python

Python Programming

Natural language processing in Python using NLTK

UMKC

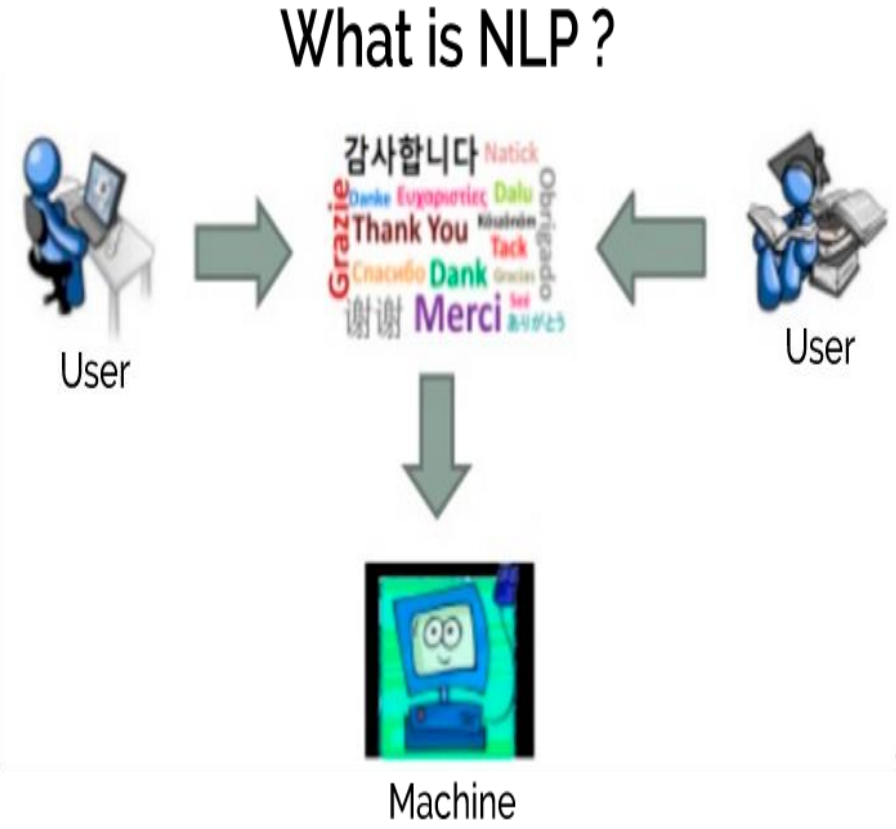
Feedback is greatly appreciated!

Objective

- NLP (Natural language processing)
- NLTK (Natural Language Toolkit)

Natural Language Processing

- Computer aided text analysis of human language
- The goal is to enable machines to understand human language and extract meaning from text
- The "Natural Language Toolkit" is a python module that provides a variety of functionality that will aid us in processing text

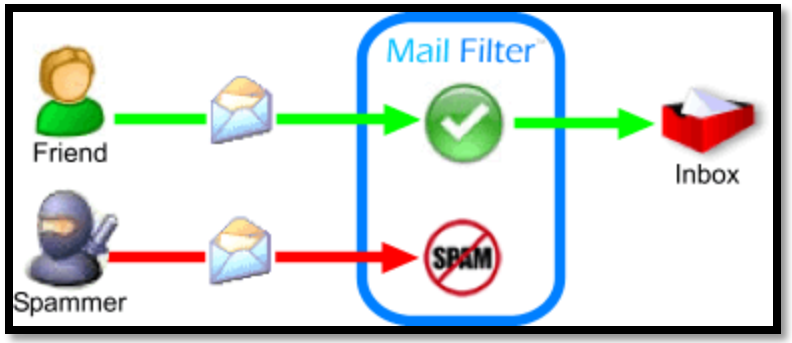


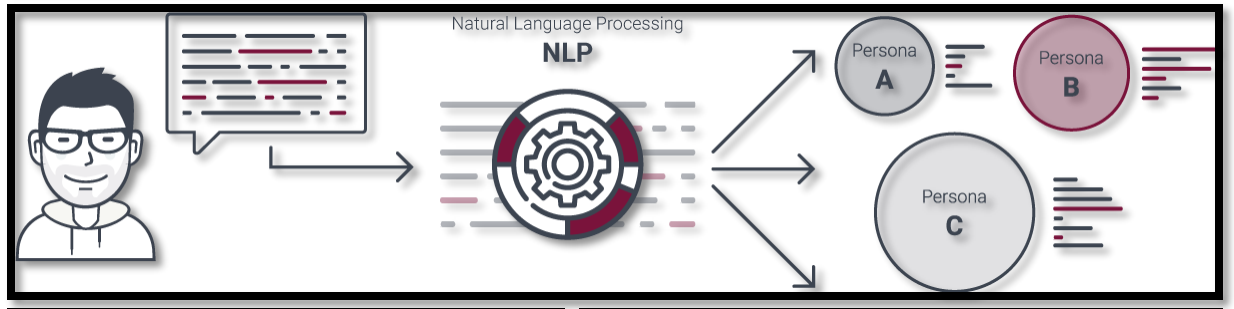
https://www.upwork.com/hiring/for-clients/artificial-intelligence-and-natural-language-processing-in-big-data/

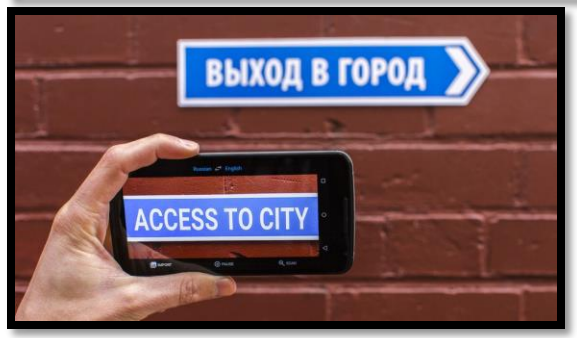
UMKC

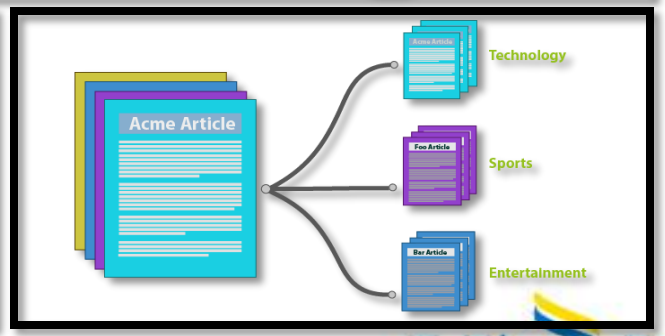
NLP Applications

- Consumer behavior Analysis
- Site recommendation
- Spam filtering
- Detecting fake news
- Automatic Summarization (to generate summary of given text) and Machine Translation (translation of one language into another)









UMKC

NLP Libraries

- NLTK (Natural Language Toolkit)
- SpaCy
- Stanford NLP
- Apache OpenNLP

https://opensource.com/business/15/7/five-open-source-nlp-tools

UMKC

NLTK (Natural Language Toolkit)

An open source library which simplifies the implementation of Natural Language Processing(NLP) in Python.

https://www.nltk.org/

UMKC

Download and Installation

For Anaconda and Python3.6

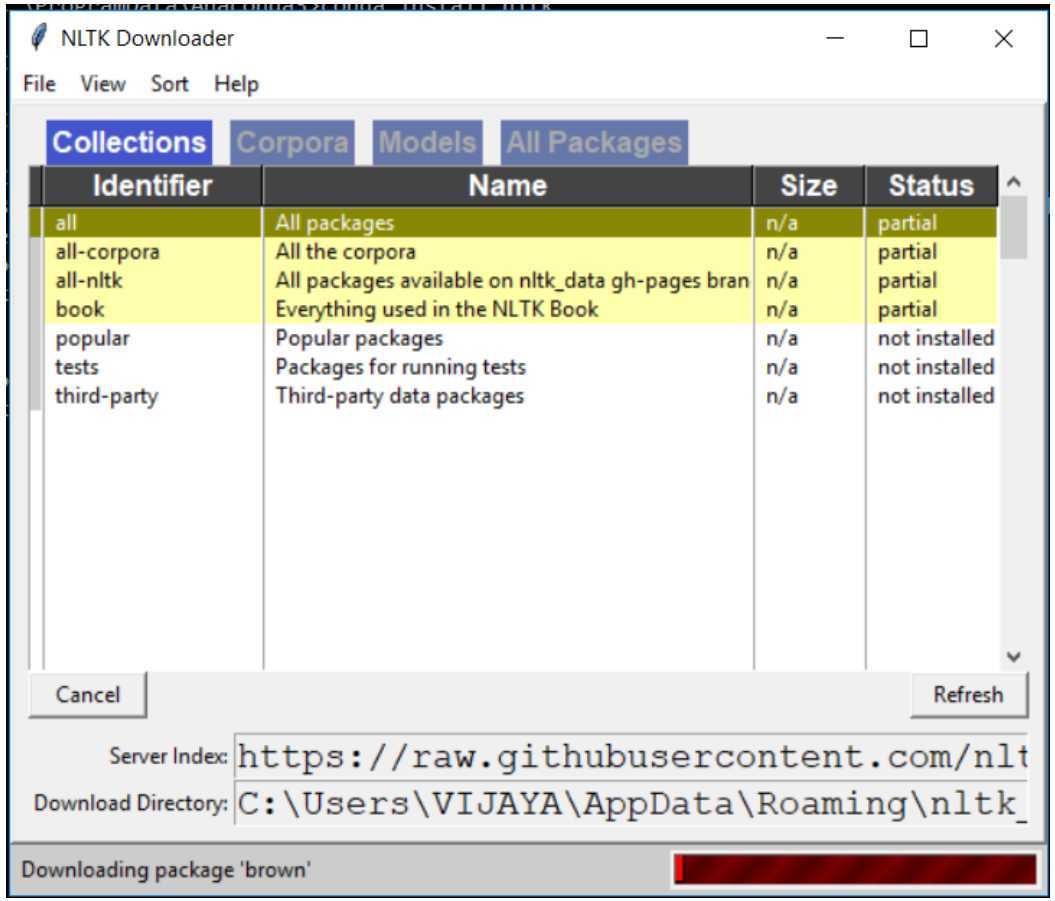
Run the Python interpreter and type the commands:

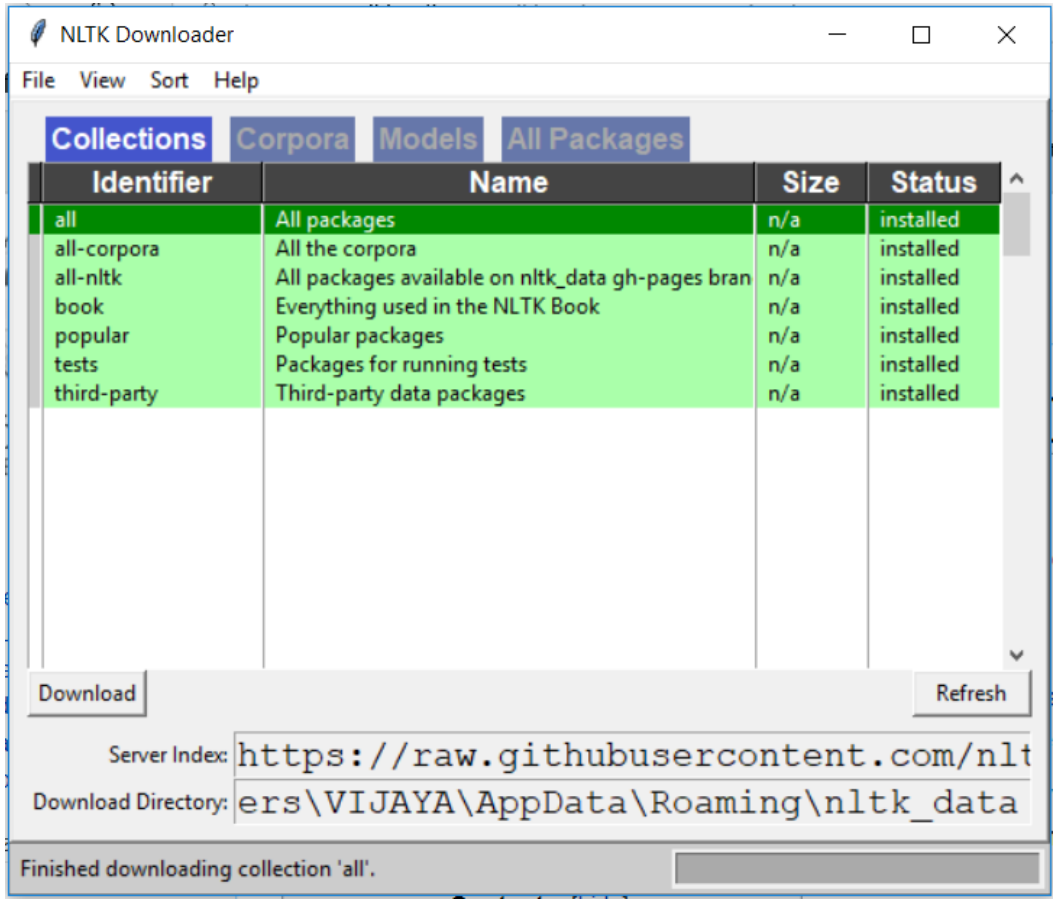
- > conda install nltk
- >python
- >>> import nltk
- >>>nltk.download()

```
(base) C:\ProgramData\Anaconda3>conda install nltk
Solving environment: -
(base) C:\ProgramData\Anaconda3>set KERAS_BACKEND=

(base) C:\ProgramData\Anaconda3>set "KERAS_BACKEND=theano"

(base) C:\ProgramData\Anaconda3>python
Python 3.6.3 |Anaconda custom (64-bit)| (default, Oct 15 2017, 03:27:45) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import nltk
>>> nltk.download()
showing info https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml
```

https://www.nltk.org/

Download and Installation





https://www.nltk.org/

UMKC

Download and Installation

For pip or other

- Installation instructions at: http://www.nltk.org/
- Run the Python interpreter and type the commands:

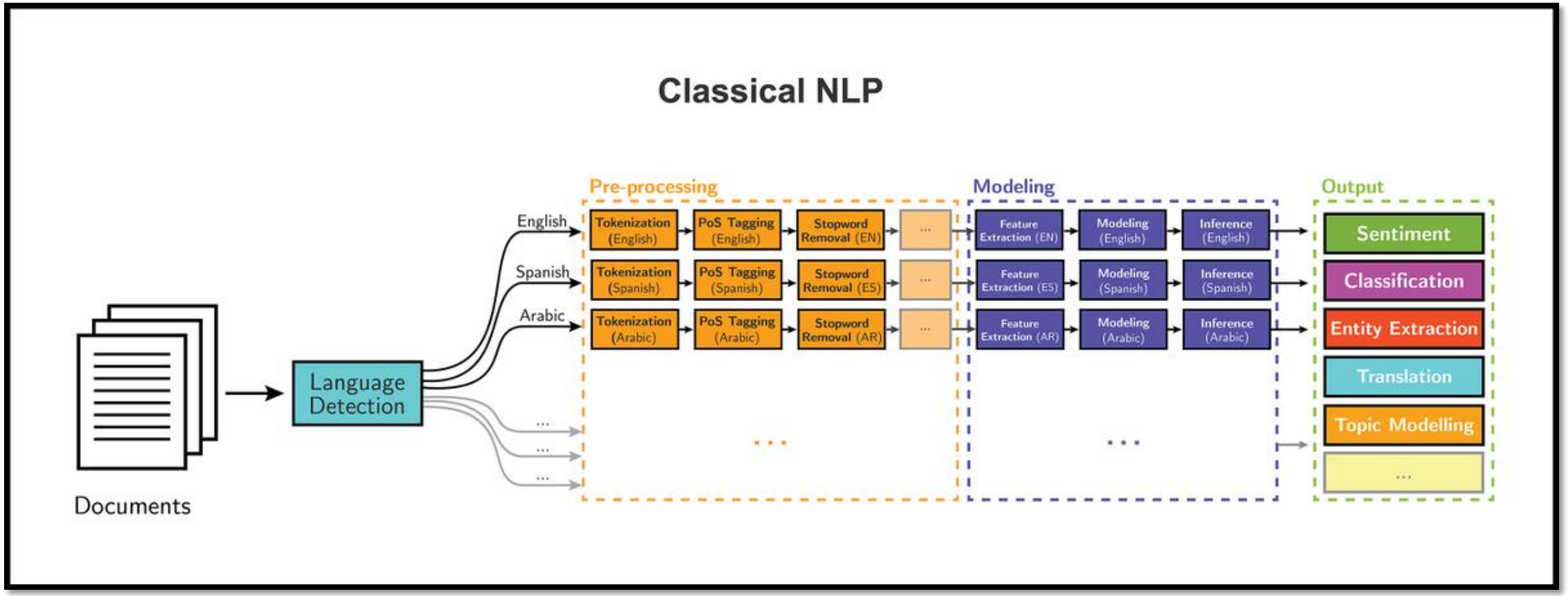
```
>>> import nltk
>>> nltk.download()
```

- Opens the NLTK downloader, you can choose what to download.

https://www.nltk.org/

UMKC

Basic NLP pipeline



https://www.nltk.org/

Text corpora

- Corpus
  - Large collection of text
  - Raw or categorized
  - Concentrate on a topic or open domain
- The plural form of corpus is **corpora**

- Examples
  - Brown Corpus - first, largest corpus, categorized by genre such as news, editorial, and so on
  - Web and Chat Text - discussion forum, reviews, etc.
  - Reuters Corpus - news, 90 topics
  - Universal Declaration of Human Rights (UDHR) corpus – multilingual (372)

https://www.nltk.org/book/ch02.html

UMKC

Working with brown corpus example

```
import nltk
nltk.download("brown")
from nltk.corpus import brown
print(brown.categories())
print(brown.words(categories ='news'))
```

['adventure', 'belles_lettres', 'editorial', 'fiction', 'government', 'hobbies', 'humor', 'learned', 'lore', 'mystery', 'news', 'religion', 'reviews', 'romance', 'science_fiction']
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]

WordNet

- WordNet is a lexical database for the English language
- It groups English words into sets of synonyms called synsets
- WordNet is freely and publicly available for download
- WordNet's structure makes it a useful tool for natural language processing
- The main relation among words in WordNet is synonymy, as between the words car and automobile

https://en.wikipedia.org/wiki/WordNet

UMKC

WordNet: Example

```
import nltk
from nltk.corpus import wordnet as wn

synsets = wn.synsets('phone')

for syns in synsets:
    print(str(syns.definition))
```

```
<bound method Synset.definition of Synset('telephone.n.01')>
<bound method Synset.definition of Synset('phone.n.02')>
<bound method Synset.definition of Synset('earphone.n.01')>
<bound method Synset.definition of Synset('call.v.03')>
```

1) 'electronic equipment that converts sound into electrical signals that can be transmitted over distances and then converts received signals back into sounds‘
2) '(phonetics) an individual sound unit of speech without concern as to whether or not it is a phoneme of some language‘
3) 'electro-acoustic transducer for converting electric signals into sounds; it is held over or inserted into the ear‘
4) 'get or try to get into communication (with someone) by telephone'

https://wordnet.princeton.edu/

UMKC

Steps for natural language processing

Tokenization

- Tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens.
- The tokens may be the sentences, words, numbers or punctuation marks.

Example:
I/P: Mango, banana, pineapple and apple all are fruits.
O/P: Mango | Banana | Pineapple | and | Apple | all | are | Fruits

https://www.tutorialspoint.com/artificial_intelligence_with_python/artificial_intelligence_with_python_nltk_package.htm

UMKC

Tokenization: Example

```
import nltk

sentence = "At eight o'clock on Thursday morning. Arthur didn't feel very good."
stokens = nltk.sent_tokenize(sentence)
wtokens = nltk.word_tokenize(sentence)

for s in stokens:
    print(s)

for t in wtokens:
    print(t)
```

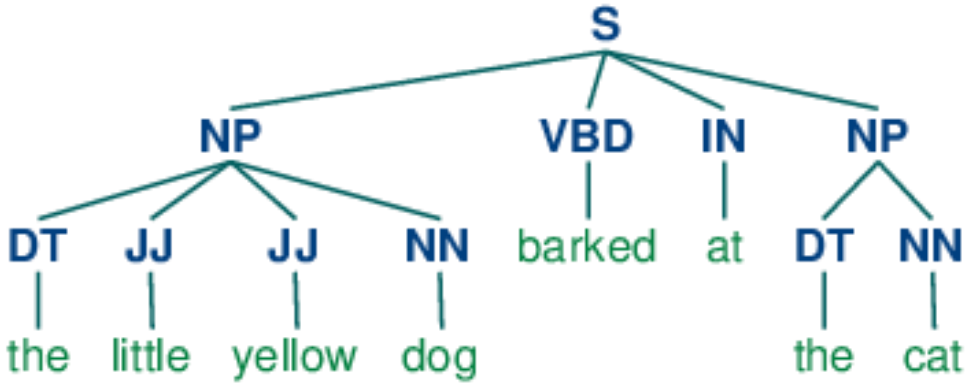
Input

```
At eight o'clock on Thursday morning.
Arthur didn't feel very good.
At
eight
o'clock
on
Thursday
morning
.
Arthur
did
n't
feel
very
good
.
```

Output

Part of Speech tagging

- The process of classifying the words in a text(corpus) into their parts of speech and labeling them accordingly is known as part-of-speech tagging, POS-tagging, or simply tagging
- Parts of speech are also known as word classes or lexical categories
- The collection of tags used for a particular task is known as a tagset
- In English the main parts of speech are noun, pronoun, adjective, determiner, verb, adverb, preposition, conjunction, and interjection



Penn Treebank Part-of-Speech Tags

TAG	DESCRIPTION	EXAMPLE
CC	conjunction, coordinating	*and, or, but*
CD	cardinal number	*five, three, 13%*
DT	determiner	*the, a, these*
EX	existential there	*there were six boys*
FW	foreign word	*mais*
IN	conjunction, subordinating or preposition	*of, on, before, unless*
JJ	adjective	*nice, easy*
JJR	adjective, comparative	*nicer, easier*
JJS	adjective, superlative	*nicest, easiest*
LS	list item marker	
MD	verb, modal auxillary	*may, should*
NN	noun, singular or mass	*tiger, chair, laughter*
NNS	noun, plural	*tigers, chairs, insects*
NNP	noun, proper singular	*Germany, God, Alice*
NNPS	noun, proper plural	*we met two Christmases ago*

https://www.clips.uantwerpen.be/pages/mbsp-tags
https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

UMKC

Part of Speech tagging: Example

```
import nltk

text = nltk.word_tokenize("And now for something completely different")
print(nltk.pos_tag(text))
```

```
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'), ('completely', 'RB'), ('different', 'JJ')]

Process finished with exit code 0
```

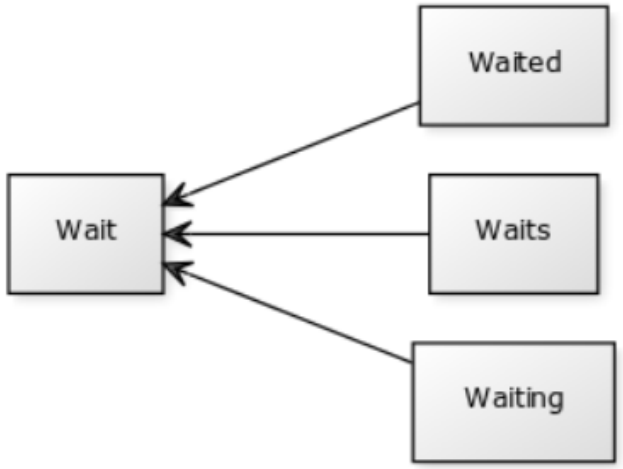
https://www.nltk.org/book/ch05.html

UMKC

Stemming

- For grammatical reasons, documents are going to use different forms of a word, such as organize, organizes, and organizing
- Stemming is the process for reducing injected words to their stem, base root form

- ies -> Y Applies -> Apply
- Ing-> reading -> read



https://pythonspot.com/nltk-stemming/

UMKC

Stemming: Example

```
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
from nltk.stem import SnowballStemmer

pStemmer = PorterStemmer()
print(pStemmer.stem('waiting'))

lStemmer = LancasterStemmer()
print(lStemmer.stem('waiting'))

sStemmer = SnowballStemmer('english')
print(sStemmer.stem('waiting'))
```

```
wait
wait
wait

Process finished with exit code 0
```

Input **Output**

http://text-processing.com/demo/stem/

UMKC

Lemmatization

- Lemmatization process involves first determining the part of speech of a word and applying different normalization rules for each part of speech
- The WordNet Lemmatizer uses the WordNet Database to lookup lemmas
- Lemmatization and stemming are related, but different. The difference is that a stemmer operates on a single word *without* knowledge of the context, and therefore cannot discriminate between words which have different meaning depending on part of speech.

https://pythonprogramming.net/lemmatizing-nltk-tutorial/

UMKC

Lemmatization: Example

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize("cats"))
print(lemmatizer.lemmatize("cacti"))
print(lemmatizer.lemmatize("geese"))
print(lemmatizer.lemmatize("rocks"))
print(lemmatizer.lemmatize("python"))
print(lemmatizer.lemmatize("better", pos="a"))
print(lemmatizer.lemmatize("best", pos="a"))
print(lemmatizer.lemmatize("run"))
print(lemmatizer.lemmatize("run", 'v'))
```

Input

```
cat
cactus
goose
rock
python
good
best
run
run

Process finished with exit code 0
```

Output

https://pythonprogramming.net/lemmatizing-nltk-tutorial/

UMKC

Stemming vs Lemmatization

```
from nltk.stem import WordNetLemmatizer
from nltk.stem import LancasterStemmer

lemmatizer = WordNetLemmatizer()
stemmer = LancasterStemmer()

print(stemmer.stem('geese'))
print(lemmatizer.lemmatize("geese"))
print('\n')
print(stemmer.stem('better'))
print(lemmatizer.lemmatize("better"))
```

```
gees
goose


bet
better

Process finished with exit code 0
```

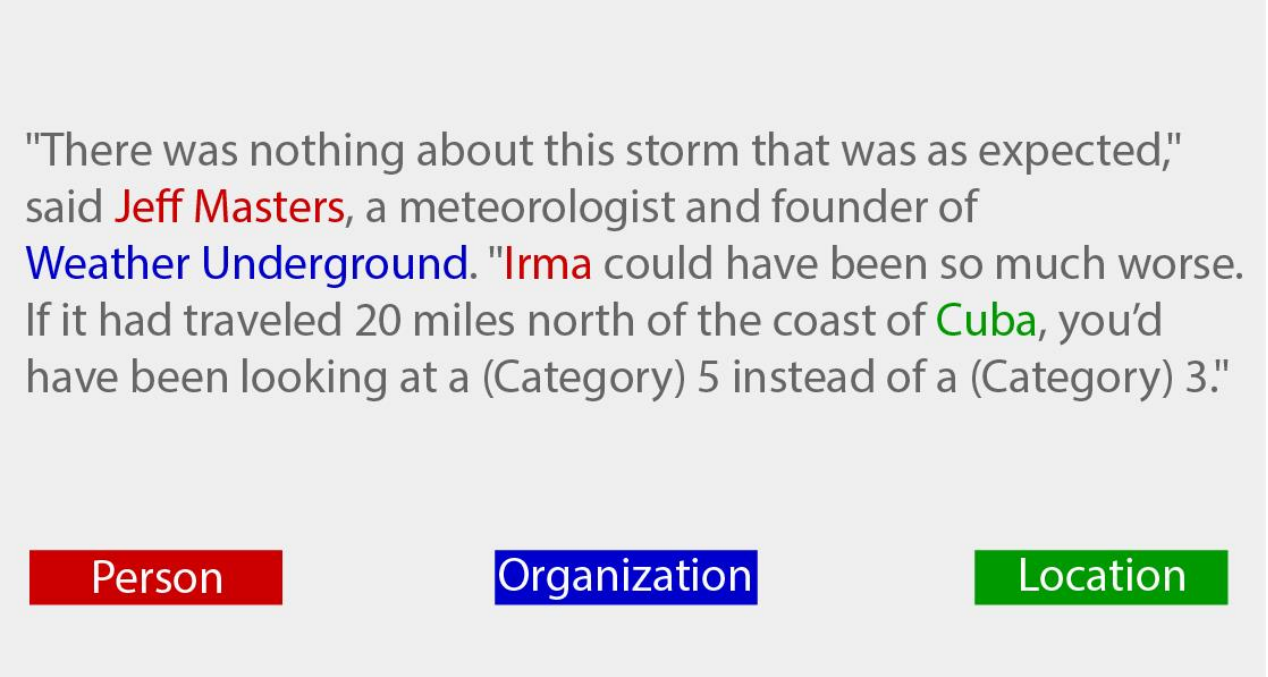
Input **Output**

https://pythonprogramming.net/lemmatizing-nltk-tutorial/

UMKC

Named Entity Recognition (NER)

Named entity recognition is the subtask of information extraction that seeks to locate and classify elements in text into pre-defined categories such as the names of the person, organizations, locations, expressions of times, quantities, etc.



https://www.nltk.org/book/ch07.html

UMKC

Named Entity Recognition (NER): Example

```
from nltk import wordpunct_tokenize, pos_tag, ne_chunk

sentence = "Mark and John are working at Google."

print(ne_chunk(pos_tag(wordpunct_tokenize(sentence))))
```

```
(S
  (PERSON Mark/NNP)
  and/CC
  (PERSON John/NNP)
  are/VBP
  working/VBG
  at/IN
  (ORGANIZATION Google/NNP)
  ./.)

Process finished with exit code 0
```

Input

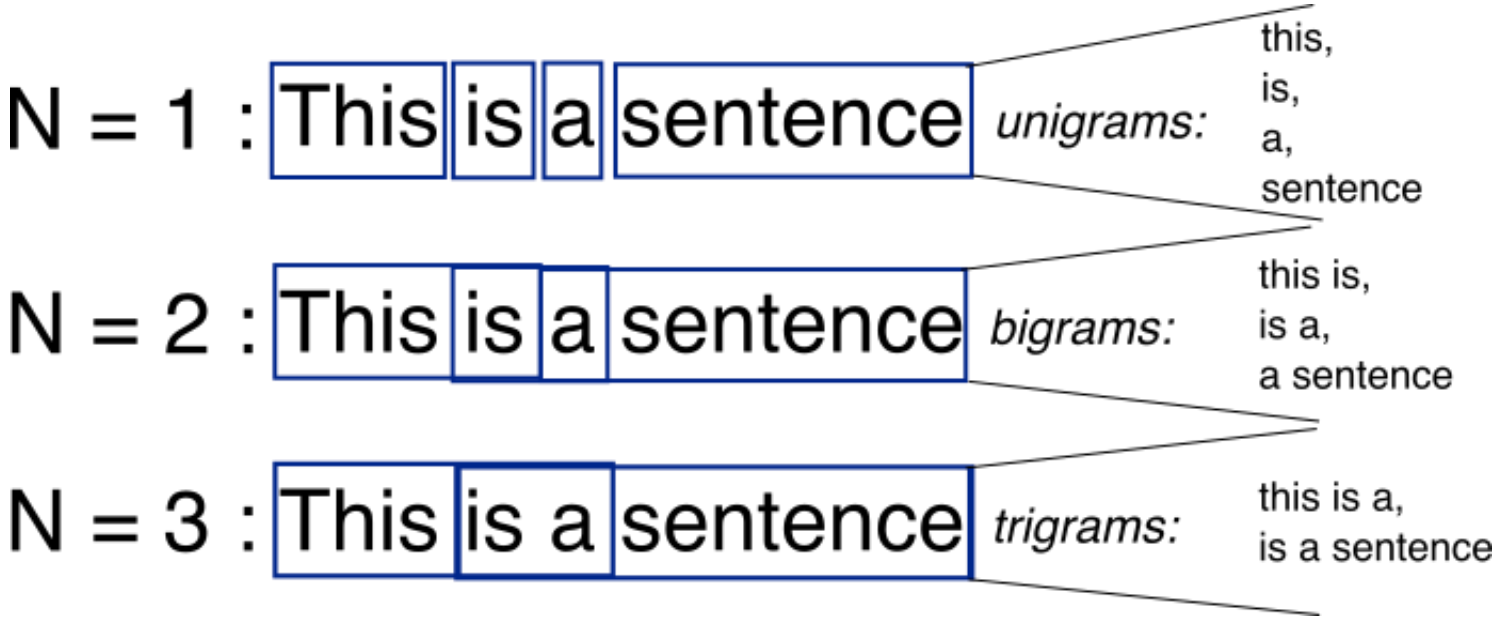
Output

https://www.nltk.org/book/ch07.html

UMKC

N-gram

- An n-gram is a contiguous sequence of n items from a given sample of text or speech.



Trigram: Example

The sentence "the quick red fox jumps over the lazy brown dog" has the following word level trigrams:

- the quick red
- quick red fox
- red fox jumps
- fox jumps over
- jumps over the
- over the lazy
- the lazy brown
- lazy brown dog

UMKC

NLTK modules and functionality

NLTK modules	Functionality
nltk.corpus	Corpus
nltk.tokenize, nltk.stem	Tokenizers, Stemmer
nltk.collacations	T-test, chi-squared, mutual-info
nltk.tag	N-gram
nltk.cluster, nltk.classify	Decision tree, Naïve Bayes, K means
nltk.chunk	Regex, n-gram, named-entity
nltk.parsing	parsing
nltk.metrics	Evaluation metrics
nltk.probability	Probability and estimation
nltk.app, nltk.chat	Applications

UMKC

zip() in Python

The purpose of zip() is to map the similar index of multiple containers so that they can be used just as single entity

Syntax :

*zip(*iterators)*

Parameters :

Python iterables or containers (list, string etc)

Return Value :

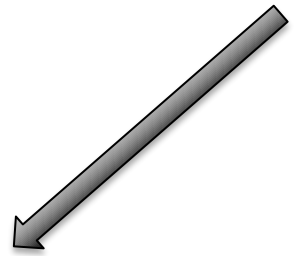
Returns a single iterator object, having mapped values from all the containers.

zip(): Example

```
# Python code to demonstrate the working of
# zip()

# initializing lists
name = ["Student1", "Student2", "Student3", "Student4"]
roll_no = [4, 1, 3, 2]
marks = [40, 50, 60, 70]

# using zip() to map values
mapped = zip(name, roll_no, marks)

# converting values to print as set
mapped = set(mapped)

# printing resultant values
print("The zipped result is : ", end="")
print(mapped)
```

Input

Output



```
The zipped result is : {('Student2', 1, 50), ('Student3', 3, 60), ('Student4', 2, 70), ('Student1', 4, 40)}

Process finished with exit code 0
```

UMKC

We are here to think about

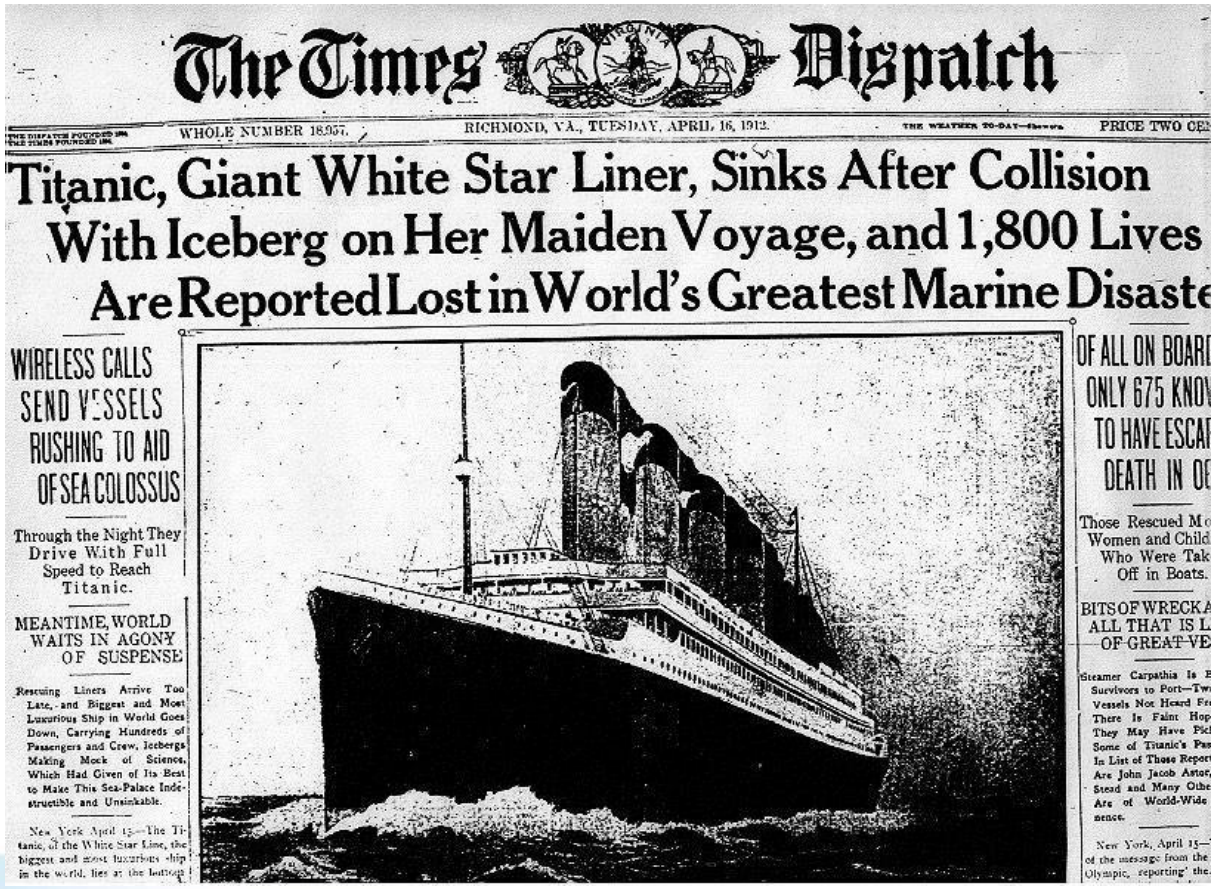
- Determining the topic of an article or a book
- Deciding if an email is spam or not
- Determining who wrote a text
- Determining the meaning of a word in a particular context
- Creating a new poem out of a text

Text Classification

- Text classification is the process of assigning tags or categories to text according to its content
- Unstructured data in the form of text is everywhere: emails, chats, web pages, social media, support tickets, survey responses, and more.
- Text can be an extremely rich source of information, but extracting insights from it can be hard and time-consuming due to its unstructured nature.

Example: News article classification

What is the category of this news article?



The Times Dispatch

WHOLE NUMBER 18,957. RICHMOND, VA., TUESDAY, APRIL 16, 1912. PRICE TWO CE

Titanic, Giant White Star Liner, Sinks After Collision With Iceberg on Her Maiden Voyage, and 1,800 Lives Are Reported Lost in World's Greatest Marine Disaste

WIRELESS CALLS SEND VESSELS RUSHING TO AID OF SEA COLOSSUS

Through the Night They Drive With Full Speed to Reach Titanic.

MEANTIME, WORLD WAITS IN AGONY OF SUSPENSE

Rescuing Liners Arrive Too Late, and Biggest and Most Luxurious Ship in World Goes Down, Carrying Hundreds of Passengers and Crew, Icebergs Making Mock of Science, Which Had Given of Its Best to Make This Sea-Palace Indestructible and Unsinkable.

New York April 15—The Titanic, of the White Star Line, the biggest and most luxurious ship in the world, lies at the bottom

OF ALL ON BOARD ONLY 675 KNOW TO HAVE ESCAP DEATH IN OC

Those Rescued Mo Women and Child Who Were Tak Off in Boats.

BITS OF WRECKA ALL THAT IS L OF GREAT VE

Steamer Carpathia Is E Survivors to Port—Tw Vessels Not Heard Fr There Is Faint Hop They May Have Pic Some of Titanic's Pas In List of Those Report Are John Jacob Astor, Stead and Many Othe Are of World-Wide nence.

New York, April 15— of the message from the Olympic, reporting the

Example: Every word is a feature

Feature vector

Feature space

Feature dimensions	Document 1: Class A	Document 2: Class A	Document 3: Class B	Document 4: Class B
1: arrived	0	1	4	5
2: received	1	2	3	5
3: gold	4	4	4	1
4: a	1	0	1	2
5: energy	5	5	5	3

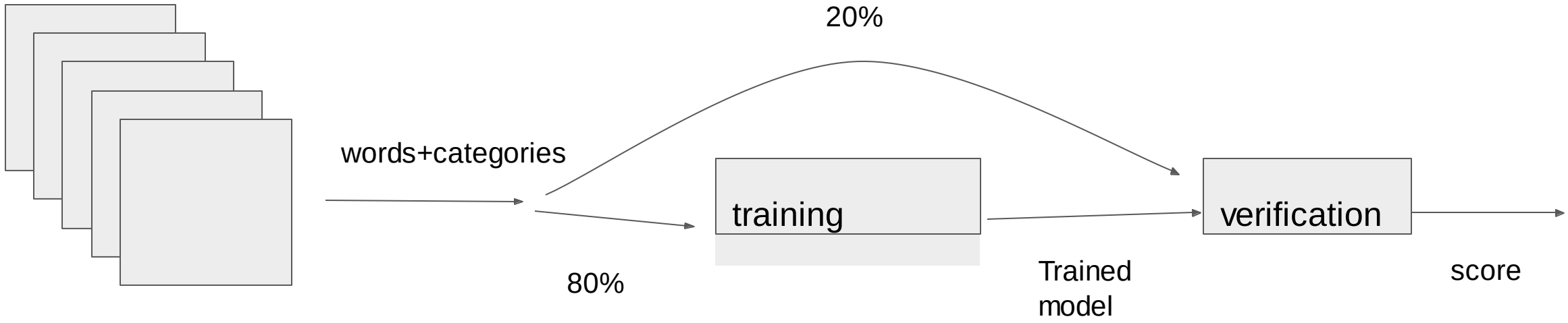
Text = sparse

Feature dimensions	Document 1: Class A	Document 2: Class A	Document 3: Class A	Document 4: Class A	Document 5: Class ?
1: acquired	0	0	1	0	0
2: received	0	2	0	0	0
3: collected	1	0	0	0	0
4: a	0	0	0	2	0
5: energy	0	0	0	0	1

UMKC

Classification score - pipeline

Documents = text + category



Use Case: Text Classification on 20 newsgroup

alt.atheism	4/20/1997 12:08 AM	File folder
comp.graphics	4/20/1997 12:07 AM	File folder
comp.os.ms-windows.misc	4/20/1997 12:09 AM	File folder
comp.sys.ibm.pc.hardware	4/20/1997 12:07 AM	File folder
comp.sys.mac.hardware	4/20/1997 12:07 AM	File folder
comp.windows.x	4/20/1997 12:09 AM	File folder
misc.forsale	4/20/1997 12:08 AM	File folder
rec.autos	4/20/1997 12:08 AM	File folder
rec.motorcycles	4/20/1997 12:08 AM	File folder
rec.sport.baseball	4/20/1997 12:08 AM	File folder
rec.sport.hockey	4/20/1997 12:08 AM	File folder
sci.crypt	4/20/1997 12:09 AM	File folder
sci.electronics	4/20/1997 12:08 AM	File folder
sci.med	4/20/1997 12:08 AM	File folder
sci.space	4/20/1997 12:07 AM	File folder
soc.religion.christian	4/20/1997 12:09 AM	File folder
talk.politics.guns	4/20/1997 12:09 AM	File folder
talk.politics.mideast	4/20/1997 12:09 AM	File folder
talk.politics.misc	4/20/1997 12:07 AM	File folder
talk.religion.misc	4/20/1997 12:09 AM	File folder

PycharmProjects › mini_newsgroups › comp.sys.ibm.pc.hardware

Name	Date modified	Type
58829	4/20/1997 12:07 AM	File
58831	4/20/1997 12:07 AM	File
58922	4/20/1997 12:07 AM	File
58966	4/20/1997 12:07 AM	File
58983	4/20/1997 12:07 AM	File
58994	4/20/1997 12:07 AM	File
60134	4/20/1997 12:07 AM	File
60137	4/20/1997 12:07 AM	File
60150	4/20/1997 12:07 AM	File
60151	4/20/1997 12:07 AM	File
60154	4/20/1997 12:07 AM	File
60156	4/20/1997 12:07 AM	File
60159	4/20/1997 12:07 AM	File
60191	4/20/1997 12:07 AM	File
60199	4/20/1997 12:07 AM	File
60221	4/20/1997 12:07 AM	File
60232	4/20/1997 12:07 AM	File
60235	4/20/1997 12:07 AM	File
60271	4/20/1997 12:07 AM	File
60273	4/20/1997 12:07 AM	File

UMKC

Step1:Dataset preparation

- Reading the data
- twenty_train = fetch_20newsgroups(subset='train', shuffle=True)

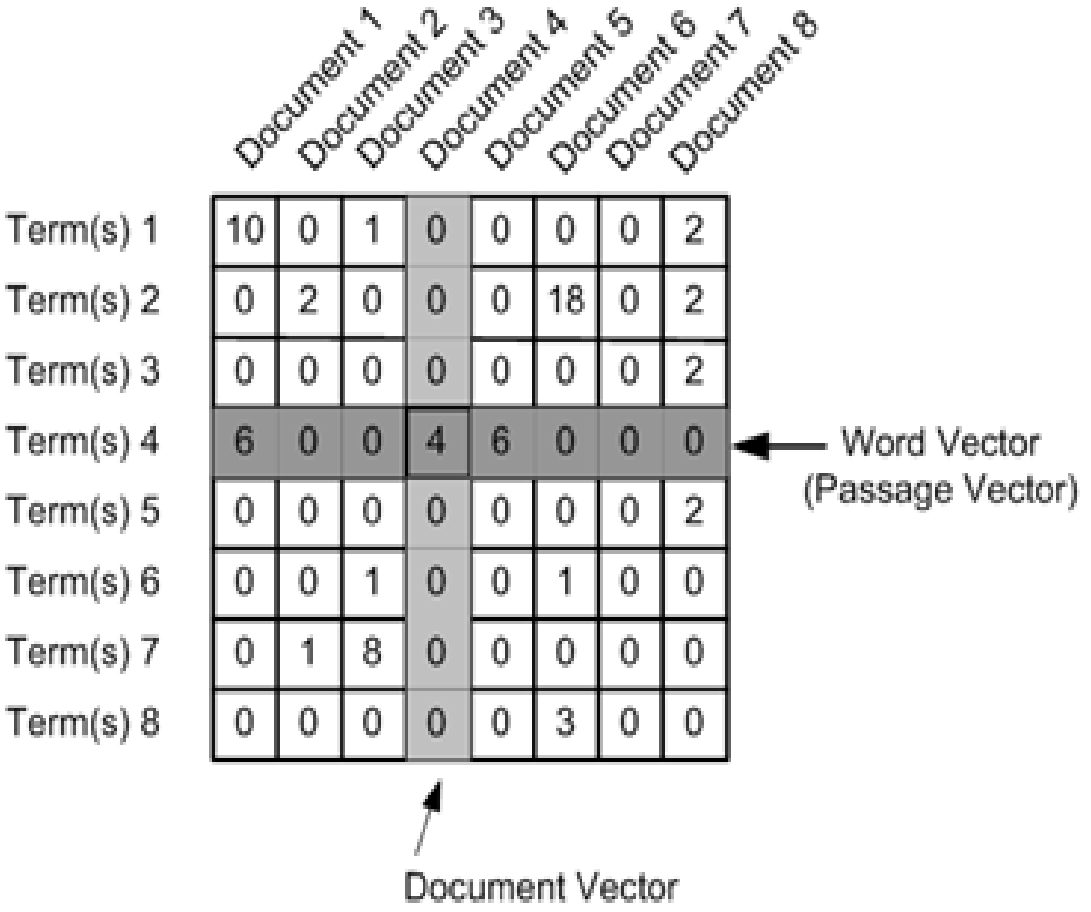
2. Feature Engineering

- The next step is the feature engineering step.
- In this step, raw text data will be transformed into feature vectors
- new features will be created using the existing dataset.

2.1 Count Vectors as features

- Count Vector is a matrix notation of the dataset in which every row represents a document from the corpus,
- every column represents a term from the corpus, and
- every cell represents the frequency count of a particular term in a particular document.

```
count_vect = CountVectorizer()
X_train_counts = count_vect.fit_transform(twenty_train.data)
```



2.2 TF-IDF Vectors as features

- TF-IDF score represents the relative importance of a term in the document and the entire corpus. TF-IDF score is composed by two terms:

- TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)
  IDF(t) = log_e(Total number of documents / Number of documents with term t in it)

```
tfidf_Vect = TfidfVectorizer()
X_train_tfidf = tfidf_Vect.fit_transform(twenty_train.data)
```

3. Model Building

- The final step in the text classification framework is to train a classifier using the features created in the previous step.

```
clf = MultinomialNB()
clf.fit(X_train_tfidf, twenty_train.target)
```

Evaluation

```
twenty_test = fetch_20newsgroups(subset='test', shuffle=True)
X_test_tfidf = tfidf_Vect.transform(twenty_test.data)

predicted = clf.predict(X_test_tfidf)

score = metrics.accuracy_score(twenty_test.target, predicted)
print(score)
```

References

- https://github.com/wade12/WikiScraper/blob/master/
- http://www.w3resource.com/python-exercises/
- https://www.learnpython.org/