# CS565, Assignment 3: Implementation of Bigram word2vec model

Input: Single word

Output: Single word

Reference: word2vec Parameter Learning Explained, Xin Rong

You are supposed to implement the model described in the section 1.1 of this document.

Implementation language: C++

You should submit your code as a single file: <roll number>_assign3.cpp

How your program should compile: g++ <roll number>_assign3.cpp

Make sure that g++ version on your system is 10.2 or higher.

How your program should run: ./a.out <input.txt  output>txt

Read every input from standard input.

Write every output to the standard output.

Format of input.txt

Number of distinct words in the vocabulary. If this value is n then all words will have id from 1 to n.

Number of dimensions in the resulting word vectors. This also represents the number of units in the hidden layer.

Learning rate.

Number of iterations.

Number of word pairs in the training corpus.

<triplet: pair id input word output word>

In a training iteration, you will go through all training pairs sequentially. You will use stochastic gradient descent to train your word vectors.

Initialize all word vectors with value 0.5 for each dimension. We understand that it might not be a good choice. But it is required for ease of evaluation.

Try to use STL classes and library functions as far as possible.

A useful tip for implementing softmax:

https://stackoverflow.com/questions/52897031/softmax-implementation-in-c

Use "long double" data type to represent all floating point numbers.

You will use the extreme case of negative sampling. In the output layer, you will update weights only on the single word that is the output word. (Equation 9, page 4)

However, while updating weights for the input word, you will consider error from all output layer neurons. (Equation 11, page 4)

After processing each pair, you should write the following to the standard output:

<iteration id> <pair id> <number of negative weight updates> <number of non-negative weight updates>

Total number of lines in your output: number of iterations X number of training pairs.

Each line will have four integer values separated by a space.

With each training pair, you will update two vectors: output vector of output word and input vector of input word.

If number of hidden layer neurons are d then total number of weight updates: 2X d

Number of negative weight updates: Out of these 2d updates, number of updates where you are reducing the value of existing weight.

Number of non-negative updates: Out of 2d updates, number of updates where you are either increasing the existing weight or keeping it constant.

Consider the given example input.txt file

We have four words in our vocabulary. (line 1)

We have to generate word embeddings with two dimensions. (line 2)

The learning rate is 0.01. (line 3)

We have to perform 5 iterations. (line 4)

There are 20 training word pairs. (line 5)

Training pair 1, input word id: 1, output word id: 2

We will initialize input weight matrix as follows

0.5 0.5

0.5 0.5

0.5 0.5

0.5 0.5

We will initialize output weight matrix as follows

0.5 0.5 0.5 0.5 0.5

0.5 0.5 0.5 0.5 0.5

Input one hot encoding will be the following vector

1

0

0

0

Output of the hidden layer will be the product of input weight matrix transpose multiplied by the input vector. It will produce a vector of size d, where d is the number of neurons in the hidden layer.

In this case hidden layer output will be:

0.5

0.5

We will compute unthresholded output at each output neuron using Equation 2, on page 2.

In this particular case, output at neuron will be scalar value 0.5 ( sum of 0.25+0.25).

The softmax calculation will assign value 0.25 for each word.

Now, you will compute update for output word vector of word id 2 only using Equation 9, page 4.

You will not update rest of the output word vectors as we you are using extreme case of negative sampling.

Now, you will compute update to the input word vector of word id 1 using Equation 15, page 5.

You just updated two vectors. Keep track of how many updates were negative and how many were non-negative. Write a line to the standard output

<iteration id> <pair id> <number of negative weight updates> <number of non-negative weight updates>

Your output file will have 5X20=100 such lines.