Introduction to Data Science

Course 094201

Lab 5:

Term weighting for textual classification and retrieval

Spring 2020

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"

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

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

tf based representation

- $tf_{t,d}$ is the number of occurrences of a term *t* in a document *d*
- While there is a large difference between 0 and 1, the increase in importance of this signal with respect to the topic is not growing linearly
- *tf v*ariants:
  1. Raw count of term *t* in document *d*
  2. ***wf (implement this variant)***

$$wf_{t,d} = 0 \text{ if } tf_{t,d} = 0,\ 1 + \log tf_{t,d} \text{ otherwise}$$

idf

- One of the most important measures of informativeness of a term: its rarity across the whole corpus
  - Widely used in practice in different IR applications today
- Variant 1:

  inverse of the raw count of number of documents the term occurs in ($idf_i = 1/df_i$)
- Variant 2 (widely used):

$$idf_i = \log\left(\frac{n}{df_i}\right)$$

  where n is the total number of documents in the corpus

*tf*idf* based representation

- Assign a *tf*idf* weight to each term *i* in each document *d*

$$w_{i,d} = tf_{i,d} \times \log(n / df_i)$$

$tf_{i,d}$ = frequency of term *i* in document *d*

n = total number of documents

df_i = the number of documents that contain term *i*

- Increases with the number of occurrences *within* a doc
- Increases with the rarity of the term *across* the whole corpus

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	?

- $\overrightarrow{Centroid}\ (C1) = (0,0,0,\frac{1}{3},\frac{1}{3},\frac{1}{3})$, $\overrightarrow{Centroid}\ (C2) = (0,1,1,0,0,0)$
- $0 = dist\left(\overrightarrow{Centroid}\ (C2), d5\right) < dist\left(\overrightarrow{Centroid}\ (C1), d5\right) = 1.53$
  $\rightarrow C2$

Cosine Similarity

- A vector can be *normalized* (given a length of 1) by dividing each of its components by its length – here we use the L_2 norm

$$\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$$

- This maps vectors onto the unit sphere: $\| \vec{d}_j \|_2 = \sqrt{\sum_{i=1}^{n} w_{i,j}^2} = 1$
- There is no bias towards longer documents:

$$Cos(q,d) = \frac{q \cdot d}{||q|| \cdot ||d||} = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2} \sqrt{\sum_{i=1}^{n} d_i^2}}$$

Assignment In Class

- The code and the data can be found in the Moodle.

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

The dataset

- **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 whether a given sentence represents a positive tone or a negative tone.

Text pre-processing

Complete the function **pre_process_word** in “file_reader.py” which gets a word and returns a new word with the following changes:

1. Lowercase: change all the words in the documents to lower case letters.
2. Remove punctuation marks (.,?!:)
3. Stopwords are extremely common words that can be considered noise. E.g.: *the*, *and*, *or*. Stopword removal reduces the dimension of the vectors.

The file “stop_words.txt” contains a list of stop words, use it in order to remove stopwords from the documents (If word is a stop word, return an empty string.).

The Code

- Go over the code and make sure you understand it (documentation of the functions can help).
- Implement the function **euclidean_dist** in "rocchio_classifier.py" which gets 2 vectors (lists) and returns the Euclidean distance between them.
- Implement the function **predict** in "rocchio_classifier.py" which gets a document (list) to classify and returns the predicted class of this document.
- Run main and make sure the accuracy is ~ 0.697