Report: Assignment 1

Abhishek Kumar
CSA,IISc Bangalore
abhishekkumar@iisc.ac.in

Abstract

This document contains the final report of assignment 1 of NLU. It contains details of different task performed in assignment 1. The task was to implement and experiment with prediction-based models for learning word embedding. I have used the Reuters corpus for this. The first task was to implement a word2vec model as described by Mikolov (Mikolov et al). It includes comparison of various skip-gram model having different hyper-parameters configurations. It shows the evaluation of learned character embedding on SimLex-999 evaluation benchmarks and comparison of word analogy.

1 Task 1

1.1 Pre-processing of the dataset, model implementation and training

In task 1, word2vec skip-gram model is implemented. I have used NLTK's reuters corpus for training of the model. The model is implemented using tensorflow but the core ideas of word2vec is implemented by self only. Followings are the different stages of the implementations:

→ **Pre-processing of the reuter corpus**
The data set contains punctuation, numbers, frequent words, etc. which needed to be removed before they can be used for training our model. Followings methods were used for pre-processing:
• All the letters in the dataset were converted to lower case only. All the numeric letters/words, punctuation, accent marks and other diacritics were removed. Finally, all stop words were removed.
• For tokenizing, default word_tokenizer provide by NLTK is used.
• Subsampling of the data (as described by Mikolov) i.e., words that show up often such as 'the', 'of', and 'for' do not provide much context to the nearby words.
After all these steps a vocabulary of unique words was created where $|Vocabulary| = 34,703$. The tokenized corpus was converted into Word pairs : (center words , context word) and all models were trained on this pre-processed data.

→ **Implementation of word2vec skip-gram model.**
• Skip-gram model was implemented and trained on the above pre-processed data. Algorithm mentioned in the word2vec paper by Mikolov was used for the implementation. I have used negative sampling for faster and efficient training of the model.

1.2 Comparison between various model for different hyper-parameters.

Window Size:
Window size = $\{2, 4, 6\}$ while keeping other hyper-parameters fixed and have come to following conclusions.
• Smaller window size model has lower loss as compared to the model with larger window size.
No. of iteration it took to convergence was more or less equal (≥ 500)
Batch size:
Batch size = $\{400, 600, 1000\}$
Didn't have any conclusive effect on loss function after sufficient no. of iterations. But smaller batch size model converges faster compared to larger batch size model.
Embedding dimension:
Embedding dimension = $\{128, 300\}$
Training loss was less for the model with smaller embedding size.

1.3 Final Model.

After experimenting with different hyper-parameters, I have chosen following configurations for final model:
Window size = 2 i.e., $\{W_{i-2}, W_{i-1}, W_{i+1}, W_{i+2}\}$.
Lower window size is preferred because it leads to early convergence and it takes less resource and time to train the final model.
Batch size = 400 , as it does not have much effect on convergence.
Embedding dimension = 300 , While keeping the above two hyper-parameters fixed i.e., Window size = 2 and batch size = 400, It was giving the less loss on training data and better evaluation on SimLex-999.
Negative sample size = 10
No. of Epochs = 10

1.4 Evaluation of final model on SimLex-999

The result of the final model on SimLex-999 evaluation benchmark is shown below.

SimLex-999 Evaluation Benchmark	
Model	Spearman Correalation
NCE-300-400-10	0.0504823

Correlation Coefficient assesses how well the relationship between two variables can be described using a monotonic function. It was calculated for the cosine similarity between embedding of a pair of words generated by model and the cosine similarity value of same pair of word given by SimLex-999 dataset.

2 Task 2

Word Analogy Task
I have evaluated the final model using question answering set provided by google code archive for word analogy task.
The Model is performing very poorly for different K neighbours values.

Word Analogy Task	
K-value	Accuracy
12	0.071%
6	0.0%

Possible reasons:
• Maybe final model is not properly trained i.e., no. of epochs should have more.
• Maybe configuration is not best as final model has lower spearman correlation coefficient and it is also performing poorly for word analogy task.
• Model is performing very poorly maybe because it is trained on a smaller dataset.