NLP Preprocessing Techniques

SCHOOL OF INFOCOMM

Acknowledgement & Thanks

Materials and exercises for this lesson are adapted from Intel AI Developer Program.

https://software.intel.com/en-us/ai

Text is Unstructured !

United Airlines shares plummet after passenger dragged from plane

Shares plummeted Tuesday, wiping close to $1bn off the holding company's value, after a man was violently removed from a flight by aviation police

Shares in United Airlines' parent company plummeted on Tuesday, wiping close to $1bn off of the company's value, a day after a viral video showing police forcibly dragging a passenger off one of its plane became a global news sensation.

The value of the carrier's holding company, United Continental Holdings, had fallen over 4% before noon, close to $1bn less than the $22.5bn as of Monday's close, according to FactSet data.

AUSTEN'S NOVELS EMMA

LONDON: PBINTED BY SrOTTISWOODE AND CO., NEW-STItEEI SQUASH AND PAflLIAJONT STRELT

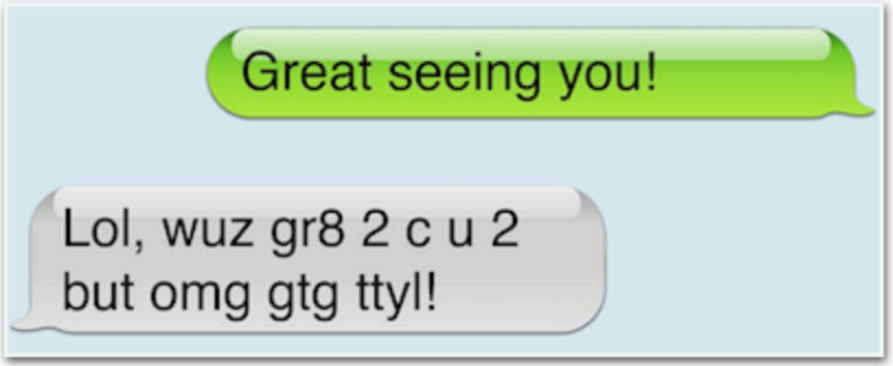
CHAPTER I.

MMA AVOODHOUSE, handsome, clever, and rich, with a comfortable home and happy dis position, seemed to unite some of the best blessings of existence; and had lived nearly twenty-one years in the world with very little to distress or vex her. She was the youngest of the two daughters of a most affec tionate, indulgent father; and had, in consequence of her sister's marriage, been mistress of his house from a very early period. Her mother had died too long ago for her to have more than an indistinct remembrance of her caresses, and her place had been supplied by an excellent woman as gover ness, who had fallen little short of a mother in affection. Sixteen years had Miss Taylor been in Mr. AVoodhouse's family, less as a governess than a friend, very fond of both daughters.

Ai woz lyin on teh stairz n @vorvolak steppd on meh! She sez she noes seez meh buh ai woz dere! 😱😖

Translated from Indonesian by bing Wrong translation?

Could not translate Tweet



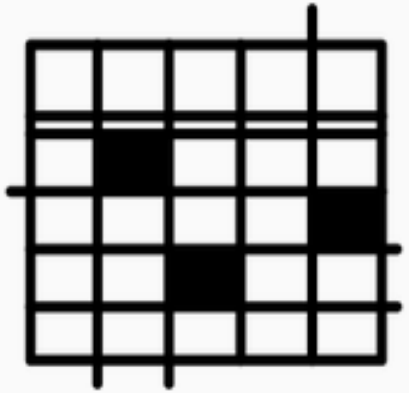
Typical Text Preprocessing Activities

Goal: Converting unstructured text to a meaningful format for analysis

Corpus of raw documents

Tokenization

- Case conversion
- Remove punctuations
- Normalize text
- Remove stop words
- Extract compound terms
- Strip special characters and noises



Data structure (features) representing parsed text

NLP Toolkits

- NLTK (Natural Language Toolkit)

  Good for getting started with NLP.

  Useful for text processing

- TextBlob

  Wraps around NLTK with simple API interfaces

- spaCy

  Built on Cython, fast and powerful

  Text analytics

  Recommend for building real products

- Scikit-learn

  Contains many machine learning algorithm

- Gensim

  Focus for topic modeling and document similarity

Code: How to Install NLTK

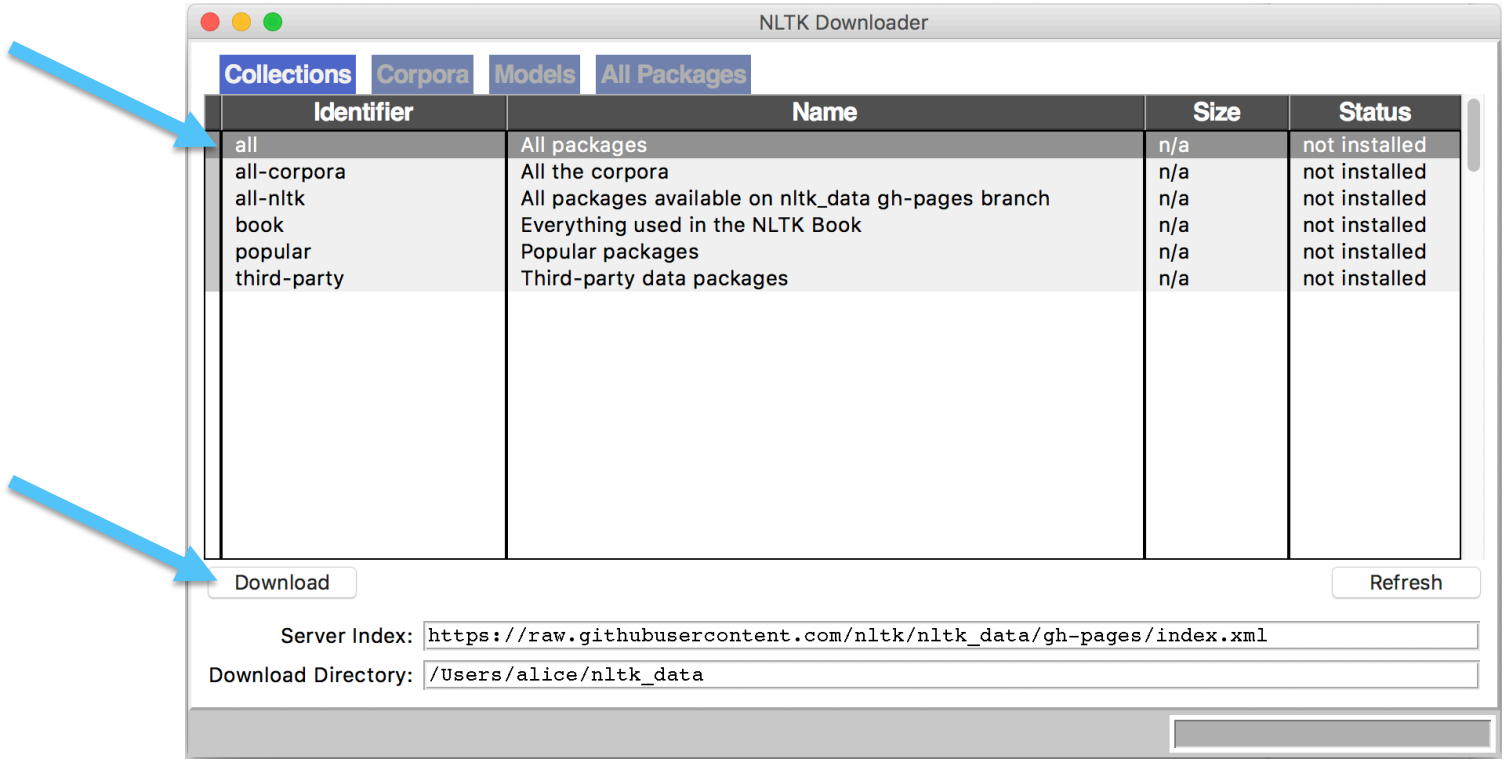
Command Line

pip install nltk

Jupyter Notebook

import nltk
nltk.download()

downloads all data & models
this will take a while



List of corpus and model: http://www.nltk.org/nltk_data

Exercise: What are some things you should do to prepare for text analysis?

Hi Mr. Smith! I'm going to buy some vegetables (tomatoes and cucumbers) from the store. Should i pick up some black-eyed peas as well?

Preprocessing Techniques

1. Turn text into a meaningful format for analysis
   - Tokenization

2. Clean the data
   - Remove - capital letters, punctuation, numbers, stop words
   - Normalization - Stemming, Lemmatization
   - Chunking - compound terms, multi-words phrases

Tokenization

Tokenization = splitting raw text into small, indivisible units for processing

These units can be:

- Sentences
- Words
- N-grams

Tokenization: Sentences

Hi Mr. Smith! I'm going to buy some vegetables (tomatoes and cucumbers) from the store. Should I pick up some black-eyed peas as well?

Tokens can be sentences.

- How would you split this into sentences?
- What rules would you put in place?

It's a difficult task. This is where tokenizers in Python can help.

Code: Tokenization (Text → Sentences)

Input:

```
### sentence tokensization
from nltk.tokenize import sent_tokenize

my_text = "Hi Mr. Smith! I'm going to buy some vegetables (tomatoes and
cucumbers) from the store. Should I pick up some black-eyed peas as
well?"

print(sent_tokenize(my_text))
```

Output:

```
['Hi Mr. Smith!',
 'I'm going to buy some vegetables (tomatoes and cucumbers) from the
store.',
 'Should I pick up some black-eyed peas as well?']
```

Code: Tokenization (Text → Words)

Input:

```
my_text = "Hi Mr. Smith! I'm going to buy some vegetables (tomatoes and
cucumbers) from the store. Should I pick up some black-eyed peas as
well?"
sentences = sent_tokenize(my_text)

my_text_tokens = []
for sentence in sentences:
   my_text_tokens.append(word_tokenize(sentence))
print (my_text_tokens)
```

Output:

```
[['Hi', 'Mr.', 'Smith', '!'], ['I', ''', 'm', 'going', 'to', 'buy', 'some',
'vegetables', '(', 'tomatoes', 'and', 'cucumbers', ')', 'from', 'the', 'store',
'.'], ['Should', 'I', 'pick', 'up', 'some', 'black-eyed', 'peas', 'as', 'well',
'?']]
```

Tokenization: Regular Expressions

Let's say you want to tokenize by some other type of grouping or pattern.

Some examples of regular expressions:

- Find white spaces: \s+
- Find words starting with capital letters: [A-Z]['\w]+

Results of word tokenizer. Not very satisfactory!

```
['Hi', 'Mr.', 'Smith', '!', 'I', ''', 'm', 'going', 'to', 'buy', 'some',
'vegetables', '(', 'tomatoes', 'and', 'cucumbers', ')', 'from', 'the', 'store',
'.', 'Should', 'I', 'pick', 'up', 'some', 'black-eyed', 'peas', 'as', 'well',
'?']
```

Code: Tokenization (Regular Expressions)

Input:

```
from nltk.tokenize import RegexpTokenizer
whitespace_tokenizer = RegexpTokenizer("\s+", gaps=True)
print(whitespace_tokenizer.tokenize(my_text))
```

Output:

```
['Hi', 'Mr.', 'Smith!', 'I’m', 'going', 'to', 'buy', 'some',
'vegetables', '(tomatoes', 'and', 'cucumbers)', 'from', 'the',
'store.', 'Should', 'I', 'pick', 'up', 'some', 'black-eyed',
'peas', 'as', 'well?']
```

N-Grams

Open a bank account. The nearest one is beside the river bank.

Tokenisation by a single word will lose the multiple meaning of 'bank’.

2-Grams : Pairs of 2 words - (a bank), (bank account), (river bank)

3-Grams: Pairs of 3 words – (a bank account), (beside the river)

More explanation at https://youtu.be/E_mN90TYnlg

Code: Tokenization (N-Grams)

Input:

```
from nltk.tokenize import word_tokenize
from nltk.util import ngrams
my_text = "European authorities fined Google a record 5.1 billion on
Wednesday for abusing its power in the mobile phone market"
bigram_mytext = list(ngrams(word_tokenize(my_text),2))
print (bigram_mytext)
```

Output:

```
[('European', 'authorities'), ('authorities', 'fined'), ('fined', 'Google'),
('Google', 'a'), ('a', 'record'), ('record', '5.1'), ('5.1', 'billion'),
('billion', 'on'), ('on', 'Wednesday'), ('Wednesday', 'for'), ('for', 'abusing'),
('abusing', 'its'), ('its', 'power'), ('power', 'in'), ('in', 'the'), ('the',
'mobile'), ('mobile', 'phone'), ('phone', 'market')]
```

Tokenization is language dependent

Chinese: 如果您在新加坡只能前往一间夜间娱乐场所，Zouk必然是您的不二之选。

English: If you only have time for one club in Singapore, then it simply has to be Zouk.

Indonesian: Jika Anda hanya memiliki waktu untuk satu klub di Singapura, pergilah ke Zouk.

Source: https://www.aclweb.org/anthology/Y11-1038

Tokenization is domain specific

Mutants in Toll signaling pathway were obtained from Dr. S. Govind: cactE8, cactIIIG, and cactD13 mutations in the cact gene on Chromosome II.

The maximal effect is observed at the IL-10 concentration of 20 U/ml.

Source: https://www.cs.cmu.edu/~ark/EMNLP-2015/proceedings/LOUHI/pdf/LOUHI05.pdf

Preprocessing Checkpoint

What have we done so far?

- Tokenized text by sentence, word, n-grams and using regex

This is only one step. There is a lot more preprocessing that we can do.

Preprocessing Techniques

1. Turn text into a meaningful format for analysis
   - Tokenization

2. Clean the data
   - Remove - capital letters, punctuation, numbers, stop words
   - Normalization - Stemming, Lemmatization
   - Chunking - compound terms, multi-words phrases

Preprocessing: Remove Characters

Hi Mr. Smith! I'm going to buy some vegetables (tomatoes and cucumbers) from the store. Should I pick up 2lbs of black-eyed peas as well?

How can we 'reduce' this text?

- Remove punctuation
- Remove capital letters and make all letters lowercase
- Remove numbers

Code: Remove Punctuation, Replace with white space

Input:

```
import re # Regular expression library
import string
# Replace punctuations with a white space
clean_text = re.sub("[.,\/#!$%\^&\*;:?{}=\-_`~()]", " ",  my_text)
print("my_text :", my_text)
print("clean_text :",clean_text)
```

Output:

```
my_text : Hi Mr. Smith! I’m going to buy some vegetables (tomatoes and
cucumbers) from the store. Should I pick up some black-eyed peas as well?

clean_text : Hi Mr Smith I’m going to buy some vegetables tomatoes and
cucumbers from the store Should I pick up some black eyed peas as well
```

Code: Make All Text Lowercase

Input:

```
clean_text = clean_text.lower()
clean_text
```

Output:

```
'hi mr  smith  i m going to buy some vegetables  tomatoes and cucumbers
from the store  should i pick up 2lbs of black eyed peas as well '
```

Code: Remove Numbers

Input:

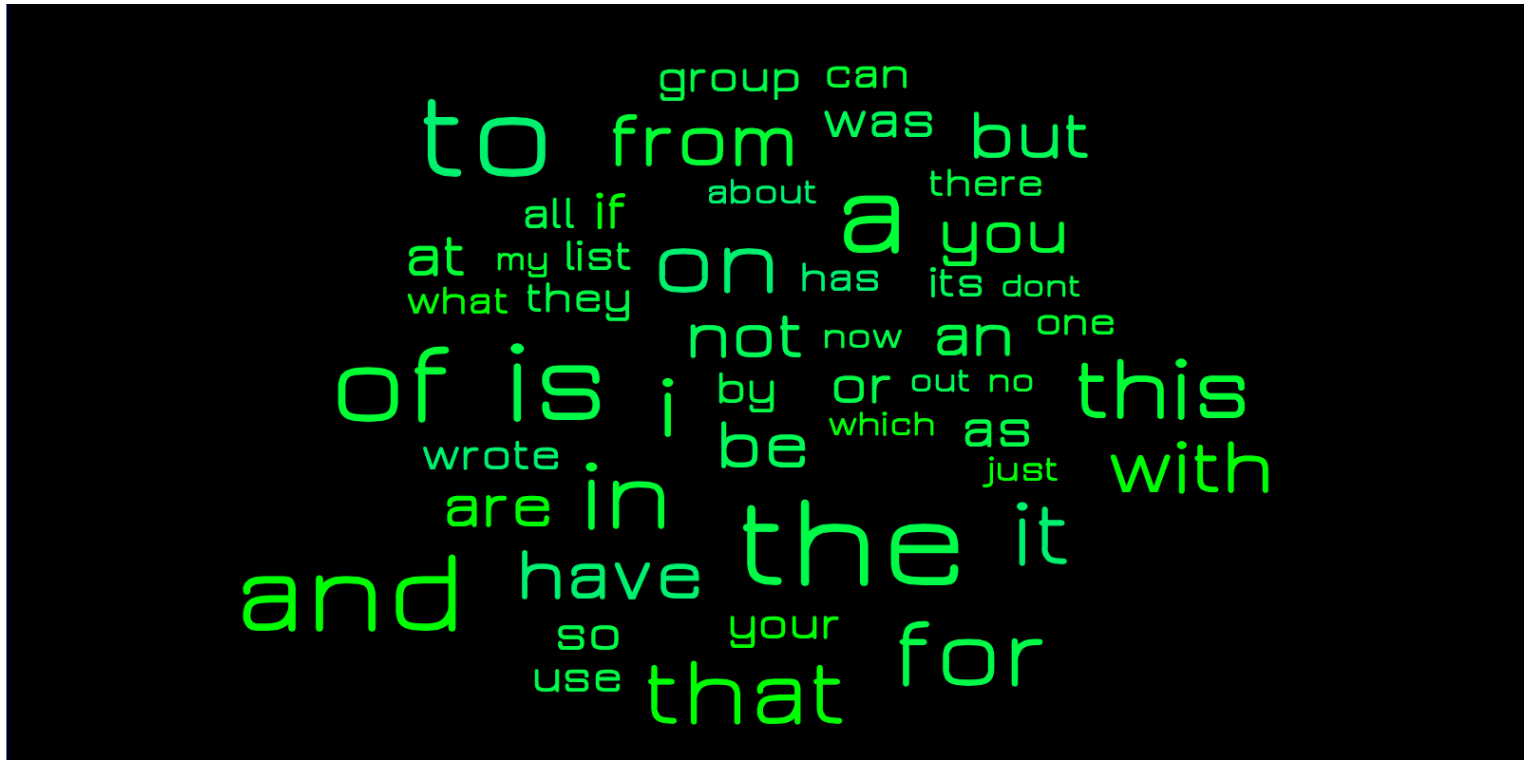
```
# Removes all words containing digits
clean_text = re.sub('\w*\d\w*', ' ', clean_text)
clean_text
```

Output:

```
'hi mr  smith  i m going to buy some vegetables  tomatoes and
cucumbers  from the store  should i pick up   of black eyed peas as
well '
```

Stop Words

Stop words are hi-frequency common words in languages that carry much less substantial information about the meaning of some text



下面	那么
不一	那么些
不久	那么样
不仅	那些
不会	那会儿
不但	那儿
不光	那时
不单	那样
不变	那边
不只	那里
不可	那麽

Example (full text)

The History of Humanity is part of UNESCO’s General and Regional Histories Collection. The publication seeks to contribute to mutual understanding and dialogue between cultures and civilizations. This series seeks to illustrate the encounters between cultures across history and their respective contributions to the general progress of humankind. This is done through the promotion of a pluralist vision of history."

source: https://en.wikipedia.org/wiki/History_of_Humanity

Discussion:

Can you identify some words that are used more frequently than others?

Example: Stop words removed

The History of Humanity is part of UNESCO's General and Regional Histories Collection. The publication seeks to contribute to mutual understanding and dialogue between cultures and civilizations. This series seeks to illustrate the encounters between cultures across history and their respective contributions to the general progress of humankind. This is done through the promotion of a pluralist vision of history."

source: https://en.wikipedia.org/wiki/History_of_Humanity

Discussion:

Will we lose meaning and context if some of the words are removed?

Code: Stop Words

Input: *NLTK provides a list of stop words*

```
from nltk.corpus import stopwords
my_stopwords = set(stopwords.words('english'))
print (my_stopwords)
```

Output:

```
{'above', 'these', 'myself', 'all', 'do', 'shouldn', "aren't", "she's", 'ma', "you're", 'very', 'few', "that'll", 'do
wn', "you'd", 'd', 'his', 'this', 'each', 'wasn', 'doesn', 'me', 'that', 'once', 'before', 'such', 'with', 'aren', "w
eren't", 'haven', 'their', 'between', 'o', 'having', "needn't", 'or', 'own', 'again', 'had', "couldn't", 'the', 'an',
'wouldn', 'as', 'she', 'themselves', 'both', 'can', 'll', 'there', 'when', 'so', "wasn't", "won't", "you'll", 'i', 'b
ut', 'further', 'than', 'yourselves', 'through', 'it', 'you', "it's", 'my', 'he', 'no', 'will', 'needn', 'after', 'am
', 'they', 'how', 'what', "don't", 'some', "should've", 'herself', 'on', 'himself', "hadn't", 'won', 'does', 'be', 'm
ore', 'which', 'until', 'have', 'are', 'ours', 'a', 'in', 'nor', 'who', 'don', 'its', 'doing', 'hasn', "haven't", 'if
', 'to', 'mustn', 'up', 'whom', 'should', 'where', 'now', 've', 'over', 'below', 'under', 'itself', 'out', 'did', 'sa
me', 'hadn', 're', 'him', 'why', 'yours', 't', 'your', 'ourselves', 'her', 'and', 'mightn', 'just', 'by', 'against',
'here', 'our', 'those', 'because', 'yourself', 'were', 'at', 'm', 'we', 'from', 'ain', 'theirs', 'isn', 'for', 'any',
'weren', 'only', "wouldn't", "you've", 'then', 'is', 'shan', 'of', "mightn't", 'hers', "shan't", "doesn't", "didn't",
'into', 'didn', 'being', 's', 'off', "hasn't", 'most', "isn't", 'about', 'other', 'been', 'them', 'y', 'has', 'while
', "mustn't", 'was', "shouldn't", 'not', 'too', 'couldn', 'during'}
```

Code: Stop Words

Input:

You may wish add or remove your own stop words

```
from nltk.corpus import stopwords
my_stopwords = set(stopwords.words('english'))
my_stopwords.add('OMG')
print (my_stopwords)
```

Output:

```
{'above', 'these', 'myself', 'all', 'do', 'shouldn', "aren't", "she's", 'ma', "you're", 'very', 'few', "that'll", 'do
wn', "you'd", 'd', 'his', 'this', 'each', 'wasn', 'doesn', 'me', 'that', 'once', 'before', 'such', 'with', 'aren', "w
eren't", 'haven', 'their', 'between', 'o', 'having', "needn't", 'or', 'own', 'again', 'had', "couldn't", 'the', 'an',
'wouldn', 'as', 'she', 'themselves', 'both', 'can', 'll', 'there', 'when', 'so', "wasn't", "won't", "you'll", 'i', 'b
ut', 'further', 'than', 'yourselves', 'through', 'it', 'you', "it's", 'my', 'he', 'no', 'will', 'needn', 'after', 'am
', 'they', 'how', 'what', "don't", 'some', "should've", 'herself', 'on', 'himself', "hadn't", 'won', 'does', 'be', 'm
ore', 'which', 'until', 'have', 'are', 'ours', 'a', 'in', 'nor', 'who', 'don', 'its', 'doing', 'hasn', "haven't", 'if
', 'to', 'mustn', 'up', 'whom', 'should', 'where', 'now', 've', 'over', 'below', 'under', 'itself', 'out', 'did', 'sa
me', 'hadn', 're', 'him', 'why', 'yours', 't', 'your', 'ourselves', 'her', 'and', 'mightn', 'just', 'by', 'against',
'here', 'our', 'those', 'because', 'yourself', 'were', 'at', 'm', 'we', 'from', 'ain', 'theirs', 'isn', 'for', 'any',
'weren', 'only', "wouldn't", "you've", 'then', 'is', 'shan', 'of', "mightn't", 'hers', "shan't", "doesn't", "didn't",
'into', 'didn', 'being', 's', 'off', "hasn't", 'most', 'OMG', "isn't", 'about', 'other', 'been', 'them', 'y', 'has',
'while', "mustn't", 'was', "shouldn't", 'not', 'too', 'couldn', 'during'}
```

Code: Remove Stop Words from Tokens

Input:

```
example_sent = "This is a sample sentence, showing off the stop words filtration."
stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(example_sent)
filtered_sentence = [w for w in word_tokens if not w in stop_words]

print("Original word tokens: " , word_tokens)
print("Filtered word tokens: " , filtered_sentence)
```

Output:

```
Original word tokens: ['This', 'is', 'a', 'sample', 'sentence', ',', 'showing',
'off', 'the', 'stop', 'words', 'filtration', '.']

Filtered word tokens: ['This', 'sample', 'sentence', ',', 'showing', 'stop',
'words', 'filtration', '.'
```

Danger: removing stop words may affect meaning

Text and labels for sentiment analysis of product reviews – before stopword removal

1. The product is really very good. — POSITIVE

2. The products seems to be good. — POSITIVE

3. Good product. I really liked it. — POSITIVE

4. I didn't like the product. — NEGATIVE

5. The product is not good. — NEGATIVE

Text and labels for sentiment analysis of product reviews – after stopword removal

1. product really good. — POSITIVE

2. products seems good. — POSITIVE

3. Good product. really liked. — POSITIVE

4. like product. — NEGATIVE

5. product good. — NEGATIVE

Example from: https://towardsdatascience.com/why-you-should-avoid-removing-stopwords-aa7a353d2a52

Example:

Input:

```
text = 'the product is not good'
stop_words = set(stopwords.words('english'))
word_tokens = word_tokenize(text)

filtered_sentence = [w for w in word_tokens if not w in stop_words]

print("Original word tokens: " , word_tokens)
print("Filtered word tokens: " , filtered_sentence)
```

Output:

```
Original word tokens: ['the', 'product', 'is', 'not', 'good']
Filtered word tokens: ['product', 'good']
```

Question: How do you put back the 'NOT' ?

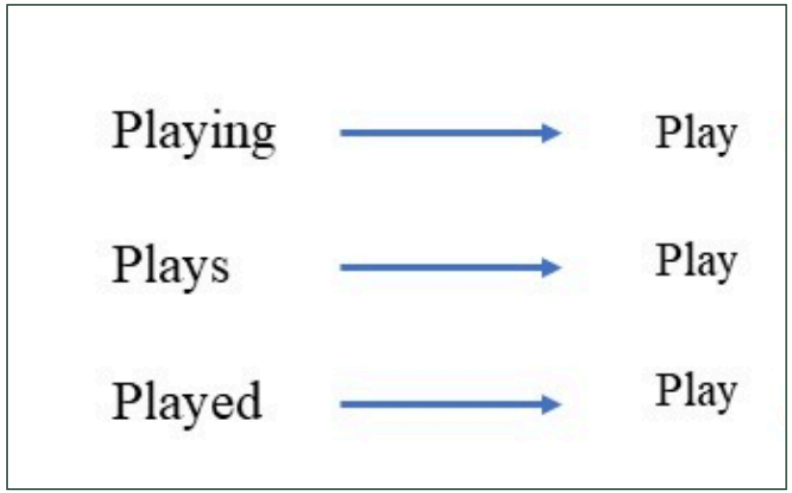
Preprocessing Techniques

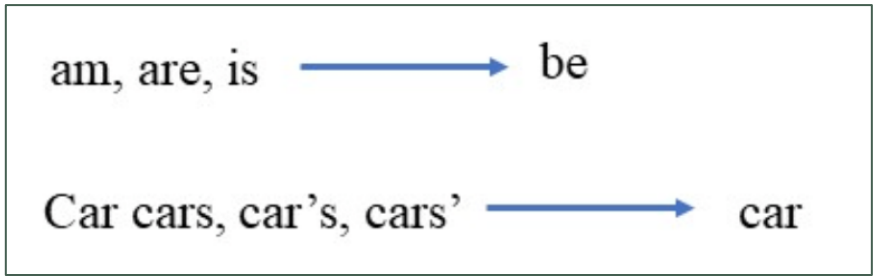
1. Turn text into a meaningful format for analysis
   - Tokenization

2. Clean the data
   - Remove - capital letters, punctuation, numbers, stop words
   - Normalization - Stemming, Lemmatization
   - Chunking - compound terms, multi-words phrases

Normalization – Key Idea

What we speak and write are made up of several words often derived from one another. Some words are derived from root words.





Normalized sentence

If we can cut down the number of unique words, we can reduce the complexity.

Example fom: https://www.datacamp.com/community/tutorials/stemming-lemmatization-python

Stemming

Caring → Car (wrong meaning!)
Cars → Car
Drawer → Draw
Happily → Happi (misspelling!)

Process of reducing inflected or derived words to their word stem by using heuristics rules

Chops without knowledge of context
- Stemmed word may not be a proper word found in the dictionary
- Misspelled or incomplete

Common approach: PorterStemmer, LancasterStemmer, SnowballStemmer

Lemmatization

Careful → Care
Caring → Care
Cars → Car
Car → Car

Process of reducing derived words to their word stem by using vocabulary set

Takes into account a word's meaning, synonyms and **part-of-speech** to ensure that words are corrected reduced to the correct base words

Dictionary: WordNetLemmatizer

Code: Lemmatization (basic)

Input:

```
from nltk import pos_tag
from nltk.stem import  WordNetLemmatizer
lemmatiser = WordNetLemmatizer()

words = ["car", "cars", "care", "caring", "careful", "boats", "boating"]

for word in words:
   print("Lemmatise %s  --> %s" % (word, lemmatiser.lemmatize(word)))
```

Output:

```
Lemmatise car  --> car
Lemmatise cars  --> car
Lemmatise care  --> care
Lemmatise caring  --> caring
Lemmatise careful  --> careful
Lemmatise boats  --> boat
Lemmatise boating  --> boating
```

Code: Lemmatization (with Parts of Speech)

Input:

```
from nltk.stem import  WordNetLemmatizer
lemmatiser = WordNetLemmatizer()

words = ["car", "cars", "care", "caring", "careful", "boats", "boating"]
for word in words:
    print("Lemmatise %s  --> %s" % (word, lemmatiser.lemmatize(word, pos="v")))

for word in words:
     print("Lemmatise %s  --> %s" % (word, lemmatiser.lemmatize(word, pos="n")))
```

Output:

```
Lemmatise car --> car
Lemmatise cars --> cars
Lemmatise care --> care
Lemmatise caring --> care
Lemmatise careful --> careful
Lemmatise boats --> boat
Lemmatise boating --> boat
```

```
Lemmatise car  --> car
Lemmatise cars  --> car
Lemmatise care  --> care
Lemmatise caring  --> caring
Lemmatise careful  --> careful
Lemmatise boats  --> boat
Lemmatise boating  --> boating
```

Code: Lemmatization (with Parts of Speech)

Input:

```
from nltk import pos_tag
from nltk.stem import  WordNetLemmatizer
lemmatiser = WordNetLemmatizer()

words = ["car", "cars", "care", "caring", "careful", "boats", "boating"]

for word in words:
   print("Lemmatise %s  --> %s" % (word, lemmatiser.lemmatize(word, pos="n")))
```

Output:

```
Lemmatise car  --> car
Lemmatise cars  --> car
Lemmatise care  --> care
Lemmatise caring  --> caring
Lemmatise careful  --> careful
Lemmatise boats  --> boat
Lemmatise boating  --> boating
```

***** Treat the word as Noun***

Preprocessing Techniques

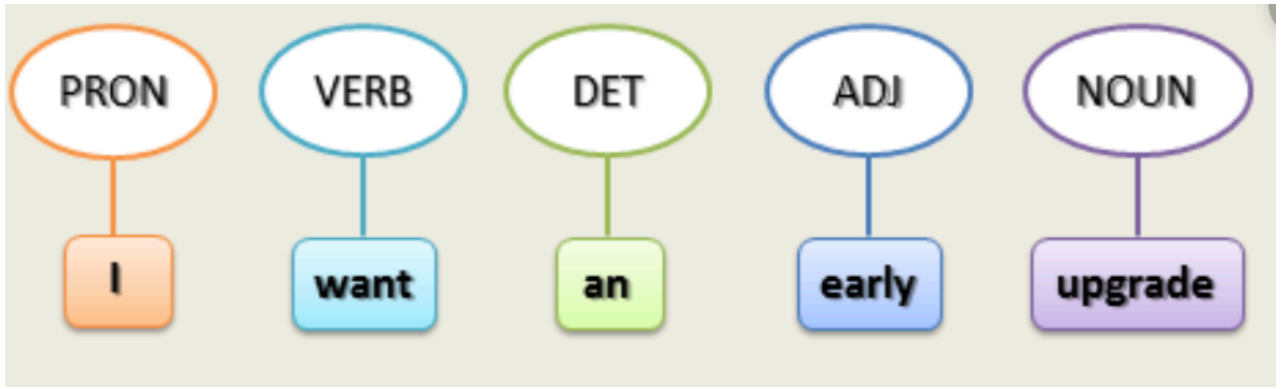
1. Turn text into a meaningful format for analysis
   - Tokenization

2. Clean the data
   - Remove - capital letters, punctuation, numbers, stop words
   - Normalization - stemming, lemmatization
   - Chunking - compound terms, multi-words phrases

Preprocessing: Parts of Speech Tagging

Parts of Speech

- Nouns, verbs, adjectives
- Parts of speech tagging labels (grammar information) for each word



Code: Parts of Speech Tagging

Input:

```
# Part of speech
from nltk.tag import pos_tag
my_text = "James Smith lives in the United States."
tokens = pos_tag(word_tokenize(my_text))
print(tokens)
```

Output:

```
[('James', 'NNP'),
 ('Smith', 'NNP'),
 ('lives', 'VBZ'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('United', 'NNP'),
 ('States', 'NNPS'),
 ('.', '.')]
```

Code: Parts of Speech Tagging

Input:

```
for tag in ['NNP', 'VBZ', 'DT']:
    print(nltk.help.upenn_tagset(tag))
```

```
NNP: noun, proper, singular
    Motown Venneboerger Czestochwa Ranzer Conchita Trumplane Christos
    Oceanside Escobar Kreisler Sawyer Cougar Yvette Ervin ODI Darryl CTCA
    Shannon A.K.C. Meltex Liverpool ...
None
VBZ: verb, present tense, 3rd person singular
    bases reconstructs marks mixes displeases seals carps weaves snatches
    slumps stretches authorizes smolders pictures emerges stockpiles
    seduces fizzes uses bolsters slaps speaks pleads ...
None
DT: determiner
    all an another any both del each either every half la many much nary
    neither no some such that the them these this those
None
```

Preprocessing: Chunking

I am interested in **data science** and **artificial intelligence**

- What is the right way to tokenize?

"data", "science“, "data_science"

"artificial" , "intelligence" ,"artificial intelligence"

Code: Chunking

Input:

```
from nltk.tokenize import MWETokenizer, word_tokenize

mwe_tokenizer = MWETokenizer([("artificial","intelligence"), ("data","science")],
separator='_')

text = "I am interested in data science and artificial intelligence"

mwe_tokens = mwe_tokenizer.tokenize(word_tokenize(text))

print("Compound words tokens -> ", mwe_tokens)
```

Output:

```
Compound words tokens ->  ['I', 'am', 'interested', 'in', 'data_science', 'and', 'artificial_intelligence']
```

Other cleansing issues

Acronym normalization and tagging – acronyms can be specified as “I.B.M.” or “IBM” so these should be tagged and normalized.

Special characters in scraped text from web pages and PDF files

Identify and remove useless sections, such as common headers, footers, and sidebars as well as legal or commercial boilerplates

Knowledge Check

Given the text below, what are some preprocessing techniques you could apply?

> Having purchased both this jacket (India Ink/Boulder) and the Green/Charcoal Heather version, I can tell you that this one is slightly tighter at the butt than the green one, which matches my old jacket. I know this because my butt is slightly bigger than it was five years ago. The green jacket is pretty much exactly the same size all over as my original jacket, but this one is maybe 1/8" tighter at the butt. It's not much of a difference, but does make me question those extra biscuits I've been consuming during this holiday season. I think these new jackets will be seeing more time on the bike this year than my old jacket did last year. I must thank Columbia for their brutal honesty regarding the circumference of my posterior. Whether the lessened draft of this jacket is a one-off deal, or if all of the India Ink/Boulder jackets got shorted is unknown to me. You'll just have to ask yourself, "Does that 1/8" really matter to me, or anything this guy is saying?"

Source: Amazon product review

References

- Natural Language Toolkit, https://www.nltk.org/

- Text Wrangling & Pre-processing: A Practitioner's Guide to NLP, https://www.kdnuggets.com/2018/08/practitioners-guide-processing-understanding-text-2.html