**CS506 Programming for Computing**

**HOP10 Computer Vision – Text Classification**

12/06/2020 Developed by Kim Nguyen

Center for Information Assurance (CIAE) @City University of Seattle (CityU)

**Before You Start**

* The directory path shown in screenshots may be different from yours.
* Some steps are not explained in the tutorial**.** If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

* Use Machine Learning to classify texts.
* Process, build, train datasets to make and verify predictions.

**Resources**

* Tensorflow.org
* TechWithTim

**Preparation**

***[NOTE: your environment might look different from the images below]***

Last week, we have learned how to

* + Use Machine Learning to classify images.
  + Process, build, train datasets to make and verify predictions.
  + Create Convolutional Neural Networks

This week, we will continue to practice working on datasets using embedding layers with text classification. We will be working on the IDMB dataset, that contains the text of 50,000 movie reviews.

**IMPORT DATASET**

1. In Visual Studio Code, open the private repository generated when you accepted the HOP10 assignment (If you cannot find that repository in your machine, you might have not cloned the repo, if so, please do before proceeding).
2. Open Jupyter Notebook, under Module 10 folder, create a new file called **Text\_Classification.ipynb** and simply click on the file to open notebook.
3. Type the following into the **Text\_Classification.ipynb** file, to all the libraries needed:

Text

Description automatically generated

1. Type the following to a new block, to import the IMDB dataset above:

A picture containing calendar

Description automatically generated

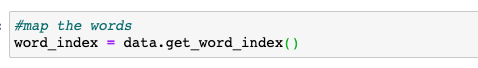
1. Let’s print out the dataset to see what we are working with:

Scatter chart

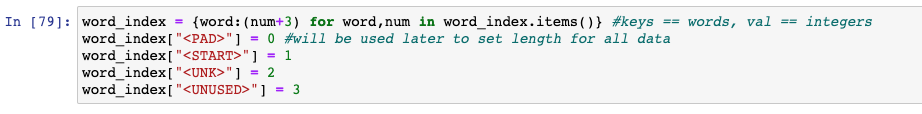
Description automatically generated

As you can see, the results are all numbers, and it’s not so easy for us to understand. We are working on movie reviews, so shouldn’t we expect letters instead of numbers? Yes. In fact, each of the number above points to a word and this is easier for the computer to handle. Let’s transform them into words, so that we can read. First, we need to create a map between the words and the numbers.

1. Put the following code into a new block:



1. Put the following code into a new block:



<PAD>: will be used for padding. As each review has different length, we need to set them to a set length, so that we can train the model accurately. Let’s say if we set the length of each review to be 10 words, if one review only contains 5 words, <PAD> will be filled in until the length is increased to 10.

<START>: will be used to indicate the beginning of the review

<UNK>: to replace the unknown words. Since we only imported 10,000 most frequently used words into the datasets, there will be some words that’s unknown. If that’s the case, <UNK> will be inserted.

<UNUSED>: to represents the words that’s unused.

1. Type the following into a new block:



1. Now, let’s decode numbers to words and print the result again. Type the following into a new block:

**Text

Description automatically generated**

We can now read the reviews in texts. Notice that there are unknown words that were replaced with <UNK>

**BUILD THE MODEL**

1. Before building the model, we still need to clean up our dataset. Recall in the previous steps, we created the <PAD>, because we want all the reviews to have the same length and will use <PAD> as padding. Before doing so, let’s check out the length of 2 reviews, to see if they are in fact, different. Type the following into a new block: Graphical user interface, text, application, chat or text message

   Description automatically generated

Seems like the review from the train\_data set is 218, while the review from test\_data set is 260. Let’s set a general cap for the length of the review.

1. Type the following into a new:

Graphical user interface, text, application, Word

Description automatically generated

The code above caps the length of each review to have maximum 256 words. If the number of words of any review is less than 256, <PAD> will be filled in the end of the review till the length gets to 256.

1. Type the following into a new block to make sure the length of our reviews are the same now:

Graphical user interface, text, application, chat or text message

Description automatically generated

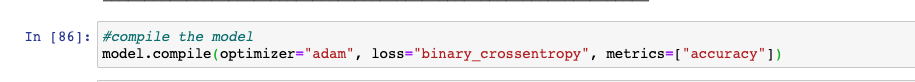
Right now, both sets have their reviews with the same length of 256. We can now go ahead and build the model

1. Type the following into a new block to do so:

Table

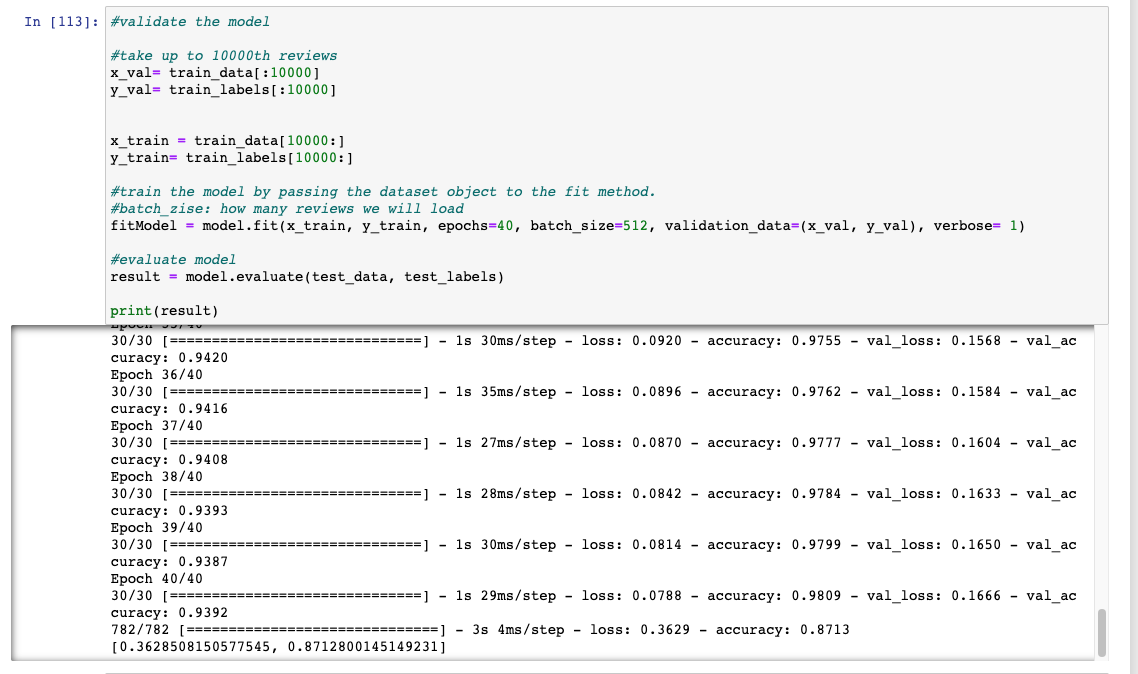
Description automatically generated

1. Type the following into a new block to compile the model:

****

**EVALUATE AND TEST THE MODEL**

1. Type the following into a new block to evaluate our model accuracy and loss:



Based on the result, the accuracy of our model is 87% while the loss is 36%

1. Type the following into a new block to test the model : Graphical user interface, text, application

   Description automatically generated
2. Now, let’s plot the accuracy and loss. Type the following into a new block:

Graphical user interface, text, application

Description automatically generated

1. In a new block, type the following to plot the loss validation:

Graphical user interface

Description automatically generated

1. Type the following into a new block to plot the accuracy validation:

Chart

Description automatically generated

**Push your work to GitHub**

Open the terminal from the VSCode by hitting the “control” + “~” key and type the following command:

>>> git add .

>>> git commit -m “Submission for Module 10 – Your Name”

>>> git push origin master