**INT401: Fundamentals of Machine Learning** **Fall Semester**

# Assignment 1: Feature Generation

*Lecturer: Xiaobo Jin* *Unit: INT Dept. of XJTLU*

## 1.1 Objectives

- Learn how to generate the features from the text collections

## 1.2 Introduction

Text categorization is the task on classifying a set of documents into categories from a set of predefined labels. Texts cannot be directly handled by our model. The indexing procedure is the first step that maps a text $d_j$ into a numeric representation during the training and validation. The standard **tfidf** function is used to represent the text. The unique words from English vocabulary are represented as a dimension of the dataset.

## 1.3 TFIDF Representation

### 1.3.1 Preprocessing

- **(20 marks)** Read the text files from 5 subdirectories in **dataset** and split the document text into words (splitting separator is non-alphabet letters).

  Hints: use **os.listdir()** function to index all files; use **str.split()** to split the text into words;

  Please use the encoding format 'Latin1' to open the text files in order to avoid the reading error.

  f = codecs.open(fname,'r',encoding='Latin1')

  Notice: The files from 5 subdirectories consist of ONE text dataset. The names of the subdirectories are the class labels of the text files.

- **(20 marks)** Remove the stopwords from the text collections, which are frequent words that carry no information. Stopwords list are given in the file **stopwords.txt** . Convert all words into their lower case form. Delete all non-alphabet characters from the text.

  Hint: use **set** collection to store all stopwords;

- **(20 marks)** Perform word stemming to remove the word suffix.

  - install the library: nltk (python language)
  - usage: see the following code on how to use Porter stemmer (https://www.nltk.org/howto/stem.html).

```
from nltk.stem.porter import *
stemmer = PorterStemmer()
plurals = ['caresses', 'flies', 'dies', 'mules', 'denied',
'died', 'agreed', 'owned', 'humbled', 'sized',
'meeting', 'stating', 'siezing', 'itemization',
'sensational', 'traditional', 'reference', 'colonizer',
'plotted']
singles = [stemmer.stem(plural) for plural in plurals]
```

Hints: use the following code to remove non-alphabet letter from the lower-case **w** to obtain **wd**

```
wd = re.sub(r'[^a-z]', '', w.lower()).strip()
```

### 1.3.2 (40 marks) TFIDF Representation

The documents are represented as the vector space model. In the vector space model, each document is represented as a vector of words. A collection of documents are represented by a document-by-word matrix $A$

$$A = (a_{ik}) \tag{1.1}$$

where $a_{ik}$ is the weight of word $k$ in document $i$.

TFIDF representation assigns the weight to word $i$ in document $k$ in proportion to the number of occurrences of the word in the document, and inverse proportion to the number of documents in the collection for which the word occurs at least once.

$$a_{ik} = \log(f_{ik} + 1.0) * \log(N/n_k) \tag{1.2}$$

- **(10 marks)** $f_{ik}$: the frequency of word $k$ in document $i$
- $N$: the number of documents in the dataset
- **(10 marks)** $n_k$: the total number of documents that word $k$ occurs in the dataset called the document frequency.

Notice that the entry $a_{ik}$ is 0 if the word $k$ is not included in the document $i$.

**(20 marks)** Taking into account the length of different documents, we normalize the representation of the document as

$$A_{ik} = \frac{a_{ik}}{\sqrt{\sum_{j=1}^{D} a_{ij}^2}} \tag{1.3}$$

The data set can be represent as a matrix $A_{N \times D}$, where $D$ is the number of the unique words in the document collection.

Finally, the dataset is save into .npz file, where $A$ is a matrix represented with the numpy array.

```
np.savez('train-20ng.npz',X=A)
```

## 1.4 Lab Report

- **Write a short report which should contain a concise explanation of your implementation, results and observations.**

  For the score of each step, such as 15 points, the proportion of the three parts to the total score is as follows:

  - Explanation of the execution of this step ( **50%** ): how to design the data structure, how to design the algorithm to realize this step; how do you think about this problem
  - Code and comments ( **30%** ): Whether the code is correct, attach comments to help understand the code
  - Results and interpretation ( **20%** ): Whether the running results are correct, explain the results to a certain extent, or what you find from them. Please insert the clipped running image into your report for each step.

- Submit the report and the python source code with the suitable comments electronically into the learning mall.
- It is highly recommended to use the latex typesetting language to write reports.
- The report in pdf format and python source code of your implementation should be zipped into a single file. The naming of report is as follows:

  e.g. StudentID_LastName_FirstName_LabNumber.zip (123456789_Einstein_Albert_1.zip)

## 1.5 Hints

Please refer to the paper for more details: K Aas and L. Eikvil, Text Categorisation: A Survey, 1999.

- Latex IDE: texstudio
- Python IDE: pycharm
- Use the python set, dict and list collections flexibly.