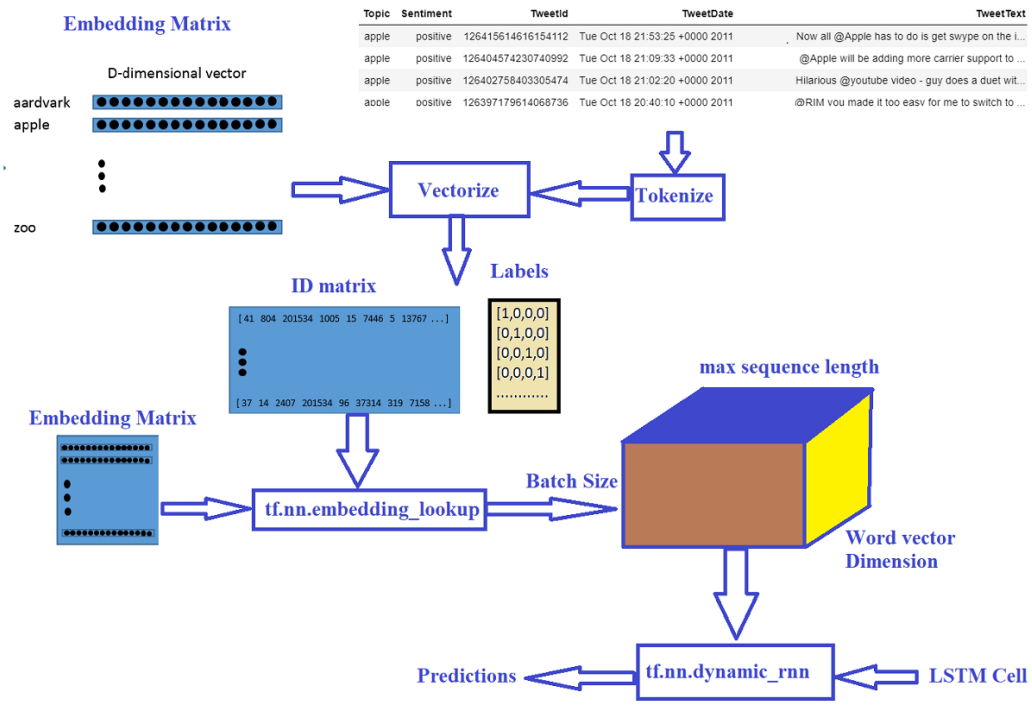


Figure 1: Diagrammatic representation of algorithmic flow

### 3.2 Vector Embeddings

The raw data is tokenized and ready to be represented through vector embeddings. With various advantages attached to GloVe representation, as described in Subsection 2.1.1 of Section 2.1, each word or token in the corpus is represented by a 25 dimensional vector embedding matrix obtained using GloVe. A representation of the embedding matrix is shown in Figure 1, in the upper left corner. Now, the vectorized ID matrix is created, wherein each ID is pointer to the respective word vector in the embedding matrix. A one-hot encoding of the labels is also constructed.

### 3.3 Classification

The classification task is performed by an RNN and LSTM, which is trained in batches. The step by step procedure is explained as follows:

1. Decide upon the maximum sequence length of words to be given to RNN at once. That is to say, observe maximum length of tweets and normalize by padding zeros so that all tweets have same length.

2. An embedding lookup function (tf.nn.embeddinglookup) is used to decode the ID matrix of words and produce a 3-D tensor consisting of batch size, maximum sequence length and word-vector dimension.

3. Now that we have the data in the format we want, the next step is to call the tf.nn.rnncell.BasicLSTMCell function. This function takes in an integer for the number of LSTM units, a hyperparameter that will take some tuning to figure out the optimal value. LSTM cell is then wrapped in a dropout layer to help prevent the network from overfitting.

4. Finally, both the LSTM cell and the 3-D tensor of input data into a function called tf.nn.dynamicrnn. This function is in charge of unrolling the whole network and creating a pathway for the data to flow through the RNN graph.

5. The first output of RNN is the last hidden state vector. This vector needs to be projected onto 2-D space by reshaping. Then the vector is multiplied by weights and bias is added to obtain predictions.
6. A softmax layer on top of the predictions gives the predictions in terms of probability of each of the labels, and correct prediction is obtained by comparing the maximum value with reference labels.
7. A cross-entropy loss layer gives the loss.

# 4 Experiments

Three experiments were conducted with respect to obtaining vector embeddings.

## 4.1 Word embeddings using random vectors

In this experiment, no DSM model such as GloVe was used. Instead random vectors were used to represent the tokens. The results (in percentage) are tabulated in Table **??**, with accuracy and loss plots shown in Figure 2.

Table 1: Results for word embeddings using random vectors

| | Training | Validation | Test |
|---|---|---|---|
| Accuracy | 50.13 | 49.76 | 49.94 |
| Precision | 50.00 | 49.99 | 49.98 |
| Recall | 0.44 | 0.44 | 0.44 |
| F1 Score | 0.46 | 0.46 | 0.46 |

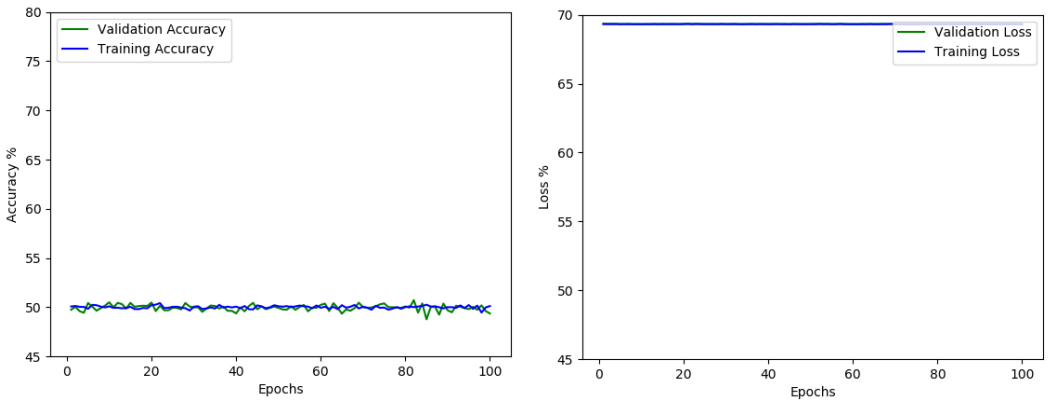


Figure 2: Accuracy and loss plots for random vector embeddings

## 4.2 Word embeddings using pretrained GloVe matrix

In this experiment, DSM model, GloVe was used. A pretrained embedding matrix contained vector representations for tokens in our corpus. The results (in percentage) are tabulated in Table **??**, with accuracy and loss plots shown in Figure 3.

## 4.3 Word embeddings using GloVe

In this experiment, GloVe embeddings were obtained exclusively for the corpus to be analyzed. The results (in percentage) are tabulated in Table 3, with accuracy and loss plots shown in Figure 4.

Table 2: Results for pretrained GloVe embeddings

| | Training | Validation | Test |
|---|---|---|---|
| Accuracy | 60.75 | 59.46 | 59.71 |
| Precision | 61.19 | 61.14 | 61.19 |
| Recall | 57.87 | 58.04 | 57.82 |
| F1 Score | 59.48 | 59.55 | 59.46 |

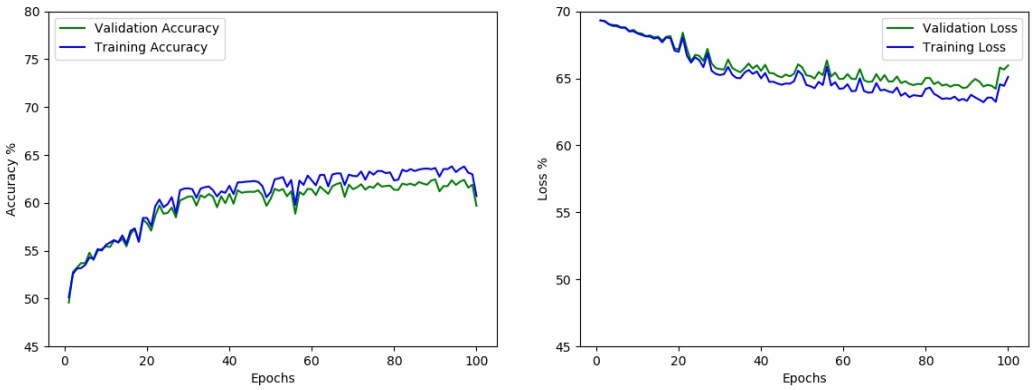


Figure 3: Accuracy and loss plots for pretrained GloVe vector embeddings

Table 3: Results for vector embeddings using GloVe

| | Training | Validation | Test |
|---|---|---|---|
| Accuracy | 76.35 | 75.67 | 75.18 |
| Precision | 72.86 | 72.84 | 72.87 |
| Recall | 72.96 | 72.91 | 72.97 |
| F1 Score | 72.91 | 72.88 | 72.92 |

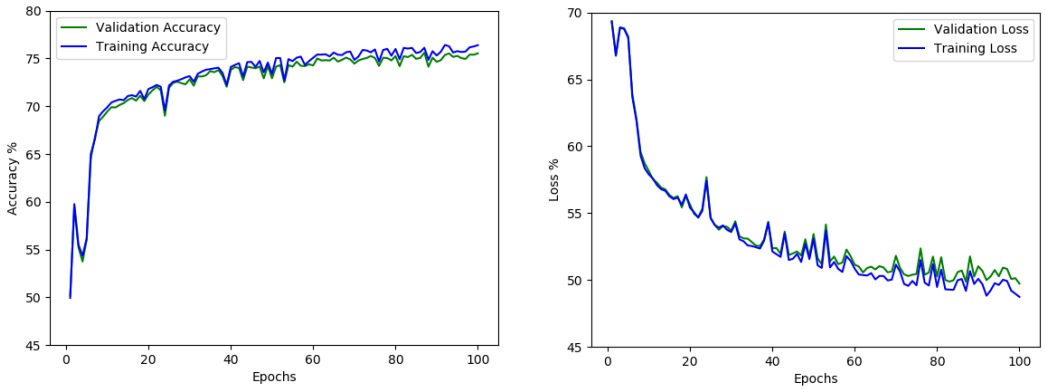


Figure 4: Accuracy and loss plots for GloVe embeddings

### 4.4 Hyperparameter tuning

Choosing the right values for hyperparameters is a crucial part of training deep neural networks effectively. The training loss curves can vary with the choice of optimizer (Adam, Adadelta, SGD, etc), learning rate, and network architecture. With RNNs and LSTMs in particular, some other important factors include the number of LSTM units and the size of the word vectors. After several

tunings, the current results were obtained with 64 LSTM units, learning rate of 0.01, using an Adam optimizer, and word vector size of 25.

## 5 Observations and Conclusion

With several experiments and tuning of the hyperparameters, the conclusive points about this project can be enumerated as below:

1. Firstly, the importance of distributional semantic models such as GloVe is understood when we compare the results of random vector embeddings GloVe embeddings. It clearly emphasizes that, representation of words with respect to the context is necessary to capture meaning from the text.
2. Secondly, using a pretrained matrix consisting of GloVe embeddings, the results show a positive trend. Though, a lack of representation for all the tokens in our vocabulary hampers the performance of RNN.
3. Lastly, having GloVe embeddings exclusively for our vocabulary results in a sharp increase in performance of the RNN, though there is a cost in terms of training time.
4. It was also observed that increasing LSTM units, increased the performance, though at the cost of computational time.

We can conclude that, RNN and LSTM's are promising learning machines with respect to sentiment analysis in NLP, because of their long-term dependencies and capability to learn temporal aspect of data. And also, distributional semantic models are excellent techniques to approach problems in NLP.

**Acknowledgments**

I extend my heartfelt gratitude to Dr. Hannaneh Hajishirzi, for providing valuable inputs and guidance throughout the course including the project; and also, to my husband for everlasting encouragement and support.

## References

[1] Manning, Christopher D & Schütze, Hinrich (1999) Foundations of statistical natural language processing, Cambridge, MA: MIT Press.

[2] Jeffrey Pennington & Richard Socher & Christopher D. Manning (2014) *Empirical Methods in Natural Language Processing (EMNLP).*

[3] Perform Sentiment Analysis with LSTM's, using TensorFlow, O'Reilly Magazine.