CourseProject

Please fork this repository and paste the github link of your fork on Microsoft CMT. Detailed instructions are on Coursera under Week 1: Course Project Overview/Week 9 Activities.

Project Presentation

https://mediaspace.illinois.edu/media/t/1_k2m1d94o

Project Documentation

The purpose of this project is to train a binary sentiment classifier using the Naïve Bayes algorithm. At the end, a unigram model will be produced that has the ability to identify if a single review is positive/negative, or if something is positively/negatively reviewed based on a collection of reviews.

Overview of Functions

This application can be used to perform sentimental analysis on reviews found on the internet. When considering a new product or service, the first thought people have is usually to read some reviews online, then form a sentiment towards it. If the specific product/service is popular enough, there are likely more than thousands of reviews online. This application processes reviews and performs sentiment analysis on them. There are two modes in this application, single review mode and collection review mode.

Single Review Mode

Single review mode is used to identify if one specific review is positive or negative. In this case, the input of the application would be a block of text, and the output is a binary statement of either positive or negative.

A possible application for this is to quickly come to a conclusion for long reviews on the internet. For example, this can be used on long review articles online. Rather than reading through it all, one can just copy and paste the text of the article into this application, and get the sentiment of the author right away.

Collection Review Mode

Collection review mode is used to determine if a product/service is positively or negatively reviewed. The input is a series of reviews on the specific product/service. The application will go through each review in the collection, and output the number of positive and negative reviews, and give out a binary conclusion on whether this product/service is positively or negatively reviewed based on the number of positive and negative reviews.

This can be used to obtain the public's opinion on the desired product/service. For example, rather than reading through all the reviews on IGN for a particular game, one can input every review into this application and obtain the overall sentiment of the game.

Implementation Documentation

There are three main files in the source code: - main.py: The main driver of this application. - unigram.py: This includes all the algorithms used to produce the unigram model and performs the sentiment analysis on the reviews. - util.py: This includes the functions used to load all the data used.

main.py

main.py is called in order to run the application. When calling this function, the mode, training set, and documents to be reviewed can be specified through in-line commands. The program first trains the unigram model using the training set. Depend on which mode is selected, the program then performs the desired tasks on the documents selected.

unigram.py

This is where the main algorithms of this application is stored. This includes functions to train the unigram model, and functions to perform sentiment analysis on the input text.

The application will first take in the training set and its corresponding labels to generate two lists of words. One for the probability of words that show up in the negative reviews, and another one for positive reviews. Those two lists and are then used to match with each word in the input reviews to determine if a particle review is more positive or negative.

```
wordProbability(train_set, train_labels, review, laplace)
```

This function takes in the documents used for training and the predetermined labels that classifies them as a positive or negative review. This function is ran twice, once for positive reviews and one for negative reviews. For each list, the probability of the word showing up in a review of the specified sentiment is recorded. The probability for unseen words is also stored. Laplace smoothing is used to improve the accurate of the model.

```
single(positiveWords, negativeWords, text):
```

This function is called when the mode is set to single review. This will take in the lists of positive and negative words. A txt document will be loaded into the function. With each word in the document, the word is matched from the unigram model of the two lists. If there is a match, the probability will be added onto the corresponding sentiment. If the word has never been seen in the training model, a set constant determined in training will be applied. At the end, the total positive probability will be compared against the total negative probability. This way, the overall sentiment of the input text will be determined.

```
multiple(positiveWords, NegativeWords, dev_set):
```

This function works very similarly with the single function. However, instead of one text, a collection of reviews within the specified fold is inputted as "dev_set". Every review is processed in the same way as above. The function outputs the number of positive reviews and negative reviews within the input folder. It will then determine the sentiment of the product/service based on the number of reviews from each side.

util.py

This is where the functions for loading both the training data and input text into the application. This reads the txt files stored in the data folder in the source code. It tokenizes each review. During the process, words are all converted into lowercase letters. Stemming is also utilized.

```
loadSingle(name, stemming, lower_case):
```

This loads a single txt file from the data folder. This file path is defined by the in-line command statement when calling main.py. In this function, the txt file is turned into a list of words, all lowercase with stemming applied. Stemming and lowercase can be turned off to increase the speed of the application when testing for functions, but doing so will decrease the accuracy a lot. This function returns a list of words.

```
loadDir(name, stemming, lower_case, silently):
```

This works the same way as loadSingle, but for a collection of reviews. The name variable is the file path that directs to the folder where the desired collection of reviews is stored. This returns a list of lists of words.

```
load_trainingset(train_dir, stemming, lower_case, silently)
```

This function is called to load the training set and labels into the application during the training phase. The training data is stored in the train folder within the data folder. The data is separated into negative and positive reviews. This function returns two lists of lists of words, one for all the positive reviews and the other for all the negative reviews.

Usage Documentation

In order to run this application, you'll need:

```
- python3
- project source code: [src.zip](https://github.com/tonydfth/CourseProject/files/7685528/src.zip)
```

Either clone the repository or copy paste the above link into a browser to download the source code.

After downloading the unzipping the project source code (src.zip above), you'll see the three files listed above and the data folder. Currently, the training data is 6000 positive movie reviews and 2000 negative movie reviews. This data is provided by the University of Illinois. This should be changed based on what type of product/service the unigram model is training for.

To checkout the command, copy and paste the code below in a terminal window:

```
python3 main.py -h
```

You'll see an option to select single or collection review, and options to change the file path for training and testing data. The training data file path is default to be the data/train folder. The testing data needs to be specified every time the application is ran.

Within the data folder, there are four pre-existing test cases. A negative and a positive review for single review, and a positive and a negative folder for collection review.

To test run single review:

```
python3 main.py --mode single --source '$filepath'
```

To test run collection review:

```
python3 main.py --mode collection --source '$filepath'
```

Note that the file path for single review should lead to a txt document, and the file path for collection review should lead to a folder of txt documents.