4. TF-IDF

March 29, 2022

[11]:
```
#Latent Semantic Indexing/Analysis (LSI/LSA)
#Latent = hidden, Semantic = meaning
# LSA aims to find hidden meaning/relations in text documents.
# LSA is a technique of analyzing relationships between a set of documents and␣
 ↪the terms they contain.
# A matrix containing word counts per document (rows represent unique words and␣
 ↪columns represent each document) is constructed from a large piece of text

# TF-IDF is most popular method of LSI
# TF-IDF: tf * idf
# tf(t) = freq(t) / total_terms
# idf(t) = idf(t) = log_e [ (1 + n) / (1 + df(t)) ] + 1

# standardTFIDF vs sklearnTFIDF
# https://towardsdatascience.com/
 ↪how-sklearns-tf-idf-is-different-from-the-standard-tf-idf-275fa582e73d
```

[12]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
corpus = [ 'This is the first document.', 'This document is the second document.
 ↪', 'And this is the third one.', 'Is this the first document?']

#vectorizer = TfidfVectorizer(norm='l2',use_idf=False) # gives TF only with L1␣
 ↪normalization
vectorizer = TfidfVectorizer()   # gives TF-IDF with L2 normalization of TF

X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names())
```

```
['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']
```

[13]:
```
print(X.shape)
print(X)
```

```
(4, 9)
  (0, 1)        0.46979138557992045
  (0, 2)        0.5802858236844359
  (0, 6)        0.38408524091481483
  (0, 3)        0.38408524091481483
```

```
  (0, 8)        0.38408524091481483
  (1, 5)        0.5386476208856763
  (1, 1)        0.6876235979836938
  (1, 6)        0.281088674033753
  (1, 3)        0.281088674033753
  (1, 8)        0.281088674033753
  (2, 4)        0.511848512707169
  (2, 7)        0.511848512707169
  (2, 0)        0.511848512707169
  (2, 6)        0.267103787642168
  (2, 3)        0.267103787642168
  (2, 8)        0.267103787642168
  (3, 1)        0.46979138557992045
  (3, 2)        0.5802858236844359
  (3, 6)        0.38408524091481483
  (3, 3)        0.38408524091481483
  (3, 8)        0.38408524091481483
```

[14]:
```
# There are 4 documents/sentences - see corpus list above
# There are 9 unique words in the corpus
# So X has dimensions 4X9
# (0,8) will refer to "this" i.e. 8th word of the vector in 0th sentence
# (0,3) will refer to "is" i.e. 3rd word of the vector in 0th sentence
# (0,6) will refer to "the" i.e. 6th word of the vector in 0th sentence
# (0,2) will refer to "first" i.e. 2nd word of the vector in 0th sentence
# (0,1) will refer to "document" i.e. 1st word of the vector in 0th sentence
# tf(0,8) = 1/5  = 0.2 (as this appears only once in first document containing␣
 ↪5 words)
# TF(0,8) = 0.2 / sqrt(0.04+0.04+0.04+0.04+0.04) = 0.2 / sqrt(0.2) = sqrt(0.2)␣
 ↪= 0.447213595
# IDF(0,8) = log_e(5/5)+1 = 1
# TF-IDF(0,8) = TF(0,8)*IDF(0,8) =  0.447213595 * 1 = 0.447213595

# tf(1,5) = 1/6 = 0.16
# TF(1,5) = 0.16 / sqrt(0.0277+0.111+0.0277+0.0277+0.0277) = 0.
 ↪35355339059327373
# IDF(1,5) = log_e(5/2)+1 = 1.916290731874155
# TF-IDF(1,5) = 0.35355339059327373 * 1.916290731874155 = 0.67751108561657736
# noarmalized(TFIDF) = 0.5386476208856763
```