

**CS 5/7322**  
**Introduction to Natural Language Processing**  
**Spring 2023**

**Programming Homework 1: (TF-IDF)**

Due date: 2/24 (Fri) 11:59pm. Late deadline: 2/27 (Mon) 11:59pm

In NLTK, there is a basic class called `CorpusReader` which allow one to read a set of text files into a structure that can be manipulated. You can see the following documentation for examples:

- <http://www.nltk.org/api/nltk.corpus.reader.html>
- <http://www.nltk.org/howto/corpus.html>

The basic corpus class in NLTK does not support TF-IDF representation of document. Your task for this project is to build a class that incorporate a corpus reader that also produce the corresponding TF-IDF representation.

***To-do list***

You are to define a class called `CorpusReader_TFIDF`. The class will take a corpus object in NLTK and construct the td-idf vector for each document.

The specific details of the class is as follows:

***Constructor***

`CorpusReader_TFIDF(corpus, tf = "raw", idf = "base", stopWord = "none", toStem = false, ignoreCase = true)`

- `corpus` (required): a corpus object in NLTK
- `tf` : the method used to calculate term frequency. The following values are supported
  - "raw" (default) = raw term frequency
  - "log" : log normalized ( $1 + \log(\text{frequency})$ ) if frequency > 0; 0 otherwise)
- `Idf` : the method used to calculate the inverse document frequency
  - "base" (default) : basic inverse frequency
  - "smooth": inverse frequency smoothed
- `stopWord`: what stopWords to remove
  - "none" : no stopwords need to be removed
  - "standard": use the standard English stopWord available in NLTK
  - Others: this should treat as a filename where stopwords are to be read. You should assume any word inside the stopwords file is a stopword.
- `toStem`: if true, use the Snowball stemmer to stem the words beforehand
- `ignoreCase`: if true, ignore the case of the word (i.e. "Apple", "apple", "APPLE" are the same word). In such case, represent the word as the all lower-case version.

*Shared Methods in the corpus reader class.*

- See Table 1.3 in <http://www.nltk.org/book/ch02.html>. You just need to call the corresponding method for the corpus reader associated with it, and return the results. You need to support the following methods: `fields()`, `raw()`, `raw(fileids=[...])`, `words()`, `words(fileids=[...])`

*Methods specific to CorpusReader\_TFIDF*

- `tfidf(fileid, returnZero = false)` : return the TF-IDF for the specific document in the corpus (specified by `fileid`). The vector is represented by a dictionary/hash in python. The keys are the terms, and the values are the tf-idf value of the dimension. If `returnZero` is true, then the dictionary will contain terms that have 0 value for that vector, otherwise the vector will omit those terms
- `tfidfAll(returnZero = false)` : return the TF-IDF for all documents in the corpus. It will be returned as a dictionary. The key is the `fileid` of each document, for each document the value is the `tfidf` of that document (using the same format as above).
- `tfidfNew([words])` : return the tf-idf of a “new” document, represented by a list of words. You should honor the various parameters (`ignoreCase`, `toStem` etc.) when preprocessing the new document. Also, the idf of each word should not be changed (i.e. the “new” document should not be treated as part of the corpus).
- `idf()` : return the idf of each term as a dictionary : keys are the terms, and values are the idf
- `cosine_sim([fileid1, fileid2])` return the cosine similarity between two documents in the corpus
- `cosine_sim_new([words], fileid)`: return the cosine similarity between a “new” document (as if specified like the `tfidf_new()` method) and the documents specified by `fileid`.

You will be given a program and you should implement the class such that the program can be run without modification.

### **Bonus (15%)**

Implement the following method:

- `query([words])` : return a list of (document, `cosine_sim`) tuples that calculate the cosine similarity between the “new” document (specified by the list of words as the document). The list should be ordered in decreasing order of cosine similarity.

### **What to hand in**

All you need to hand in is the `CorpusReader_TFIDF.py` file to Canvas.