Term Frequency(TF)

Term Frequency also known as TF measures the number of times a term (word) occurs in a document.

```
import math
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer


#documents
doc1 = "create a session"
doc2 = "remove a session"
doc3 = "invalidate a session"
```

Machine learning algorithms cannot work with raw text directly. Rather, the text must be converted into vectors of numbers. In natural language processing, a common technique for extracting features from text is to place all of the words that occur in the text in a bucket. This aproach is called a bag of words model or BoW for short. It’s referred to as a “bag” of words because any information about the structure of the sentence is lost.

```
bagOfWords1 = doc1.split(' ')
bagOfWords2 = doc2.split(' ')
bagOfWords3 = doc3.split(' ')
print(bagOfWords1)
print(bagOfWords2)
print(bagOfWords3)
```

```
['login']
['forgot', 'password']
['change', 'password']
```

```
uniqueWords = set(bagOfWords1).union(set(bagOfWords2)).union(set(bagOfWords3))
uniqueWords
```

```
{'change', 'forgot', 'login', 'password'}
```

```
numOfWords1 = dict.fromkeys(uniqueWords, 0)
for word in bagOfWords1:
     numOfWords1[word] += 1
numOfWords2 = dict.fromkeys(uniqueWords, 0)
for word in bagOfWords2:
    numOfWords2[word] += 1
numOfWords3 = dict.fromkeys(uniqueWords, 0)
for word in bagOfWords3:
    numOfWords3[word] += 1
```

```
print(numOfWords1)
```

```
{'password': 0, 'change': 0, 'login': 1, 'forgot': 0}
```

Another problem with the bag of words approach is that it doesn't account for noise. In other words, certain words are used to formulate sentences but do not add any semantic meaning to the text. For example, the most commonly used word in the english language is the which represents 7% of all words written or spoken. You couldn't make deduce anything about a text given the fact that it contains the word the. On the other hand, words like good and awesome could be used to determine whether a rating was positive or not.

In natural language processing, useless words are referred to as stop words. The python natural language toolkit library provides a list of english stop words.

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
from nltk.corpus import stopwords
stopwords.words('english')
```

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
```

```
      'herself',
      'it',
      "it's",
      'its',
      'itself',
      'they',
      'them',
      'their',
      'theirs',
      'themselves',
      'what',
      'which',
      'who',
      'whom',
      'this',
      'that',
      "that'll",
      'these',
      'those',
      'am',
      'is',
      'are',
      'was',
      'were',
      'be',
      'been',
      'being',
      'have',
      'has',
      'had',
      'having',
      'do',
      'does',
```

1)In reality each document will be of different size. On a large document the frequency of the terms will be much higher than the smaller ones. Hence we need to normalize the document based on its size.

2)A simple trick is to divide the term frequency by the total number of terms.

For example in Document 1 the term game occurs two times. The total number of terms in the document is 10. Hence the normalized term frequency is 2 / 10 = 0.2.

```
def computeTF(wordDict, bagOfWords):
    tfDict = {}
    bagOfWordsCount = len(bagOfWords)
    for word, count in wordDict.items():
        tfDict[word] = count /float(bagOfWordsCount)
    return tfDict


tf1 = computeTF(numOfWords1, bagOfWords1)
tf2 = computeTF(numOfWords2, bagOfWords2)
tf3 = computeTF(numOfWords3, bagOfWords3)
```

```
print(tf1)
print(tf2)
print(tf3)
```

```
{'password': 0.0, 'change': 0.0, 'login': 1.0, 'forgot': 0.0}
{'password': 0.5, 'change': 0.0, 'login': 0.0, 'forgot': 0.5}
{'password': 0.5, 'change': 0.5, 'login': 0.0, 'forgot': 0.0}
```

Inverse Data Frequency (IDF)

Inverse data frequency determines the weight of rare words across all documents in the corpus.

```
def computeIDF(documents):
    import math
    N = len(documents)

    idfDict = dict.fromkeys(documents[0].keys(), 0)
    for document in documents:
        for word, val in document.items():
            if val > 0:
                idfDict[word] += 1

    for word, val in idfDict.items():
        idfDict[word] = math.log(N / float(val))
    return idfDict


idfs = computeIDF([numOfWords1, numOfWords2,numOfWords3])
print(idfs)
```

```
{'password': 0.4054651081081644, 'change': 1.0986122886681098, 'login': 1.09861
```

Lastly, the TF-IDF is simply the TF multiplied by IDF.

```
def computeTFIDF(tfBagOfWords, idfs):
    tfidf = {}
    for word, val in tfBagOfWords.items():
        tfidf[word] = val * idfs[word]
    return tfidf


tfidf1 = computeTFIDF(tf1, idfs)
tfidf2 = computeTFIDF(tf2, idfs)
tfidf3 = computeTFIDF(tf3, idfs)
df = pd.DataFrame([tfidf1, tfidf2,tfidf3])


print(df)
```

```
   password    change     login    forgot
0  0.000000  0.000000  1.098612  0.000000
```

```
1  0.202733  0.000000  0.000000  0.549306
2  0.202733  0.549306  0.000000  0.000000
```

This is the output we will get when we perform the fit function.(above one)

We are coding the fit and transform the function of TFIDFVectorizer. With Tfidfvectorizer you compute the word counts, idf and tf-idf values all at **once**

```
vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform([doc1, doc2,doc3])
feature_names = vectorizer.get_feature_names()
dense = vectors.todense()
denselist = dense.tolist()
df = pd.DataFrame(denselist, columns=feature_names)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureW
  warnings.warn(msg, category=FutureWarning)
```

```
print(df)
```

```
     change    forgot  login  password
0  0.000000  0.000000    1.0  0.000000
1  0.000000  0.795961    0.0  0.605349
2  0.795961  0.000000    0.0  0.605349
```

TF-IDF is scored between 0 and 1. The higher the numerical weight value, the rarer the term. The smaller the weight, the more common the term. Here the rare term is login

TF-IDF has several limitations: –

–It computes document similarity directly in the word-count space, which may be slow for large vocabularies.

– It assumes that the counts of different words provide independent evidence of similarity.

– It makes no use of semantic similarities between words

Advantage: –

The biggest advantages of TF-IDF come from how simple and easy to use it is.

LDA SIMILARITY

Double-click (or enter) to edit

```
# Importing modules
import pandas as pd
import os
from google.colab import files
# uploaded = files.upload()
os.chdir('..')


path="/content/USECAS1.csv"

papers = pd.read_csv(path)
# Read data into papers
# papers = pd.read_csv('USECAS.csv')

# Print head
papers.head(22)
```

	NO	Description
0	Use Case1	Browsing the Product List
1	Use Case2	Browsing the Catalog
2	Use Case3	Searching the Catalog
3	Use Case4	Browsing the items List

```
from google.colab import drive
drive.mount('/content/drive')
```

5	Use Case6	Update items in the Cart

```
# Load the regular expression library
import re

# Remove punctuation
papers['paper_desc_processed'] = \
papers['Description'].map(lambda x: re.sub('[,\.!?]', '', x))

# Convert the titles to lowercase
papers['paper_desc_processed'] = \
papers['paper_desc_processed'].map(lambda x: x.lower())

# Print out the first rows of papers
papers['paper_desc_processed'].head(22)
```

```
<>:6: DeprecationWarning: invalid escape sequence \.
<>:6: DeprecationWarning: invalid escape sequence \.
<>:6: DeprecationWarning: invalid escape sequence \.
<ipython-input-14-c202943d189e>:6: DeprecationWarning: invalid escape sequence
  papers['Description'].map(lambda x: re.sub('[,\.!?]', '', x))
0     browsing the product list
1          browsing the catalog
2         searching the catalog
3       browsing the items list
4                add  cart items
5      update items in the cart
6        remove items from cart
7                view cart items
8                   make payment
9               list order items
10             view order status
11                 confirm order
12              get total amount
13        change shipping info
14                    signing up
15                    signing in
16                       signout
17                   manage item
18                manage account
19                  manage order
20      manage category/product
Name: paper_desc_processed, dtype: object
```

```
# Import the wordcloud library
from wordcloud import WordCloud
```

```
# Join the different processed titles together.
long_string = ','.join(list(papers['paper_desc_processed'].values))

# Create a WordCloud object
wordcloud = WordCloud(background_color="white", max_words=1000, contour_width=3, contour_c

# Generate a word cloud
wordcloud.generate(long_string)

# Visualize the word cloud
wordcloud.to_image()
```


```
import gensim
from gensim.utils import simple_preprocess
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

stop_words = stopwords.words('english')
stop_words.extend(['from', 'subject', 're', 'edu', 'use'])

def sent_to_words(sentences):
    for sentence in sentences:
        # deacc=True removes punctuations
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))

def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc))
             if word not in stop_words] for doc in texts]


data = papers.paper_desc_processed.values.tolist()
data_words = list(sent_to_words(data))

# remove stop words
data_words = remove_stopwords(data_words)

print(data_words)
```

```
/usr/local/lib/python3.7/dist-packages/scipy/sparse/sparsetools.py:21: Deprecat
scipy.sparse.sparsetools is a private module for scipy.sparse, and should not b
  _deprecated()
[['browsing', 'product', 'list'], ['browsing', 'catalog'], ['searching', 'catal
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
import gensim.corpora as corpora

# Create Dictionary
id2word = corpora.Dictionary(data_words)

# Create Corpus
texts = data_words

# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]

# View
print(corpus)
```

```
[[(0, 1), (1, 1), (2, 1)], [(0, 1), (3, 1)], [(3, 1), (4, 1)], [(0, 1), (1, 1),
```

```
from pprint import pprint

# number of topics
num_topics = 4

# Build LDA model
lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                       id2word=id2word,
                                       num_topics=num_topics)

# Print the Keyword in the 10 topics
pprint(lda_model.print_topics())
doc_lda = lda_model[corpus]
```

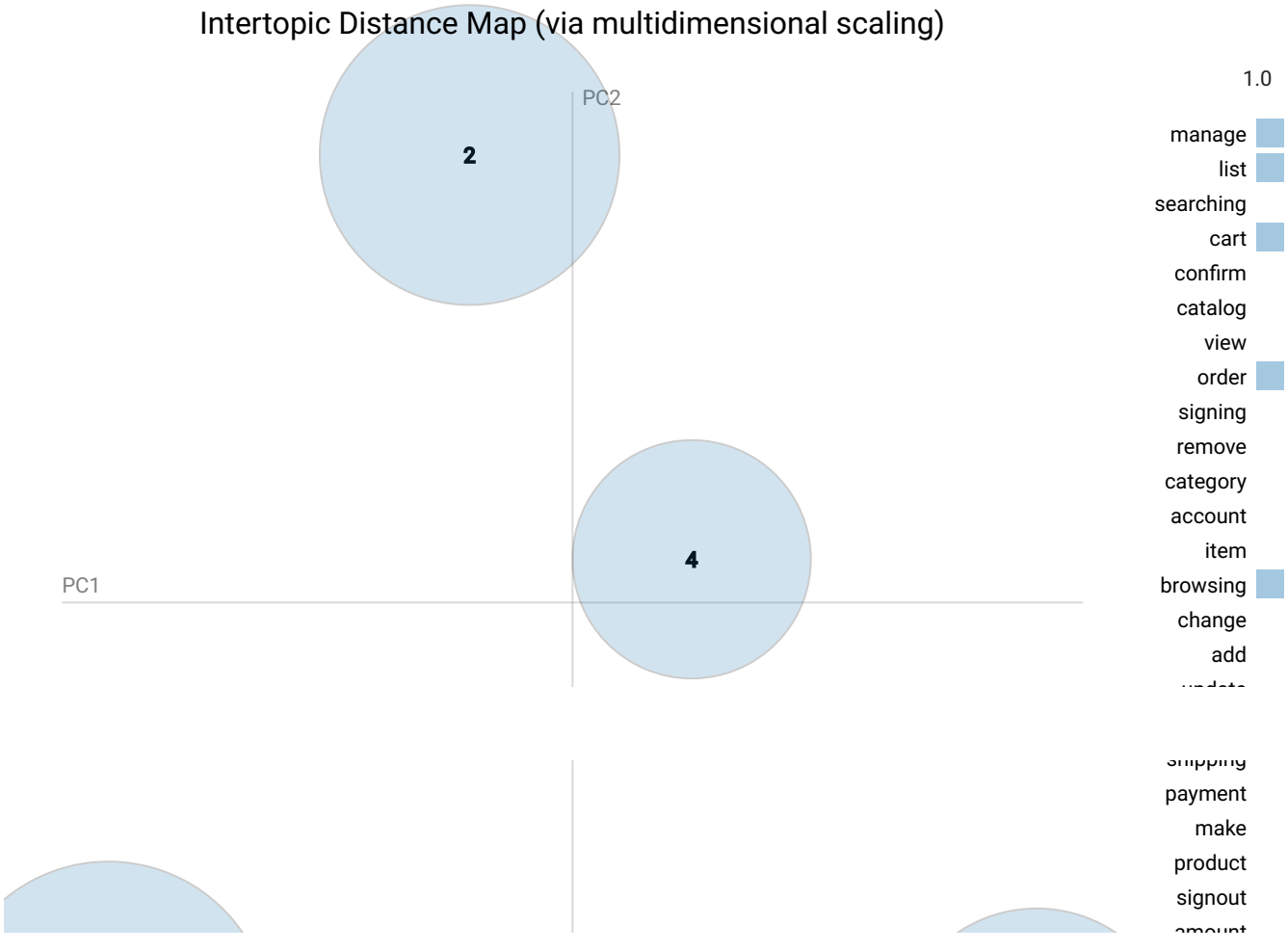
```
WARNING:gensim.models.ldamulticore:too few updates, training might not converge
[(0,
  '0.218*"manage" + 0.066*"items" + 0.065*"cart" + 0.065*"order" + '
  '0.065*"product" + 0.065*"remove" + 0.065*"category" + 0.064*"account" + '
  '0.064*"item" + 0.016*"signing"'),
 (1,
  '0.089*"items" + 0.088*"cart" + 0.088*"order" + 0.087*"view" + '
  '0.087*"catalog" + 0.086*"searching" + 0.086*"confirm" + 0.020*"signing" + '
  '0.020*"manage" + 0.019*"signout"'),
 (2,
  '0.134*"list" + 0.094*"items" + 0.093*"order" + 0.093*"browsing" + '
  '0.088*"signing" + 0.052*"view" + 0.052*"amount" + 0.052*"total" + '
  '0.052*"get" + 0.052*"product"'),
 (3,
  '0.107*"items" + 0.106*"cart" + 0.059*"browsing" + 0.059*"change" + '
  '0.059*"catalog" + 0.059*"add" + 0.059*"update" + 0.059*"info" + '
  '0.058*"shipping" + 0.058*"payment"')]
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
```

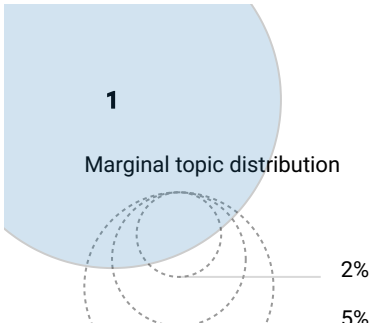
```
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
  score += np.sum(cnt * logsumexp(Elogthetad + Elogbeta[:, int(id)]) for id, cn
/usr/local/lib/python3.7/dist-packages/gensim/models/ldamodel.py:1077: Deprecat
```

```
!pip install pyLDAvis
import pyLDAvis
import pyLDAvis.gensim_models
import pickle


# Visualize the topics
pyLDAvis.enable_notebook()


vis=pyLDAvis.gensim_models.prepare(lda_model,corpus,id2word, mds="mmds",R=30)
vis
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wh
Requirement already satisfied: pyLDAvis in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: numexpr in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: gensim in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: funcy in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy>=1.20.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: sklearn in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: pandas>=1.2.0 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: future in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.7/di
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dis
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packa
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/pytho
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7
/usr/local/lib/python3.7/dist-packages/pyLDAvis/_prepare.py:247: FutureWarning:
  by='saliency', ascending=False).head(R).drop('saliency', 1)
```

Selected Topic: 0 Previous Topic Next Topic Clear Topic Sli





Marginal topic distribution