**CS5590 APS -Deep Learning Programming**

**LAB ASSIGNMENT-3**

**LOGISTIC REGRESSION**

**&**

**WORD EMBEDDINGS**

**Team-No:11**

|  |  |  |  |
| --- | --- | --- | --- |
| Class-ID | Name | Mail-Id | Contribution |
| 7 | Kranthi Kumar Gangineni | kgh68@mail.umkc.edu | 50% |
| 26 | Venkata Bhavesh Reddy Polareddy | vp5dt@mail.umkc.edu | 50% |

1. **Introduction:**
   1. **Logistic Regression:**

It’s the statistical model used in Machine learning where we have only two types of data or outcomes. Some of the examples like Gender(Male/Female), Validation(True/False), Binary (0/1) and so on. Then we can make some predictions using this Logistic Regression.

* 1. **Word Embeddings:**

It is a technique where we map the words based on their vocabulary to the corresponding vectors. Then Word embeddings will map words with the related words and display them in the single cluster. Some of the mappings such as King and Queen, Man and Woman, Boys and Girls etc.,

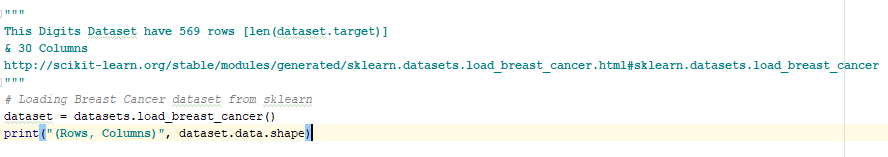
1. **Objectives:**

We should Accomplish the following tasks on the Logistic Regression and Word Embedding programs:

* 1. Change the data set which was not used before.
  2. Then Display the graph on the Tensor Board.
  3. Then Change the Hyper Parameters and compare the result.

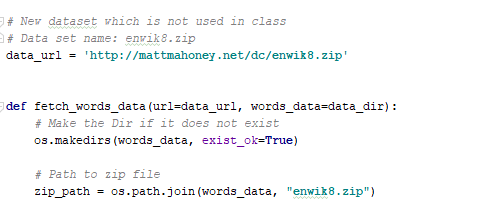
1. **Datasets:**
   1. **Data Set in Logistic Regression:**

We are using Breast cancer dataset taken from the Sklearn library where it has 569 Rows and 30 columns.



* 1. **Data Set in Word Embeddings:**

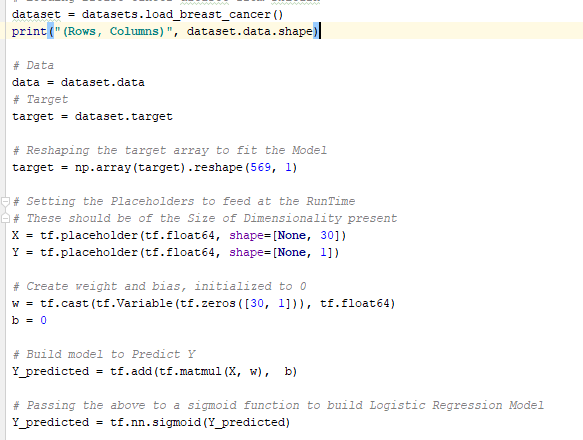
Here we are using the enwik8.zip dataset taken from the <http://mattmahoney.net/dc/textdata.html>.

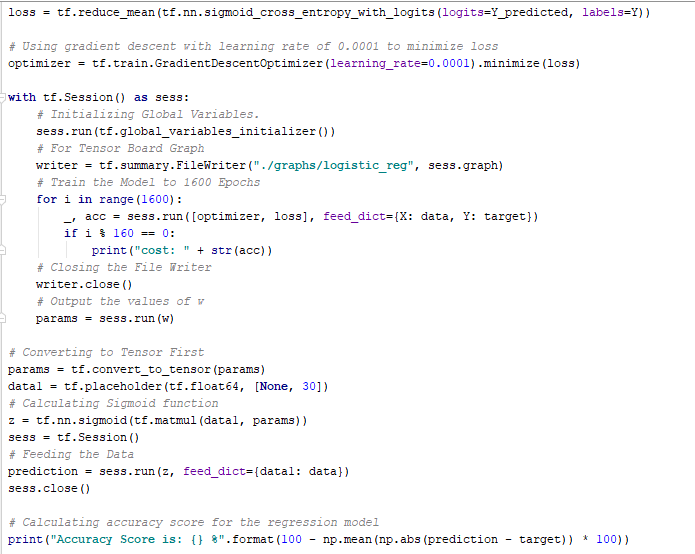


1. **Approaches:**
   1. **Approaches for Logistic Regression:**

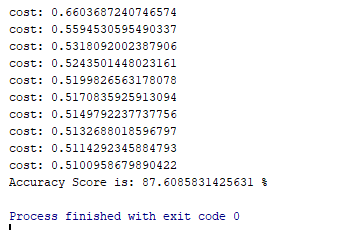
* Here Firstly, we are loading the breast cancer dataset.
* Later we are reshaping the array into the matrix using numpy.
* We are using the placeholders to use the data later at runtime.
* Then we are calculating the predicted value of y using the formulae wX+b.
* We are passing this value to the sigmoid function to calculate the Logistic Regression.
* Then calculate the loss using mean function.
* With the help of tensor board session variables, we are displaying the graph on the tensor board.
* Then we are calculating the accuracy by calculating the difference between the prediction and target value.

**Code Snippets:**

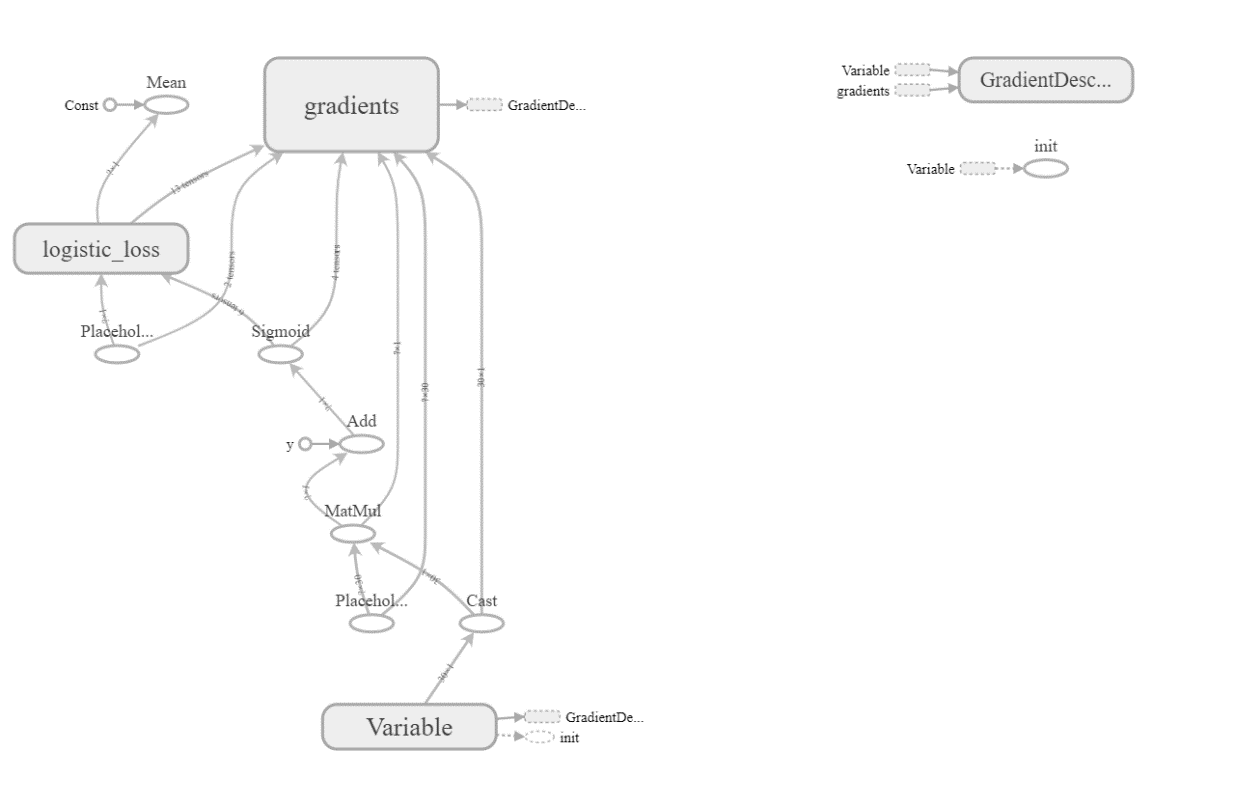




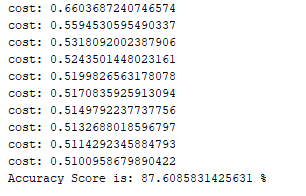
**Output Screenshots:**



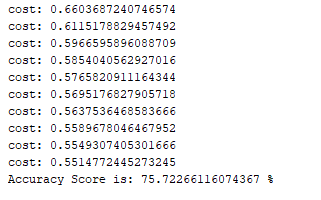
* + 1. **Graph on Tensor board:**



* + 1. **Changing Hyper Parameters:**
       1. **Case-1: Learning Rate= 0.0001 results:**

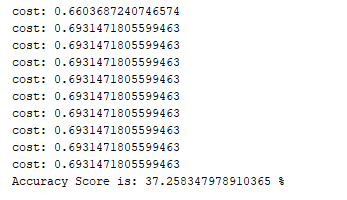


* + - 1. **Case-2: Learning Rate= 0.00001 results:**



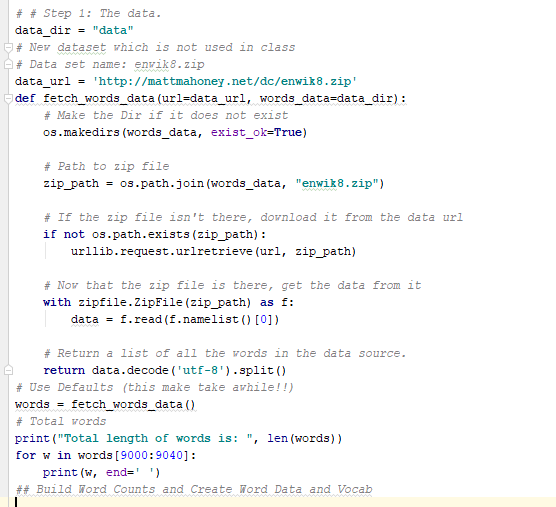
* + - 1. **Case-3: Learning Rate= 0.1,0.01,0.001 results:**

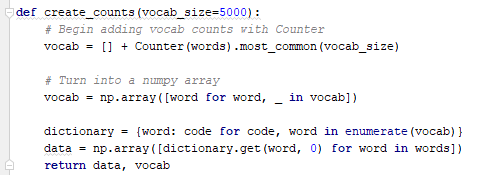
For all the learning rates above the output is almost similar the change was in the last digits of the decimals.

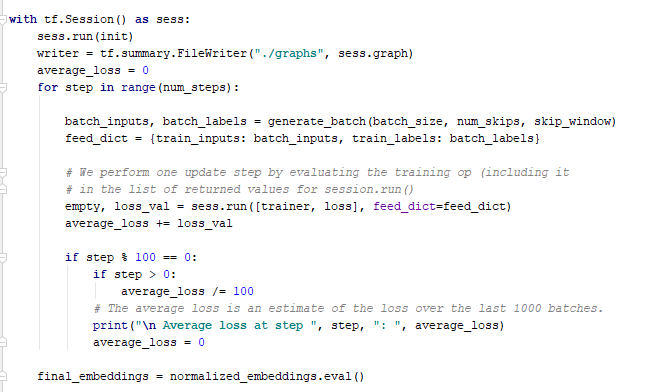


* 1. **Approaches for Word Embeddings:**
* Firstly, we are loading the enwik8.zip data set
* Then we are fetching the words from the URL
* Based on the vocabulary size we are creating the count of words
* We generate the batch based on the batch\_size, num\_skip and skip\_window
* We are loading the input data using the place holders
* We are initializing the hyper parameter variables
* Calculate the loss using the reduced mean function
* With the help of the tensor flow session we are calculating the average loss for every 100 steps of 500 steps.
* Later we are plotting the graph using the sklearn.manifold.
* Then display the graph on the tensor board
* Finally, Change the parameters and compare the results.

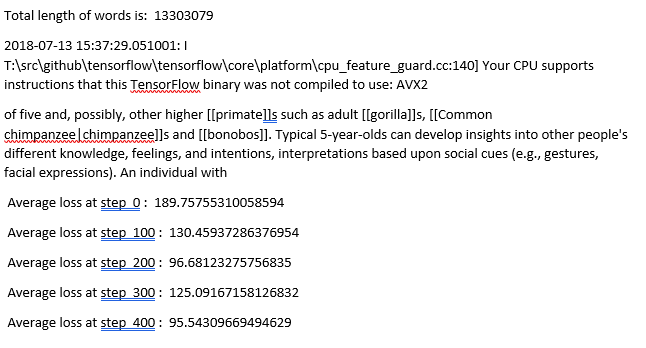
**Code Snippets:**



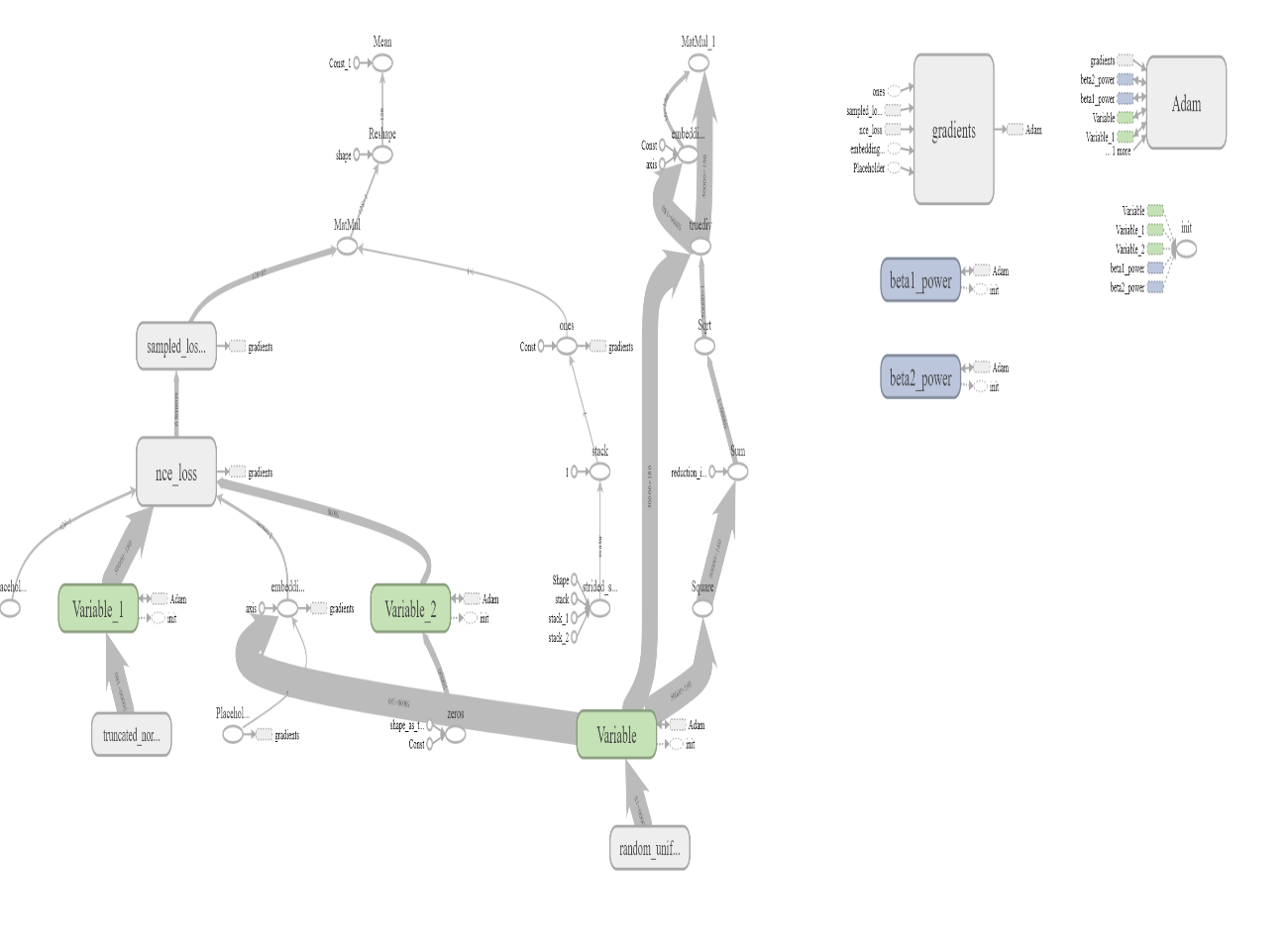




**Output Screenshots:**

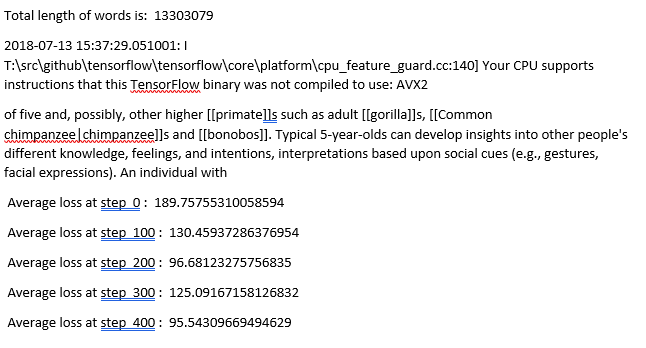


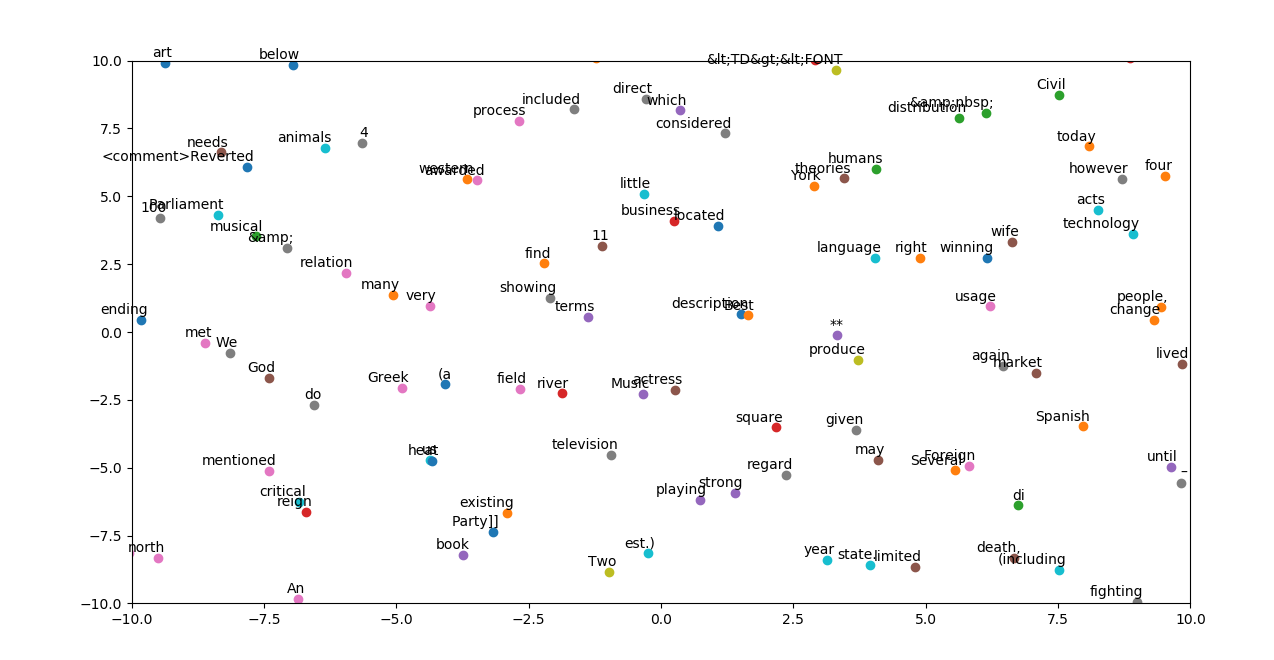
* + 1. **Graph on Tensor Board:**



* + 1. **Changing Hyper Parameters:**
       1. **Case-1: Learning rate=0.01, num\_steps=500, embedding\_size=150, valid\_size=16**

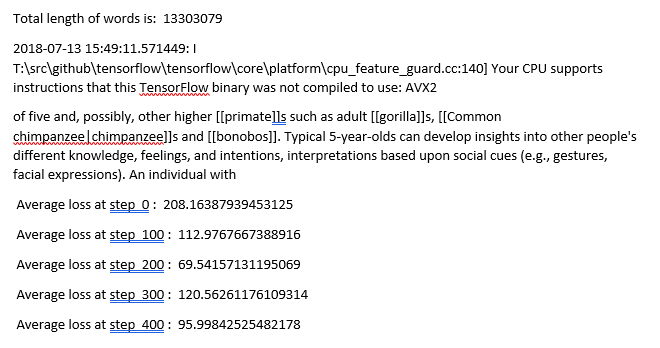
**OUTPUT:**

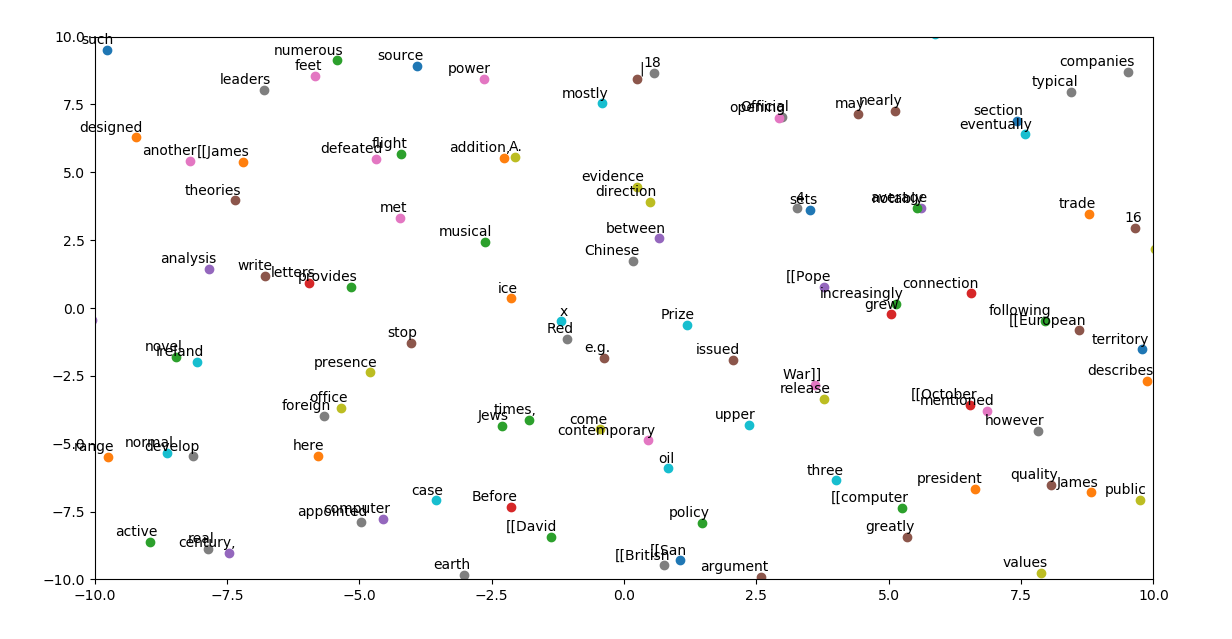




* + - 1. **Case-2: Learning rate=0.05, num\_steps=500, embedding\_size=150, valid\_size=16**

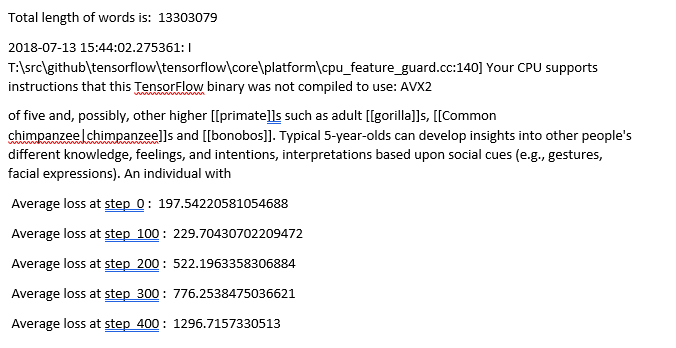
**OUTPUTS:**

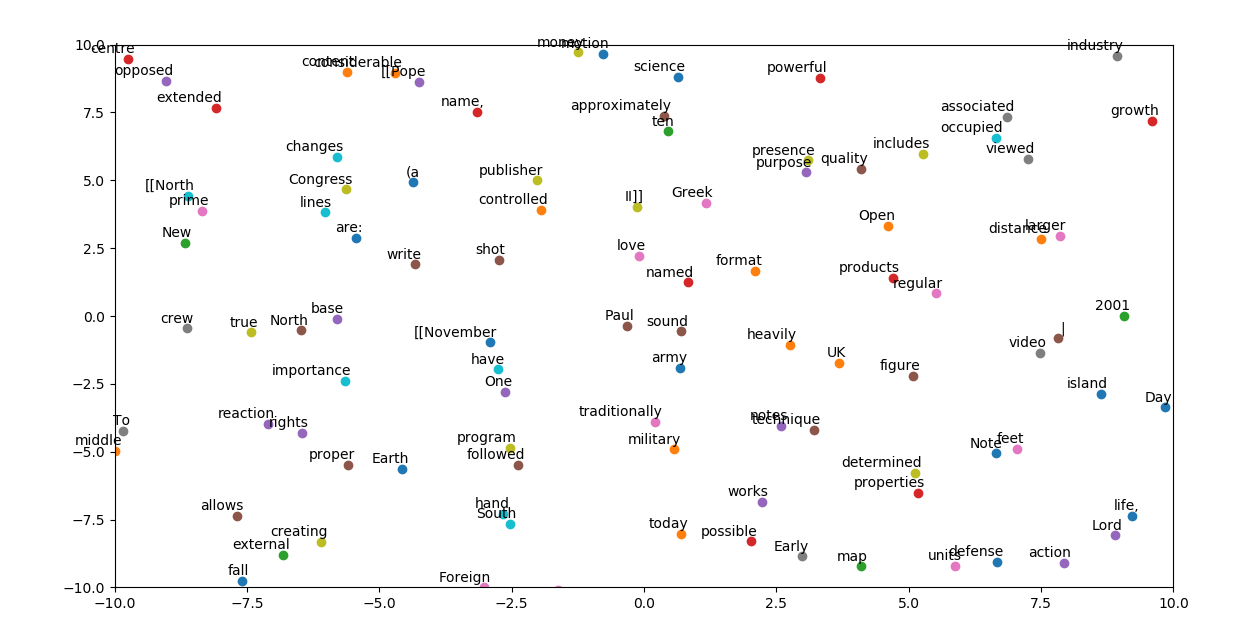




* + - 1. **Case-3: Learning rate=0.5, num\_steps=500, embedding\_size=150, valid\_size=16**

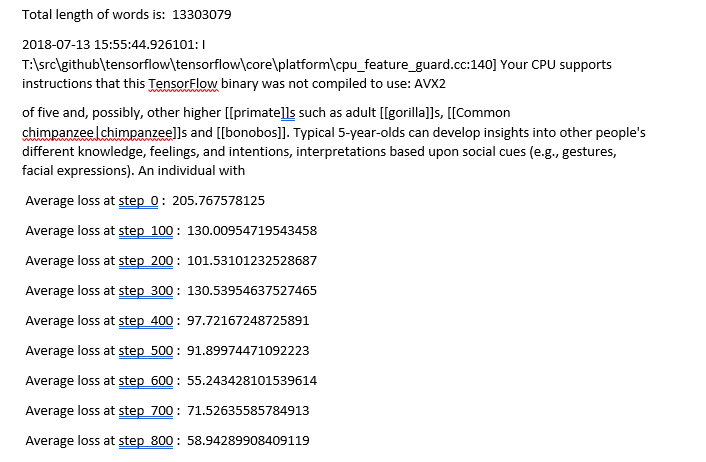
**OUTPUTS:**

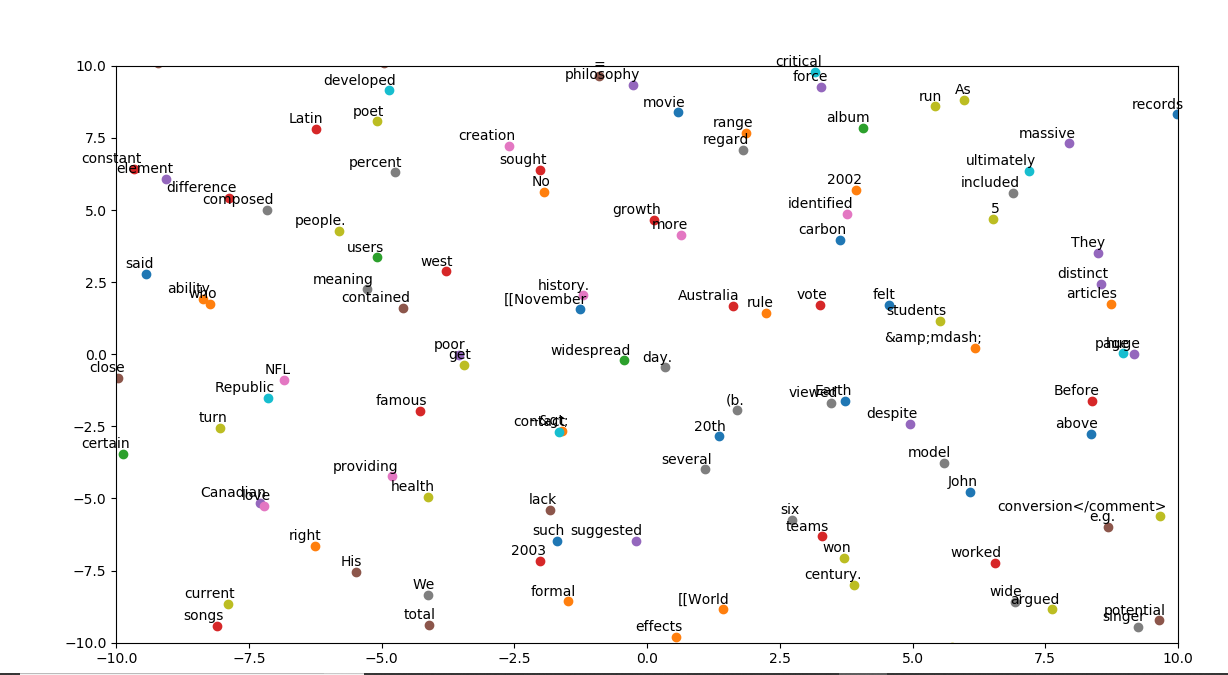




* + - 1. **Case-4: Learning rate=0.01, num\_steps=900, embedding\_size=150, valid\_size=16**

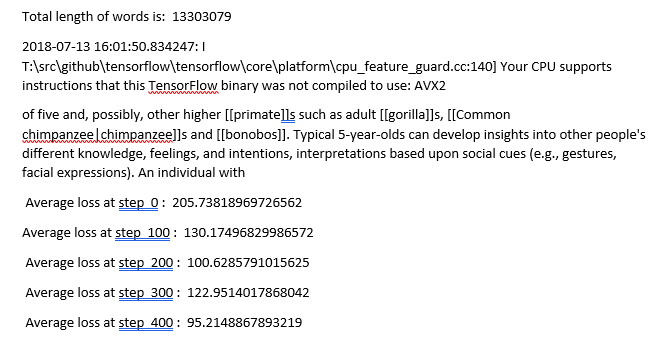
**OUTPUTS:**

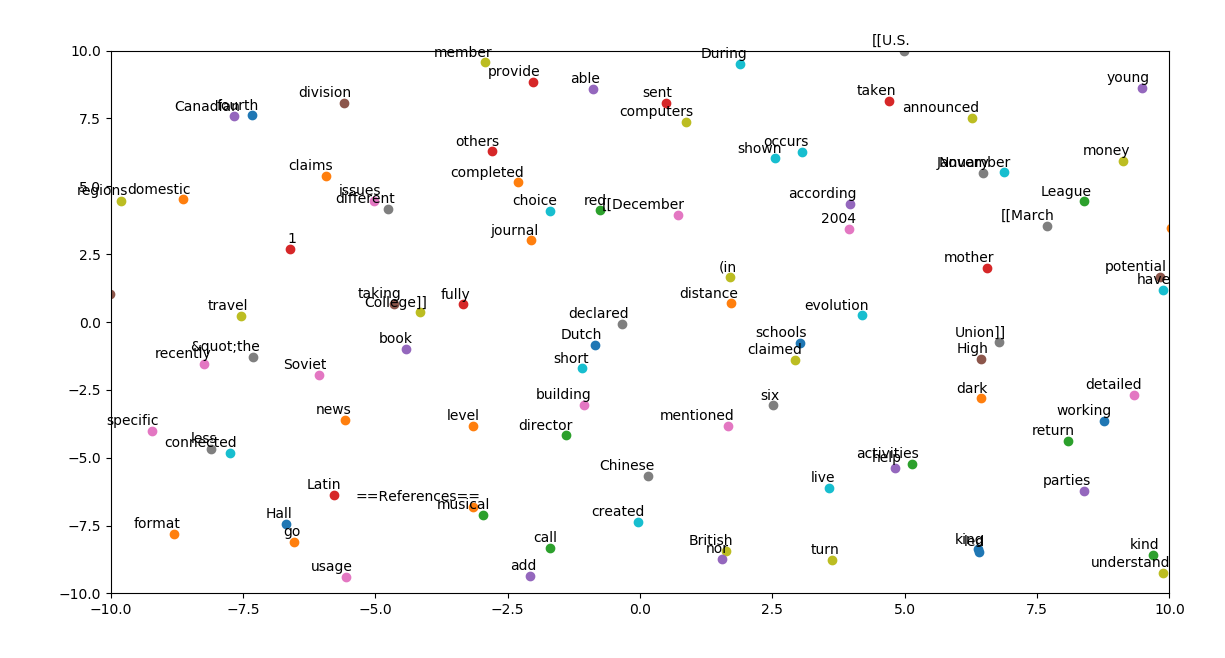




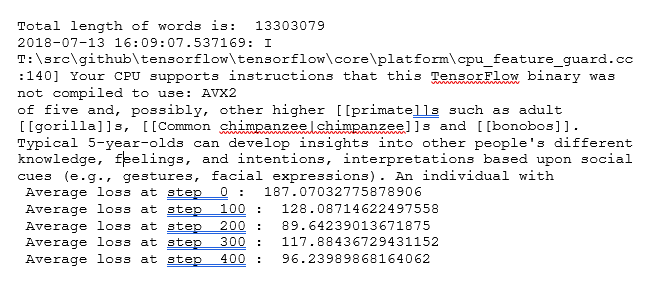
* + - 1. **Case-5: Learning rate=0.01, num\_steps=500, embedding\_size=150, valid\_size=64**

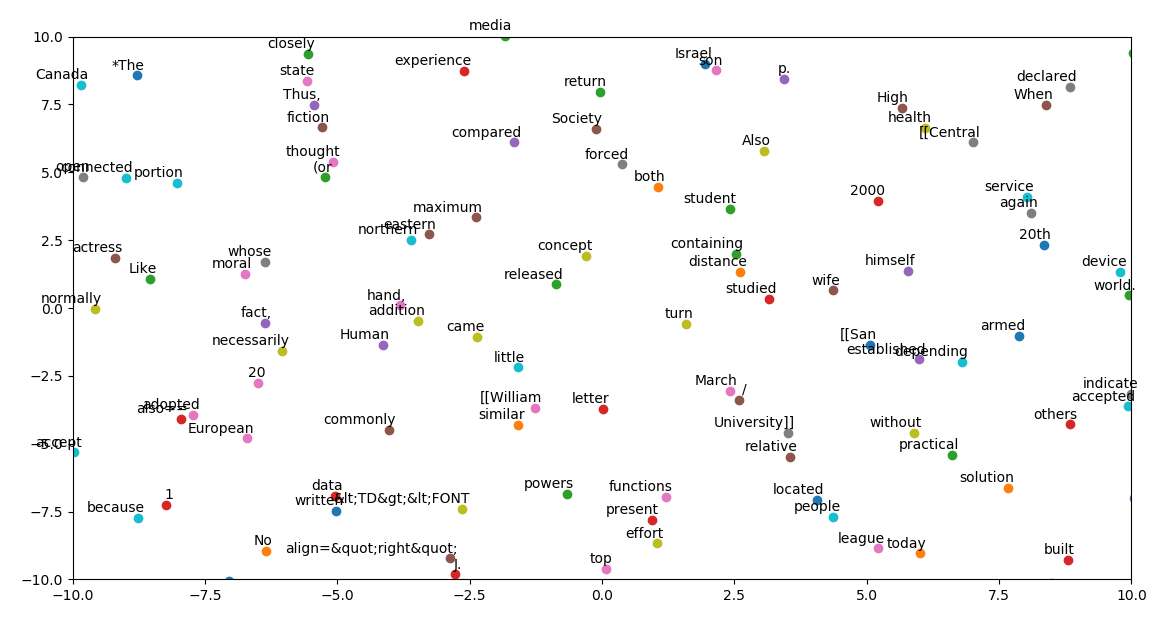
**OUTPUTS:**





* + - 1. **Case-6: Learning rate=0.01, num\_steps=500, embedding\_size=300, valid\_size=16**





1. **Conclusion:**
   1. **For Logistic Regression,** when we have the learning rate of 0.0001 then we are getting the accuracy of 87.6, it is the best case and later we had tried with the values of 0.00001 then we are getting the accuracy of 75.72 and finally for the values of 0.01, 0.05, 0.1, 0.5 the accuracy was 37.25.
   2. **For Word Embeddings,** if we increase the learning rate the Average loss has been gradually increasing. Suppose if the learning rate is 0.01 the average loss for 500 steps is 95 approx. and when we change the learning rate from 0.01 to 0.5 the average loss increased to 1296 approx. Later when we changed the number of steps the average loss was gradually reduced. When we change the valid size from 16 to 64 there is not much difference we can see the little bit difference in the decimals. When we changed the embedded size from 150 to 300 then we can see the increase in the average loss.
2. **References:**
   1. <https://www.quora.com/What-is-word-embedding-in-deep-learning>
   2. <http://mattmahoney.net/dc/textdata.html>
   3. <http://mattmahoney.net/dc/enwik8.zip>
   4. <https://github.com/Elhamkesh/Breast-Cancer-Scikitlearn>