Deep Learning for Sentiment Analysis of Movie Reviews

Akilesh B, CS13B1042

Indian Institute of Technology, Hyderabad

November 4, 2016

Introduction

- Sentiment analysis is the automated extraction of writers attitude from the text and one of the major challenges in Natural Language Processing.
- Traditional approaches involve building a lexicon (vocabulary) of words with positive and negative polarities, and identifying the attitude of the author by comparing words in the text with the lexicon. (Ref: this)

Related work

- In [1], the algorithm consists of tokenization of the text, feature extraction and classification using different classifiers such as Naive Bayes, MaxEntropy or SVM
- Supervised (Ref: [2]) and Unsupervised (Ref: [3]) approaches for building high quality lexicons have been explored.

Need for Deep Learning in Sentiment Analysis

- The above traditional approaches fail to extract structural relations in language. It will be quite challenging for a machine to understand sarcasm in a review.
- The classical approaches to sentiment analysis and NLP are heavily based on engineered features, but it is very difficult to hand-craft the best set of features.
- Deep Learning models do not need to be provided with pre-defined features hand-picked by an engineer, but they can learn sophisticated features from the dataset by themselves.

Our work

- In this project, we explore the performance of different deep learning architectures for semantic analysis of movie reviews.
- Firstly, a Naive Bayes baseline classifier is implemented on the dataset.
- Then, different deep learning architectures are applied to the dataset - Recurrent Neural Networks, Convolutional Neural Networks (with word2vec as well), Recursive Neural Networks.

Dataset

- Stanford Sentiment Treebank Dataset (SST).
- It contains 11,855 sentences extracted from movie reviews. These sentences have 215,154 unique phrases and have fully labeled parse trees.
- The dataset has five classes for its labels : Strongly Negative, Negative, Neutral, Positive and Strongly Positive.
- A cross-validation split of 8,544 training examples, 1,101 validation samples and 2,210 test cases is already provided with the data.

Structure of a sample from SST dataset

Figure 1: Structure of a sample from Stanford Sentiment Treebank dataset.

Preliminary Analysis

- Practically, when the dataset is heavily biased towards one of the label classes, using accuracy is not the best way to measure performance.
- However, the distribution of sample labels in SST dataset is not dominated by any single class. This holds true for the distribution of labels in the validation set as well.
- Therefore, accuracy is used as a measure to compare results of different classifiers.
- The model is evaluated on the whole sentence as a unit, rather than evaluating it in phrase level. (although SST provides sentiments of phrases). This is because, sentiment analysis engines are usually evaluated on the whole sentence.

Distribution of labels in training set

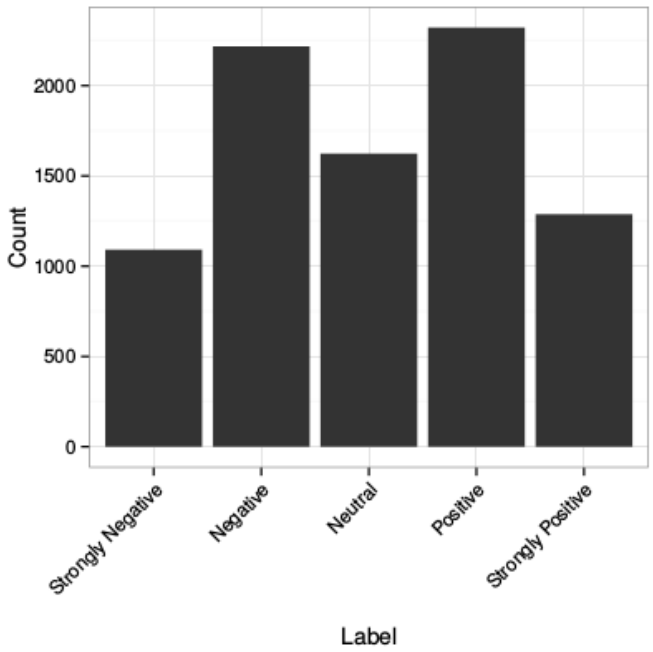


Figure 2: Distribution of labels in the training set of SST dataset.

Naive Bayes (NB) model

- It is based on the bag-of-words model.
- With the Naive Bayes model, we do not take only a small set of positive and negative words into account, but all words the NB Classifier was trained with, i.e. all words presents in the training set.
- If a word has not appeared in the training set, we have no data available and apply Laplacian smoothing (use 1 instead of the conditional probability of the word).

NB contd. . . .

- The probability a document belongs to a class C is given by the class probability $P(C)$ multiplied by the products of the conditional probabilities of each word for that class.

$$P = P(C) \cdot \prod_i P(d_i|C) = P(C) \cdot \prod_i^n \frac{count(d_i,C)}{\sum_i count(d_i,C)} = P(C) \cdot \prod_i \frac{count(d_i,C)}{V_C}$$

- Here $count(d_i, C)$ is the number of occurences of word d_i in class C , V_C is the total number of words in class C and n is the number of words in the document we are currently classifying.

NB results

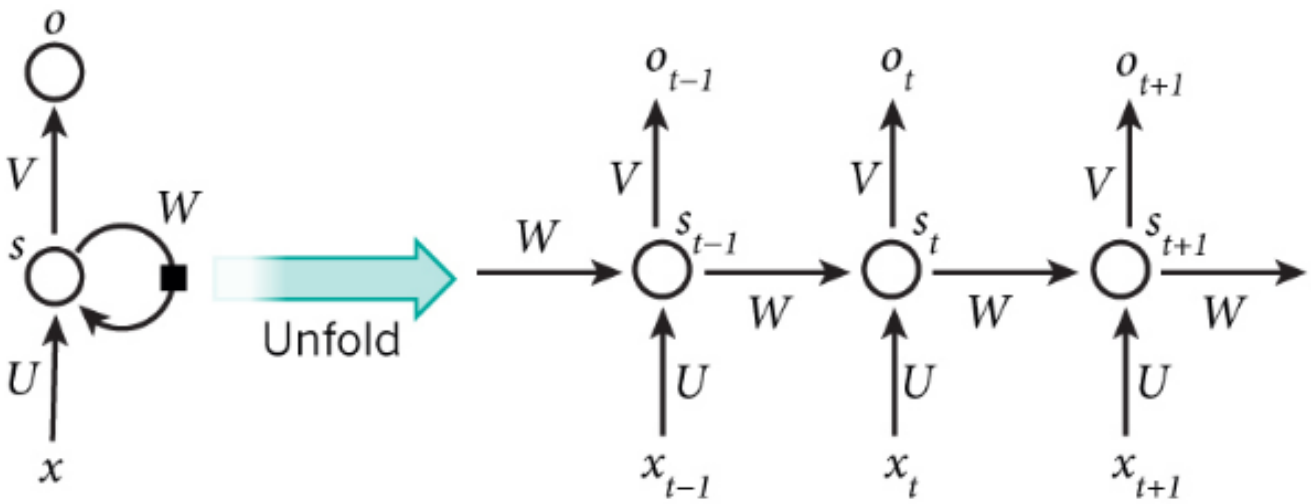
- The validation accuracy is 35.2%, the test accuracy is 38.4%.
- The Naive Bayes classifier performs relatively well in separating positive and negative sentiments.
- However, it is not very successful in modeling the lower level of separation between strong and regular sentiment.

Word2Vec

- Word2vec is a two-layer neural net that processes text. Its input is a text corpus and its output is a set of vectors: feature vectors for words in that corpus.
- The purpose and usefulness of Word2vec is to group the vectors of similar words together in vectorspace.
- Word2vec can make highly accurate guesses about a words meaning based on past appearances.
- Those guesses can be used to establish a words association with other words (e.g. man is to boy what woman is to girl), or cluster documents and classify them by topic.
- This forms the basis of using Word2Vec for sentiment analysis tasks.

Recurrent Neural Networks (RNN)

- The idea behind RNNs is to make use of sequential information.
- RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations.



RNN contd. . . .

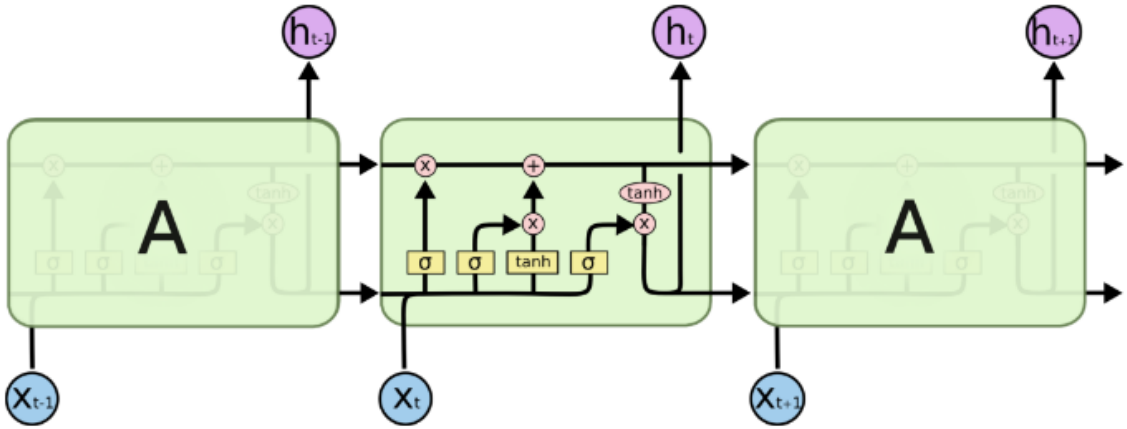
- The hidden state s_t is the memory of the network. s_t captures information about what happened in all the previous time steps.
- The output at step o_t is calculated solely based on the memory at time t.
- Unlike a traditional deep NN, which uses different parameters at each layer, a RNN shares the same parameters (U, V, W above) across all steps.
- This reflects the fact that we are performing the same task at each step, just with different inputs.
- This greatly reduces the total number of parameters we need to learn.

LSTMs

- The vanishing gradient problem prevents standard RNNs from learning long-term dependencies.
- LSTMs were designed to combat vanishing gradients through a *gating* mechanism.

$$
\begin{aligned}
i &= \sigma(x_t U^i + s_{t-1} W^i) \\
f &= \sigma(x_t U^f + s_{t-1} W^f) \\
o &= \sigma(x_t U^o + s_{t-1} W^o) \\
g &= \ tanh(x_t U^g + s_{t-1} W^g) \\
c_t &= c_{t-1} \circ f + g \circ i \\
s_t &= \tanh(c_t) \circ o
\end{aligned}
$$

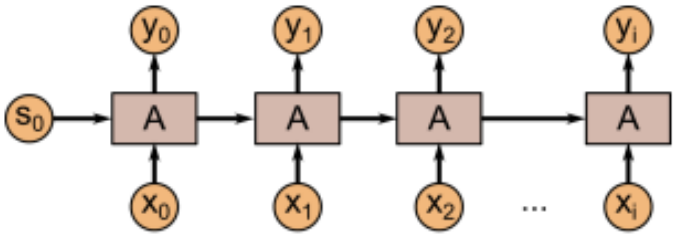
RNN / LSTM results



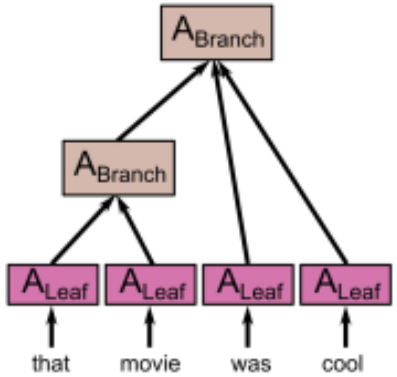
- Validation Accuracy : 26.49%
- Test Accuracy : 28.36%

Recursive Neural Networks

- Recursive neural networks (also called TreeNet) are generally used for learning tree-like structures.
- They can be seen as generalization of recurrent networks with skewed tree structure.



Recurrent Neural Net



Recursive Neural Net

Recursive NN contd. . . .

- Parent representations are generated in bottom up fashion, by combining tokens to produce representations of phrases , eventually producing whole sentence.
- Sentence level representation can be used for final classification.
- Given binary tree structure with leaves(words) having word vector representations, recursive network computes representation at each internal node as follows :

$$x_\eta = f(W_L x_{l(\eta)} + W_R x_{r(\eta)} + b) \tag{1}$$

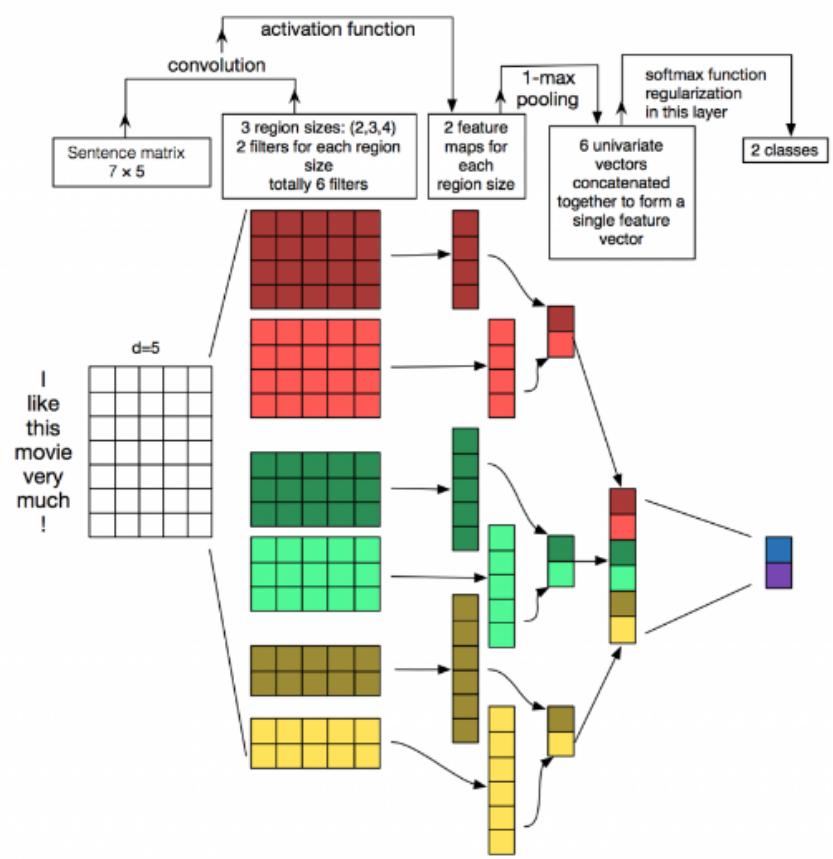
- The output layer above the representation layer is given by:

$$y_n = g(U x_\eta + c) \tag{2}$$

Recursive NN contd. . . .

- The advantage of TreeNets is that they can be very powerful in learning hierarchical, tree-like structure.
- The disadvantages are, firstly, that the tree structure of every input sample must be known at training time.
- The second disadvantage of TreeNets is that training is hard because the tree structure changes for each training sample and its not easy to map training to mini-batches and so on.
- **Results**: Validation Accuracy : 40.1 %, Testing Accuracy : 40.2 %

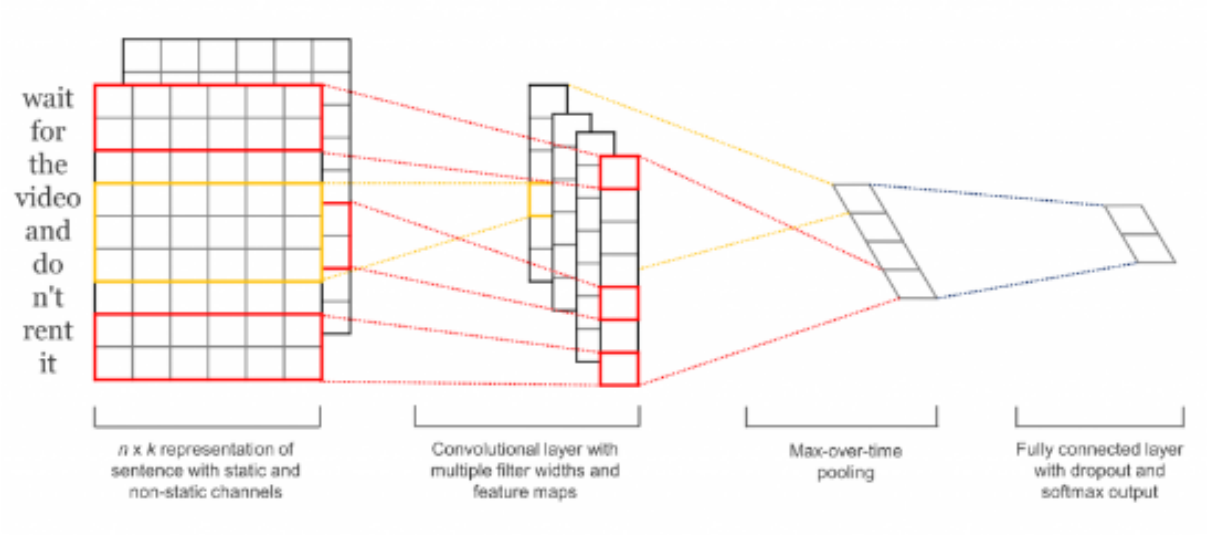
Using CNNs for NLP tasks



Source: CNNs for sentence classification

CNNs

The network we used looks roughly as shown below:



CNNs contd . . .

- The first layers embeds words into low-dimensional vectors (Using Word2Vec or Glove).
- The next layer performs convolutions over the embedded word vectors using multiple filter sizes. For example, sliding over 3, 4 or 5 words at a time
- Next, we max-pool the result of the convolutional layer into a long feature vector, add dropout regularization, and classify the result using a softmax layer.

CNN results

Validation Accuracy : 35.02
Testing Accuracy : 36.3

With pre-trained word2vec
Validation Accuracy : 44.92
Testing Accuracy : 47.38

Comparison of results

Table: Reference report results

Model	Validation accuracy	Test accuracy
Naive Bayes	38.0	40.3
Recurrent Neural Network	40.2	40.3
Recursive Neural Network	38.6	42.2
Convolutional Neural Network	41.1	40.5
CNN with Word2Vec	44.1	**46.4**

Table: Our results

Model	Validation accuracy	Test accuracy
Naive Bayes	35.2	38.4
Recurrent Neural Network	26.49	28.36
Recursive Neural Network	40.1	40.2
Convolutional Neural Network	35.02	36.3
CNN with Word2Vec	44.92	**47.38**

Important contributions and tech. challenges faced

- The reference report doesn't give details on the specifications of the architecture used (the number of layers etc.)
- It also doesn't info about hyperparameters (like batch size, number of epochs, filter size etc.) used which play a key role in determining the accuracy.

Scope for improvement

- This work does an extensive analysis of model variants (e.g. filter widths, k-max pooling, word2vec vs Glove, etc.) and their effect on performance.
- This work combines structure learning properties of a CNN with the sequence learning of an LSTM.