RAMAIAH
Institute of Technology

Department of Computer Science and Engineering

Coding Assignment for Deep Learning CSE754

Title: Convolutional Neural Networks for Sentence Classification	
Student USN	**Student Name**
1MS18CS130	**V G Nandan**
1MS18CS133	**Varun M**
1MS18CS134	**Venkat Satish A**
1MS18CS135	**Venkatesh S**

Brief Report

Problem Statement	*Include a brief summary of the problem and/or your tasks to be completed in this assignment*
Structure Chart	*Show your design rationale by creating a structure chart that indicates your top-down, stepwise refinement of the problem solution. You may create the structure chart by hand if you wish (use pen) but there are a number of automated graphical tools available.*
Implementation and Results	*Input: Description of the data received* *Output: Description of the information returned* *Algorithm: How the program accomplishes the task using pseudocode.* *Describe your tests, summarize your results, and argue that they cover all types of program behavior.*
Epilogue	*What were the trouble spots in completing this assignment? What parts caused you the most grief? What did you like about the assignment? What did you learn from it?*
Attachments	*Github link of the code assignment*

NOTE: The report need to be minimum 4 pages and maximum 6 pages

1. Problem Statement:

Text Classification is the one of the tasks which is widely used in Supervised Machine Learning and Natural Language Processing. It is mainly used to solve business related problems. In text classification, classification is done using a labelled dataset containing text documents, since it is a supervised machine learning classifier.

Automatic classification of text documents into one or more predefined categories is the aim of text classification. Sentiment analysis of the audience through social media activities, spam and non spam mail detection, covid vaccine tweet analysis are some of the examples of text classification. It is an active area of research both in industry and academia.

Our task is to do the sentence classification using Convolutional Neural Networks. Convolutional Neural Network (CNN) belongs to a class of deep, feed-forward artificial neural networks. CNNs are mostly used in computer vision (CV) but nowadays they are also used in solving NLP related tasks. In our experiment, we are specifically doing sentiment analysis on Movie Review dataset using three different variants:

- CNN-rand (Random initialization of word vectors)
- CNN-static (No updation of pre-trained Word2Vec vectors)
- CNN-non-static (Updates the pre-trained Word2Vec vectors too while training, for fine-tuning to the current task)

2. Structure Chart:

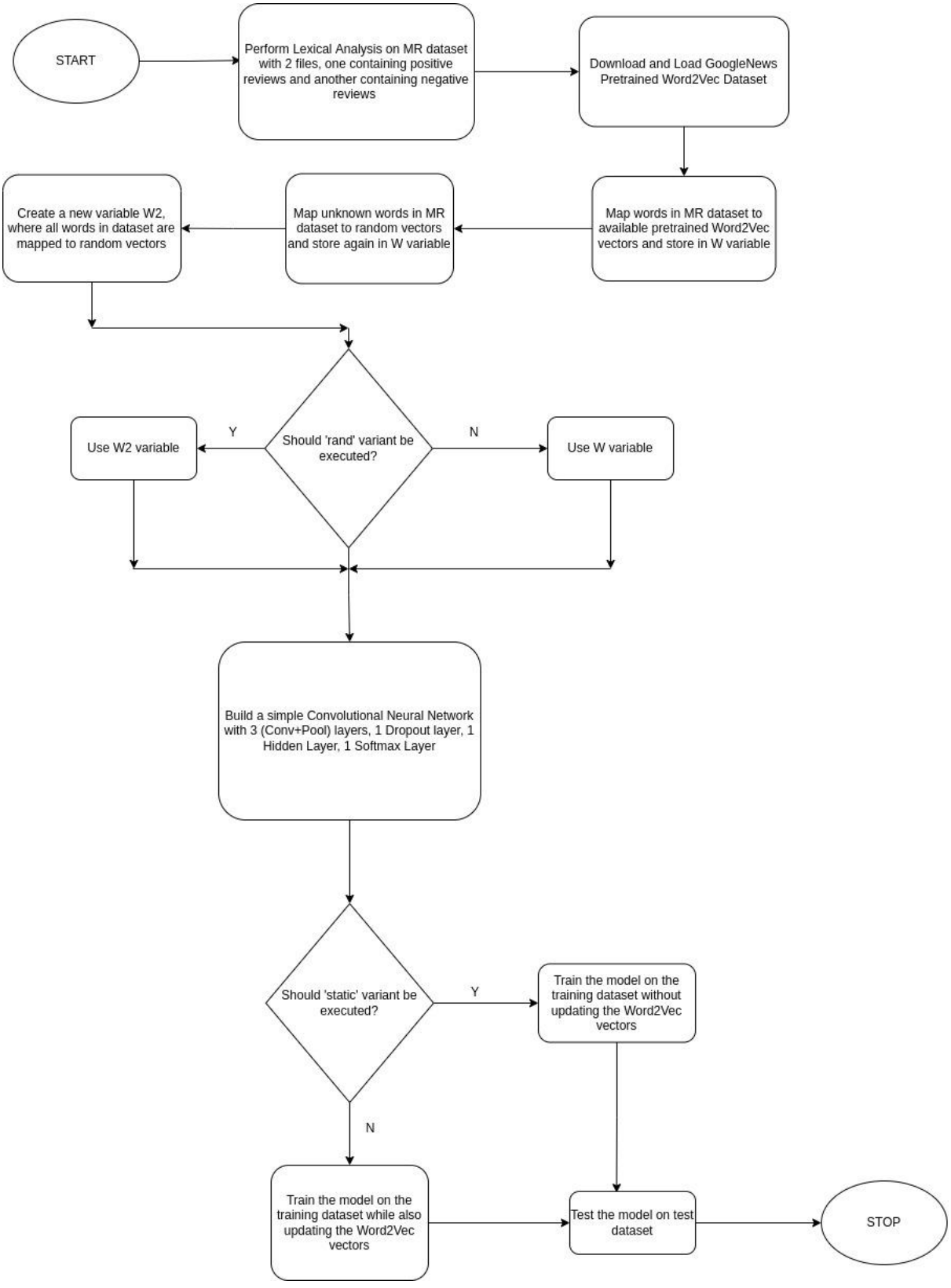

Fig. 1. Structure chart of sentence classification using CNN

3. Implementation:

a. Input:

i. Movie review dataset which contains two text files. One which contains positive reviews and other contains negative reviews for process_data.py file

ii. A pre-trained Word2Vec binary file downloaded from https://code.google.com/p/word2vec/ for process_data.py file

iii. The model type (rand, static, non-static) has to be passed as command line argument for conv_net_sentences.py file

b. Output:

i. **From process_data.py file:** A binary file called mr.p as output which contains the cleaned sentences, mapping word to the word index, mapping word index to word vector and the vocabulary containing the frequency of all words in the Movie Review dataset.

ii. **From conv_net_sentences.py file:** Training loss, training accuracy, test loss and test accuracy.

c. Algorithm Pseudo Code:

i. Perform lexical analysis on the Movie Review dataset of positive and negative reviews, such as removing non-alphanumeric characters, adding space before and after parentheses etc

ii. Define a variable word2vec containing key-value pairs of words and their vector representations from the pre-trained Word2Vec binary file

iii. Define a variable W as a key-value pair data structure

iv. FOR each word in document:

```
        IF word present in word2vec dataset
            W[word] = word2vec[word]
        END-IF
        ELSE
            W[word] = random vector
        END-ELSE
    END-FOR
```

v. Define a variable W2 as a key-value pair data structure

vi. FOR each word in document:

```
        W2[word] = 300x1 vector with random values
```

vii. IF variant == ‘rand’:

choose W2 as dataset

END-IF

ELSE

choose W as dataset

END-ELSE

viii. Build a simple Convolutional Neural Network model with 3 (Conv+Pool) pair layers, 1 Dropout layer (dropout rate = 0.5), 1 Hidden Layer (number of neurons = 100) and 1 Softmax Layer (number of neurons = 2, for the 2 classes). The 3 conv layers have filter sizes as 3, 4 and 5 respectively

ix. IF variant == ‘non-static’

train the model on the training dataset by updating the learnable parameters as well as the vector representation of words

END-IF

ELSE

train model on the training dataset by only updating the learnable parameters

END-ELSE

x. Test the model on the test dataset

4. Results:

All three variants (rand, static and non-static) were carried out by passing the desired variant as a command-line argument. Each model variant was trained three times for 13 epochs each and the average of the accuracies was found and recorded as shown in Table 1 below.

Table 1. Expected and obtained accuracies on different model variants

	Expected Accuracy (%)	**Obtained Accuracy (%)**
CNN Rand	76.1	76
CNN Non-static	81.5	79.4
CNN Static	81	80.1

Thus, we managed to obtain accuracies that are comparable to the accuracies mentioned in the paper for all the three variants. The differences between the expected and obtained accuracies can be attributed to reasons such as different environments and different random vector initializations. CNN-rand gives the least accuracy in the paper as we map each word to a random vector representation. CNN-non-static gives a better accuracy than CNN-static in the paper as in the case of CNN-non-static, even the Word2Vec representations are updated during model training which helps in fine-tuning them for our task.

5. Epilogue:

a. What were the trouble spots in completing this assignment?

Initially there was a lot of trouble to pick which implementation to go ahead with. We initially found an implementation which gave a good result but only one use case (CNN-rand) was done but our assignment had other use cases as well (CNN-static and CNN-non-static). Later we tried out another implementation with all mentioned use cases. CNN-rand and CNN-non-static performed well but CNN-static gave a poor accuracy of 65% where the expected accuracy was 80.1%. Later we found this implementation which was done in python 2.7. But python 2.7 is outdated and the code had many unresolved bugs such as we had to type convert the data from bytes to string, memory errors when reading files etc, so we had to fix the code so that it can be run.

b. What parts caused you the most grief?

Understating the working of code caused us the most grief as each and everything was implemented from scratch using the old Theano framework of which we had little idea.

c. What did you like about the assignment?

Though understanding the code caused us the most grief, it also helped us understand the working of deep neural networks (here, CNNs) at a very low level, which further enhanced our knowledge about it.

d. What did you learn from it?

i. Extensive text data pre-processing
ii. Got to know about Theano framework for running deep learning models
iii. Text classification using CNNs

6. Attachments:

Github Link: https://github.com/venkatesh2000/deep-learning-assignment