Introduction to Data Science

Course 094201

Lab 8:

Boolean model and Rocchio

Spring 2017

Python Reminder - types

Regular Types:

- Boolean
  - *True / False*
- String
  - *"Hello World"*
- Integer
  - 27
- Float
  - 1.61803

Data structures:

- **List** – a list of values. Each one of them is numbered, starting from zero. You can remove values from the list, and add new values to the end.

- **Tuple** – just like lists, but you can't change their values. The values that you give it first up, are the values that you are stuck with for the rest of the program. Again, each value is numbered starting from zero.

- **Dictionary** – (maps in c++) are similar to what their name suggests - a dictionary. In a dictionary, you have an 'index' of words, and for each of them a definition. In python, the word is called a 'key', and the definition a 'value'. You can add, remove, and modify the values in dictionaries.

Python Reminder – types - examples

- **List**

```
Items= [‘chair’, ‘desk’, ‘hat’]
Print(Items[2])
Items[2] = ‘shirt’
Print(Items[2])
Items.append(‘hat’)
Print(Items)
```

```
hat
shirt
[‘chair’, ‘desk’, ’shirt’ ‘hat’]
```

- **Tuple**

```
Days = (‘Sun’, ‘Mon’, ‘Tue’, ‘Wed’, ‘Thu’, ‘Fri’, ‘Sat’)
Print(Days[2])
Print(len(Days))
Print(‘Sun’ in Days)
```

```
Tue
7
True
```

- **Dictionary**

```
Grades= {‘Ben’:82, ‘Don’:79, ‘Bill’:98}
Print(Grades[‘Ben’])
Grades[‘Ben’] = 88
Print(Grades[‘Ben’])
Grades[‘Roee’] = 99
Del Grades[‘Don’]
Print(Grades.keys())
```

```
82
88
[‘Ben’, ‘Bill’, ‘Roee’]
```

Python Reminder – Opening and reading files

- The open() function:
  - In order to open a file in Python we usually rely on the built -in ***open()*** function.

  file = **open**("filename", "mode") where filename is the file we want to open.
    - "mode"='r': **Read Mode** which is used when the file is being read.

```
file  = open("f.txt","r")
Print(file.readline()) #prints the first line
for line in file: #iterate over the lines
        Print(line)
file.close()
```

This is our new text file

This is our new text file
and this is another line.
Why? Because we can

    - "mode"='w': **Write Mode** which is used to write new information to the file.

```
file  = open("f1.txt","w")
f.write("Hello World") #write to the file
```

    - "mode"='a': **Appending Mode** which is used to add new data to the end of the file.
  - At the end we need to close the file: file.close
  - More: www.pythonforbeginners.com/files/reading-and-writing-files-in-python

Python Reminder – Opening and reading files

- With Statement:
  - You can also work with file objects using the with statement.

with open(“filename”) as file

```
with open(“hello.text”, “r”) as f:
        data = f.readlines()
        for line in data:
                words = line.split()
        print words
```

[“hello”, “world”]

- With ensures that you release resources - any files opened will be closed automatically after you are done.
- With also handles exceptions automatically (later this semester)

The Boolean Model

- Each Document $\vec{d}$ (or general object) is represented by a **binary representation**.
- Each entry in the vector represents the existence of a word in the document.
- The order is ignored.
- In order to rank documents we usually use **a similarity\distance measure**.
- Example:

Given 2 documents: d1="Hello world" and d2="hello you"

$$general\ doc(d) representaion: \left(\begin{cases} 1\ if\ \text{hello}\ in\ d \\ 0\ otherwise \end{cases}, \begin{cases} 1\ if\ \text{world}\ in\ d \\ 0\ otherwise \end{cases}, \begin{cases} 1\ if\ \text{you}\ in\ d \\ 0\ otherwise \end{cases} \right)$$

$$\text{vec}(d1) = (1,1,0)$$
$$\text{vec}(d2) = (1,1,0)$$

Rocchio classification

- Uses centroids to define the boundaries.

- The centroid of a class c is computed as the vector average or center of mass of its members:

$$\overrightarrow{Centroid}\,(C) = \frac{1}{|D_C|} \sum_{d \in D_C} \overrightarrow{vec}(d)$$

- D_C is the set of documents associated with class C.

Rocchio classification - Example

- Assign d5 to a class using Euclidean distance as the distance measure.

	Chinese	Japan	Tokyo	Macao	Beijing	Shanghai	class
d1 (Beijing)	0	0	0	0	1	0	c1
d2 (Shanghai)	0	0	0	0	0	1	c1
d3 (Macao)	0	0	0	1	0	0	c1
d4 (Tokyo Japan)	0	1	1	0	0	0	c2
d5 (Japan Tokyo)	0	1	1	0	0	0	?

- $\overrightarrow{Centroid}\,(C1) = (0,0,0,\frac{1}{3},\frac{1}{3},\frac{1}{3}),\ \overrightarrow{Centroid}\,(C2) = (0,1,1,0,0,0)$
- $0 = dist\left(\overrightarrow{Centroid}\,(C1), d5\right) < dist\left(\overrightarrow{Centroid}\,(C2), d5\right) = \frac{1}{3} \rightarrow C2$

The dataset and the code

- The code and the data can be found at:

/mnt/share/students/LAB8

- Copy everything to your **local folder** and unzip the code:

unzip lab8-students.zip

- This dataset contains sentiment analysis over amazon domain (see readme.txt file in dataset folder).

The dataset and the code

- Sentiment analysis: The process of determining the emotional tone behind a series of words, used to gain an understanding of the attitudes, opinions and emotions expressed within a mention.
- In our case each line contains amazon products reviews and a **class** (0 for **negative tone** and 1 for **positive tone**) separated by a tab.
- **Examples**:
  - I love this thing! 1
  - VERY DISAPPOINTED. 0

- Our goal is to use Rocchio classifier in order to predict weather a given sentence represents a positive tone or negative tone.

Assignment In Class

- Go over the code and document each of the functions.

- See the function “create_words_bank” in “file_reader.py” file as an example.

Assignment

1. Implement the function euclidean_dist in “rocchio_classifier.py” which gets 2 vectors (lists) and returns the Euclidean distance between them.
2. Implement the function predict in “rocchio_classifier.py” which gets the class centroids (dictionary) and a document (list) to classify and returns the predicted class of this document.

Assignment

3. Run main and save the output for the test set in a file called “test_set_output.txt”. (you have the expected out put in the output folder)

4. Create a svm-light format (See the next slide) file for the test set. (you have the expected out put in the output folder)

SVM light format

<line> .=. <target> <feature>:<value> <feature>:<value> ... <feature>:<value> # <info>

<target> .=. +1 | -1 | 0 | <float>

<feature> .=. <integer> | "qid"

<value> .=. <float>

<info> .=. <string>

The target value and each of the feature/value pairs are separated by a space character. Feature/value pairs MUST be ordered by increasing feature number. Features with value zero can be skipped.