==========================================================================================

Introduction to Natural Laguage Processing Assignment 1

Name: Hemasai Aishwarya Vijayakumar

SBU ID: 112673842

==========================================================================================

1. In this assignment, you will be asked to:

- generate batch for skip-gram model (word2vec\_basic.py)

- implement two loss functions to train word embeddings (loss\_func.py)

- tune the parameters for word embeddings

- apply learned word embeddings to word analogy task (word\_analogy.py)

2. Generating batch

For skip-gram model, A window is created of the size 2\*skip\_window+1. After this all the target words (Center words) are appended to the batch array. The context words are appended to the labels array. A list of context words is created by deleting the center/target index from the list of indices in the window.

[Example]

Suppose that we have a text: "The quick brown fox jumps over the lazy dog."

And batch\_size = 8, window\_size = 3

"[The quick brown] fox jumps over the lazy dog"

Context word would be 'quick' and predicting words are 'The' and 'brown'.

This will generate training examples:

context(x), predicted\_word(y)

(quick , The)

(quick , brown)

And then move the sliding window.

"The [quick brown fox] jumps over the lazy dog"

In the same way, we have two more examples:

(brown, quick)

(brown, fox)

Moving the window again:

"The quick [brown fox jumps] over the lazy dog"

We get,

(fox, brown)

(fox, jumps)

Finally we get two more instances from the moved window,

"The quick brown [fox jumps over] the lazy dog"

(jumps, fox)

(jumps, over)

Since now we have 8 training instances, which is the batch size,

stop generating this batch and return batch data.

data\_index is the index of a word. You can access a word using data[data\_index].

batch\_size is the number of instances in one batch.

num\_skips is the number of samples you want to draw in a window(in example, it was 2).

skip\_windows decides how many words to consider left and right from a context word(so, skip\_windows\*2+1 = window\_size).

batch will contains word ids for context words. Dimension is [batch\_size].

labels will contains word ids for predicting words. Dimension is [batch\_size, 1].

3. Loss Functions:-

a. Cross entropy:

Equations used to calculate cross entropy are:

A = log(exp({u\_o}^T v\_c))

B = log(\sum{exp({u\_w}^T v\_c)})

Final result: B-A

Calculation for ‘A’:

1. The dot product of inputs and transpose of true\_w is taken

2. Then exp followed by log is applied on this dot product

Calculated in step 1.

Calculation for ‘B’:

1. The dot product of inputs and transpose of true\_w is taken

2. Then reduce sum is done on the exp of this product(column

Wise)

3. log is applied on the value calculated in step 2.

Then returning B-A(for both A and B, 1e-10 is added as log

Correction)

b. Noise Contrastive Estimation:-

1.Obtained values of dimensions- batch\_size, embedding\_size

And num\_sampled

2. Unigram probability of the Target is converted to tensor

3. Pr(D=1,wo|wc) is calculated by reshaping and using

Embedding\_lookup

4. Target word vector, uo is also calculated

5. Pr(D=1,wx|wc) is initialized and all unigram probabilities

Of negative samples are appended to this.

6. Sigmoid function is used based on the formula given

7. A log correction of 1e-10 is added to all the logs

3. Analogies using word vectors

The task is to find the least similar and most similar pair of words to

A given set of pair of words. The given pair of words are called the

Examples. We need to choose the most and least similar words from

‘choices’ given.

For example,

Examples can be pilgrim:shrine, hunter:quarry, assassin:victim

Choices can be

(1) pig:mud

(2) politician:votes

(3) dog:bone

(4) bird:worm

Method followed:

1. Calculate embeddings of each word in example pairs
2. Get the difference of these embeddings for each pair
3. Get the average of the values calculated in step 2 for all example pairs
4. Similarly, calculate these differences for all pairs in the choices list
5. Now, to compare the list of choice embeddings with the average of example embedding use:-

Cosine similarity = 1- spatial.distance.cosine(first\_embedding, second\_embeddings)

1. The pair with the highest cosine similarity is the highest
2. Illustrative pair and least cosine similarity is the least

Illustrative pair

4. Model Configuration:

Python: 3.7

Tensorflow: 1.14.0

Cross Entropy Best Model:

batch\_size=256

num\_sampled=128

skip\_window=4

num\_skips=8

max\_num\_steps= 200001

Accuracy: 31.6%

NCE Best Model:

batch\_size=256

num\_sampled=128

skip\_window=4

num\_skips=8

max\_num\_steps= 200001

Accuracy: 31.8%

Models are found here:

<https://drive.google.com/open?id=1XCQrAy10eorM2iiHhn9zZe2kdMDL1LYA>

5. This package contains several files:

For word2vec

- word2vec\_basic.py:

This file is the main script for training word2vec model.

Usage:

python word2vec\_basic.py [cross\_entropy | nce]

- loss\_func.py

This file have two loss functions

1. cross\_entropy\_loss

2. nce\_loss

For analogy task

- word\_analogy.py

For evaluating relation between pairs of words

- word\_analogy\_test\_predictions\_cross\_entropy.txt

A prediction file that was generated from the best model of cross

entropy

- word\_analogy\_test\_predictions\_nce.txt

A prediction file that was generated from the best model of NCE.

Models

Best models for Cross Entropy and Noise Contrastive

Estimation.

- word2vec\_cross\_entropy.model

- word2vec\_nce.model

README

- A file with explanation of my implementation

REPORT

- report\_112673842.pdf

A report for the project.