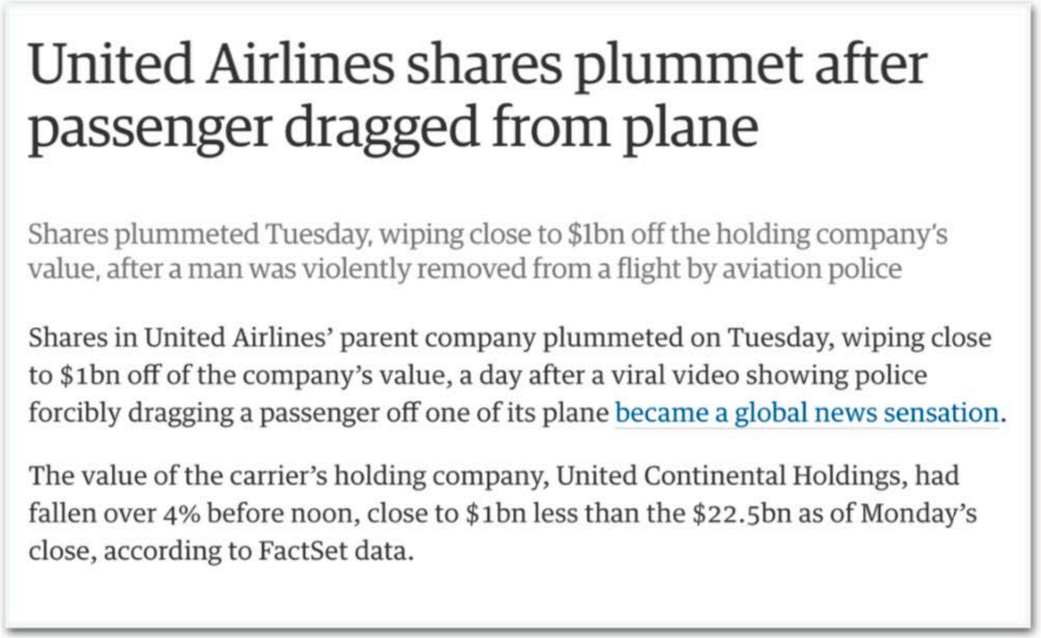
NLP Preprocessing Techniques

SCHOOL OF INFOCOMM

Text is Unstructured and Noisy!

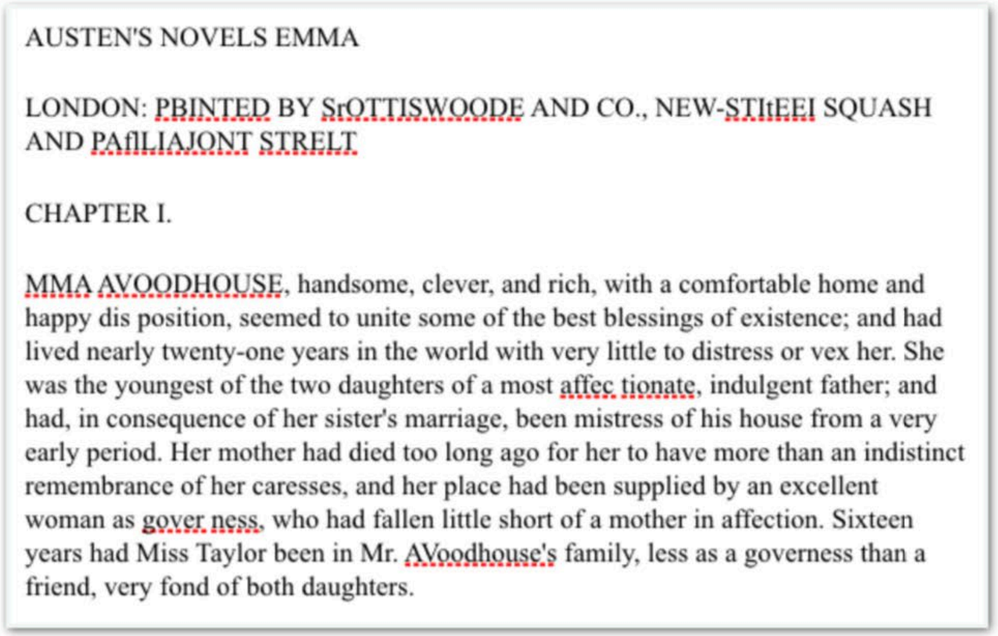


United Airlines shares plummet after passenger dragged from plane

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.

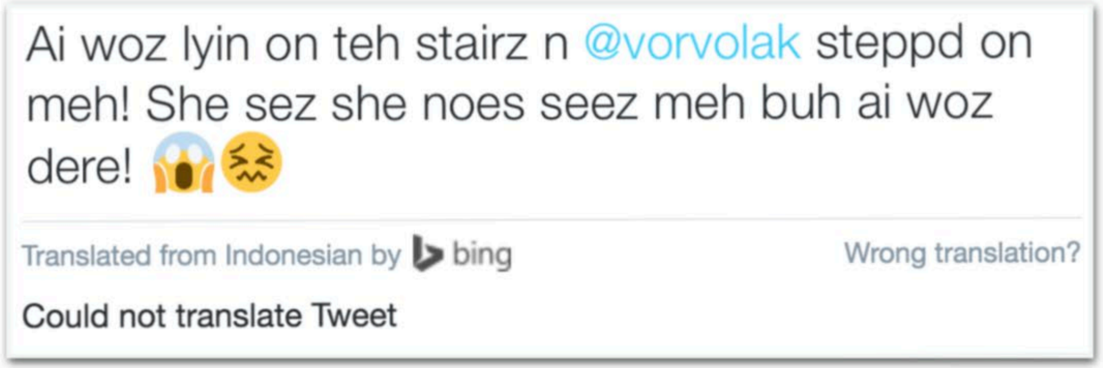


AUSTEN'S NOVELS EMMA

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.

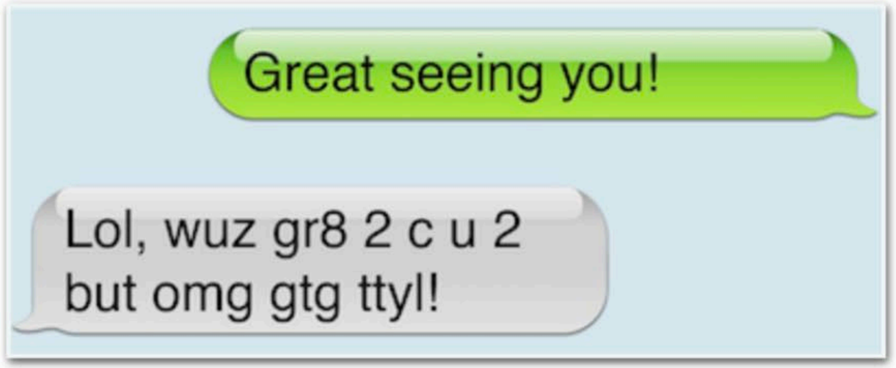


Ai woz lyin on teh stairz n @vorvolak steppd on meh! She sez she noes seez meh buh ai woz dere! 😱😣

Translated from Indonesian by bing

Wrong translation?

Could not translate Tweet



Great seeing you!

Lol, wuz gr8 2 c u 2 but omg gtg ttyl!

Software / Libraries

NLTK

GENSIM
topic modelling for humans

fastText

PyTorch NLP

scikit learn

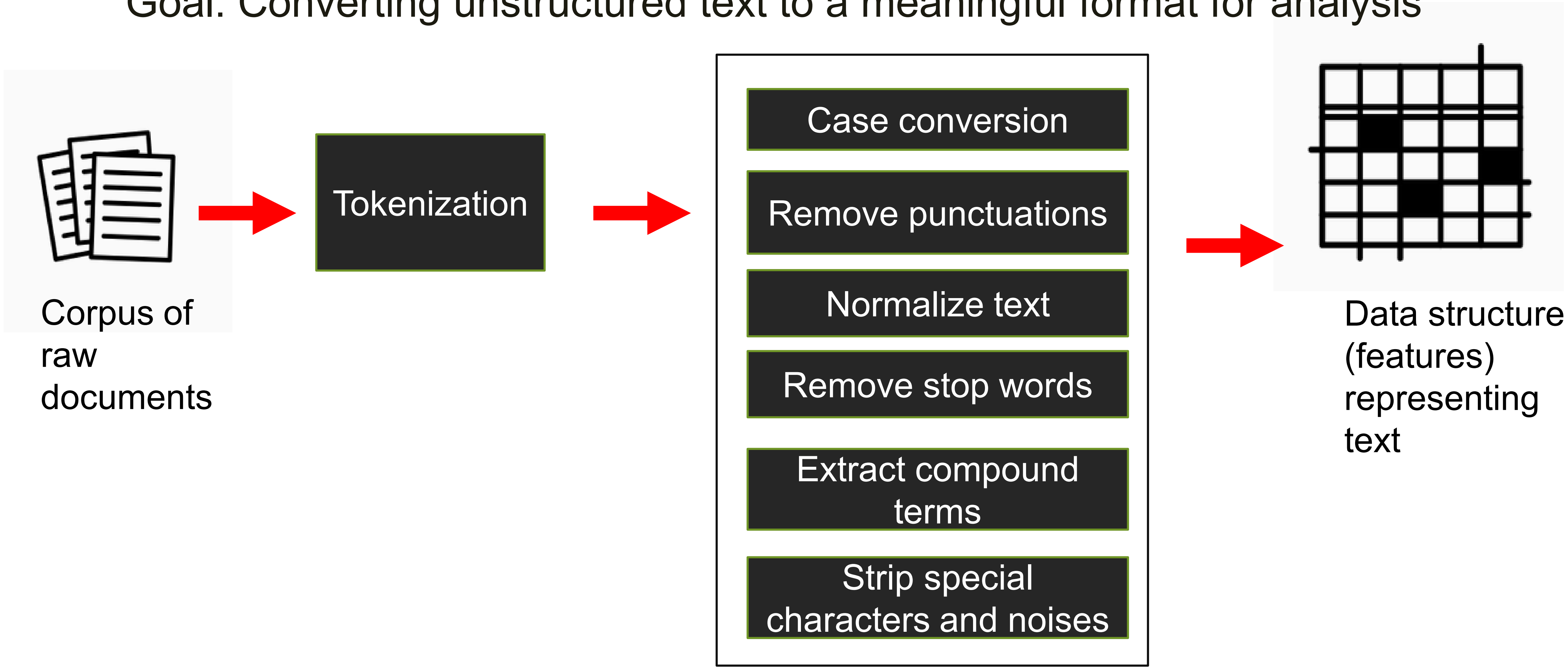
spaCy

OPEN NLP™

See https://medium.com/activewizards-machine-learning-company/comparison-of-top-6-python-nlp-libraries-c4ce160237eb

Text Preprocessing Activities

Goal: Converting unstructured text to a meaningful format for analysis



Introducing NLTK (Natural Language Toolkit)

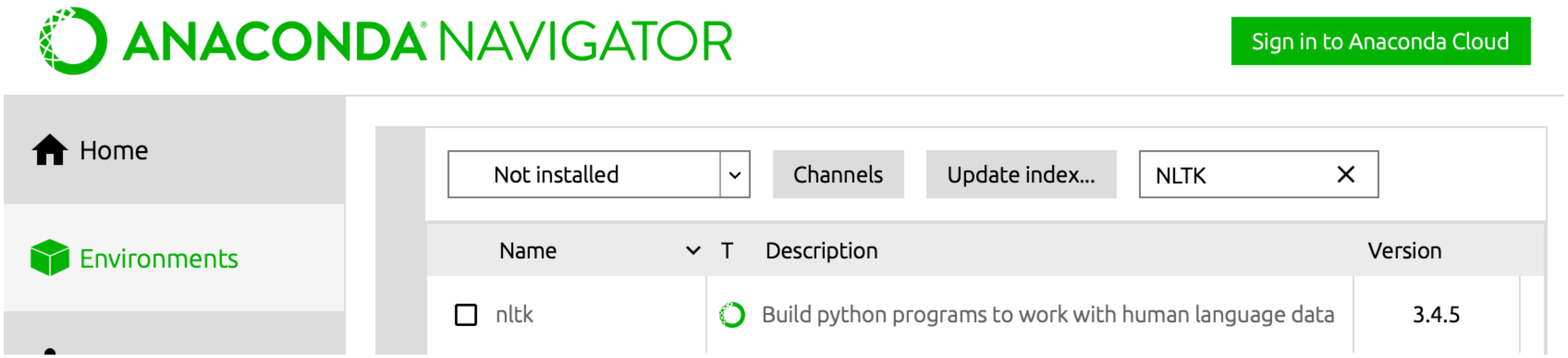
- NLTK is a popular platform for building Python programs to work with human language data

- It provides easy-to-use interfaces to over 50 corpora, lexical resources and trained models.
  http://www.nltk.org/nltk_data/

- Has a suite of text processing libraries for tokenization, stemming, tagging, parsing, and semantic reasoning, classifications wrappers for other NLP libraries

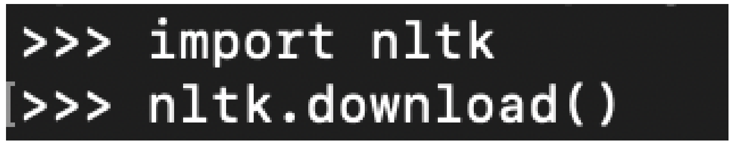
How to Install NLTK

1. Install package from Anaconda

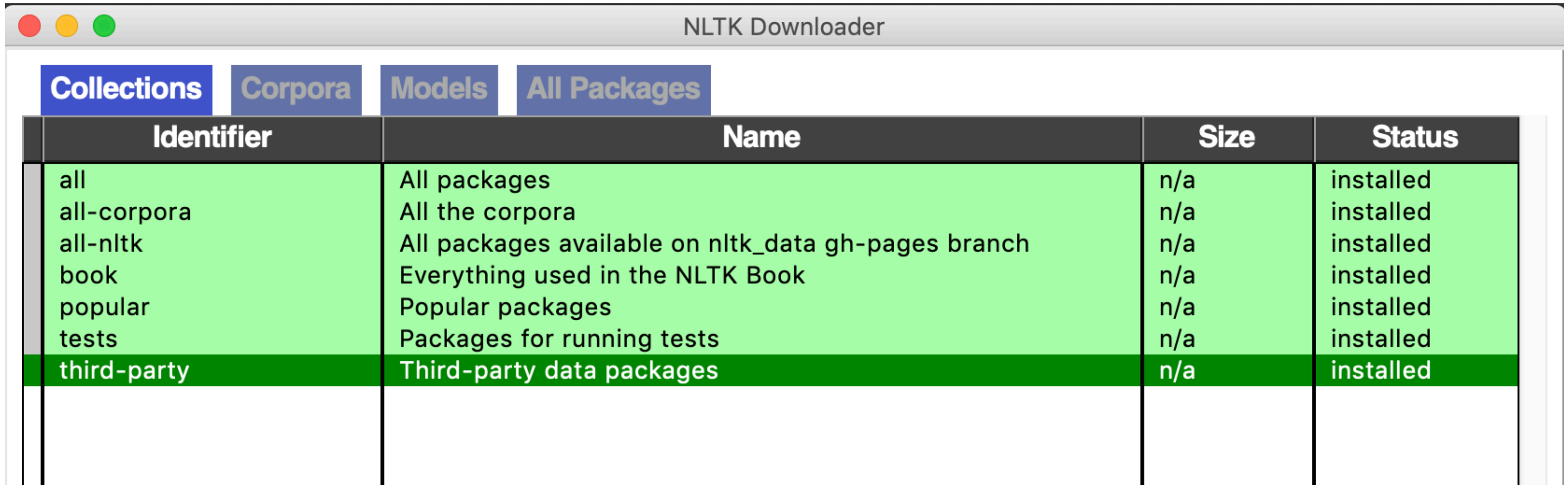


Name	T	Description	Version
☐ nltk		Build python programs to work with human language data	3.4.5

2. Download all data and models from Python shell



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



Identifier	Name	Size	Status
all	All packages	n/a	installed
all-corpora	All the corpora	n/a	installed
all-nltk	All packages available on nltk_data gh-pages branch	n/a	installed
book	Everything used in the NLTK Book	n/a	installed
popular	Popular packages	n/a	installed
tests	Packages for running tests	n/a	installed
third-party	Third-party data packages	n/a	installed

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, 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 / words

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 or words.

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

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

Code: Sentence Tokenization (Text → Sentences)

```python
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?"""
```

```python
sentences = sent_tokenize(my_text)
```

```python
print (sentences)
```

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

Refer to nltk-example-1-tokenization-sentence.ipynb

Code: Word Tokenization (Text → Words)

```python
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?"""
```

```python
words = word_tokenize(my_text)
```

```python
print (words)
```

```
['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', '?']
```

Ref to : nltk-example-2-tokenization-word.ipynb

Code: Word Tokenization
(Text → Sentences → Words)

```python
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?"""
```

```python
sentences = sent_tokenize(my_text)
```

```python
sentences_words = []

for s in sentences:
    sentences_words.append(word_tokenize(s))
```

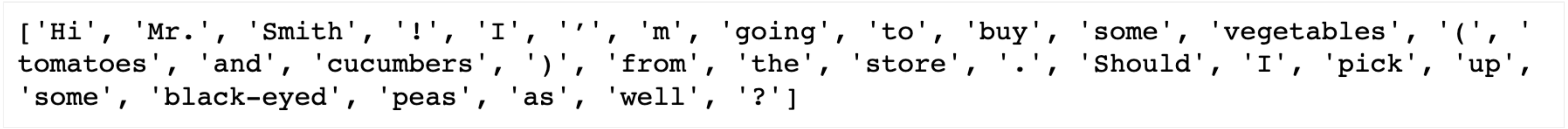
```python
print (sentences_words)
```

```
[['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', '?']]
```

Refer to nltk-example-3-tokenization-sentence_word.ipynb

Tokenization: Regular Expressions

The results of word tokenization is not very satisfying.



```
['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', '?']
```

Let’s say you want to tokenize by some other type of grouping or pattern based on the written characteristic of the language.

For English, some characteristics are:

- White spaces (\s+) between words
- Sentences starts with capital letters: [A-Z]['\w]+
- Sentences ends with punctuation (!?.)

Code: Tokenization (Regular Expressions)

```
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)
```

```
whitespace_tokenizer = RegexpTokenizer("\s+", gaps=True)

sentences_words = []
for s in sentences:
    sentences_words.append(whitespace_tokenizer.tokenize(s))
```

```
print (sentences_words)
```

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

Refer to nltk-example-4-tokenization-regular-expression.ipynb

Problem with word tokenization

Open a **bank** account. The nearest one is beside the river **bank**. On your way there, you may wish to donate some blood at the nearby blood **bank**.

```
['Open', 'a', 'bank', 'account', '.', 'The', 'nearest', 'one', 'is', 'beside', 'the', 'river
', 'bank', '.', 'On', 'your', 'way', 'there', ',', 'you', 'may', 'wish', 'to', 'donate', 'som
e', 'blood', 'at', 'the', 'nearby', 'blood', 'bank', '.']
```

The word bank has different meaning.

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

N-Gram

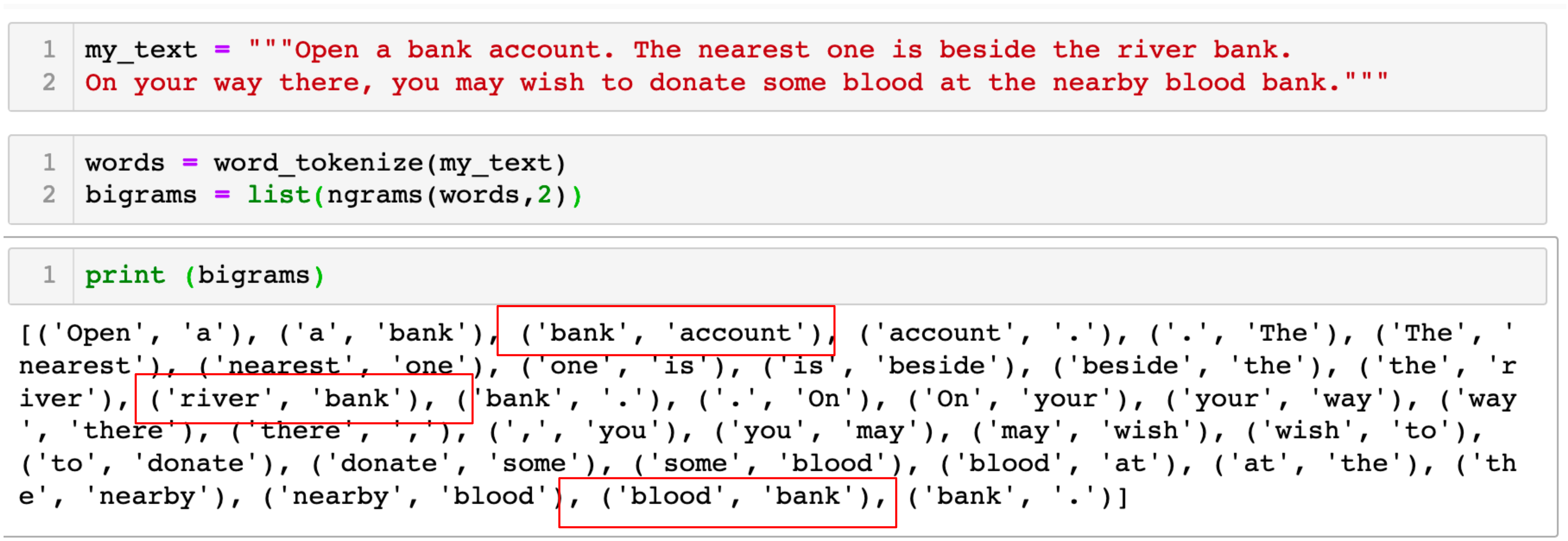
- N-gram is a contiguous sequence of **n items** from a given sample of text.

- The items can be phonemes, syllables, letters or words pairs according to the application.

- Sequence of one item – Unigram
- Sequence of two items – Bigrams
- Sequence of three items – Trigrams

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

Code: Tokenization (N-Grams)



```
my_text = """Open a bank account. The nearest one is beside the river bank.
On your way there, you may wish to donate some blood at the nearby blood bank."""
```

```
words = word_tokenize(my_text)
bigrams = list(ngrams(words,2))
```

```
print (bigrams)
```

```
[('Open', 'a'), ('a', 'bank'), ('bank', 'account'), ('account', '.'), ('.', 'The'), ('The', '
nearest'), ('nearest', 'one'), ('one', 'is'), ('is', 'beside'), ('beside', 'the'), ('the', 'r
iver'), ('river', 'bank'), ('bank', '.'), ('.', 'On'), ('On', 'your'), ('your', 'way'), ('way
', 'there'), ('there', ','), (',', 'you'), ('you', 'may'), ('may', 'wish'), ('wish', 'to'),
('to', 'donate'), ('donate', 'some'), ('some', 'blood'), ('blood', 'at'), ('at', 'the'), ('th
e', 'nearby'), ('nearby', 'blood'), ('blood', 'bank'), ('bank', '.')]
```

Tokenization by bi-grams retains some meaning the word "bank"
(blank, account), (river bank), (blood bank)

Ref er to nltk-example-5-tokenization-ngram.ipynb

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.

1. Example from tSource: https://www.aclweb.org/anthology/Y11-1038
2. Tutorial for Chinese words tokenization can be found at https://michelleful.github.io/code-blog/2015/09/10/parsing-chinese-with-stanford/

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

Exercise

Tokenization

Refer to nltk-exercise-1-tokenization.ipynb

Preprocessing Checkpoint

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

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

Preprocessing: Remove Punctuation and Upper case

Open a bank account. The nearest DBS Bank is besides the river bank. On your way there, you may wish to donate some blood at the nearby blood bank. Will you be willing to donate? It is open from 9:00 a.m to 4:00 p.m.

```
['Open', 'a', 'bank', 'account', '.', 'The', 'nearest', 'DBS', 'Bank', 'is', 'besides', 'the
', 'river', 'bank', '.', 'On', 'your', 'way', 'there', ',', 'you', 'may', 'wish', 'to', 'dona
te', 'some', 'blood', 'at', 'the', 'nearby', 'blood', 'bank', '.', 'Will', 'you', 'be', 'will
ing', 'to', 'donate', '?', 'It', 'is', 'open', 'from', '9:00', 'a.m', 'to', '4:00', 'p.m',
'.']
```

The end of sentence punctuations (e.g. full stop , question mark) may be irrelevant. The tokens (Open, open) and (The, the) are the same word, same meaning.

Code: Remove Punctuation and Change to Lower Case

```python
my_text = """Open a bank account. The nearest DBS Bank is besides
the river bank. On your way there, you may wish to donate some blood
at the nearby blood bank. Will you be willing to donate?
It is open from 9:00 a.m to 4:00 p.m."""
```

```python
tokens = word_tokenize(my_text)
```

```python
tokens_no_punctuation = []
for t in tokens:
    if not(t in string.punctuation):
        tokens_no_punctuation.append(t.lower())
```

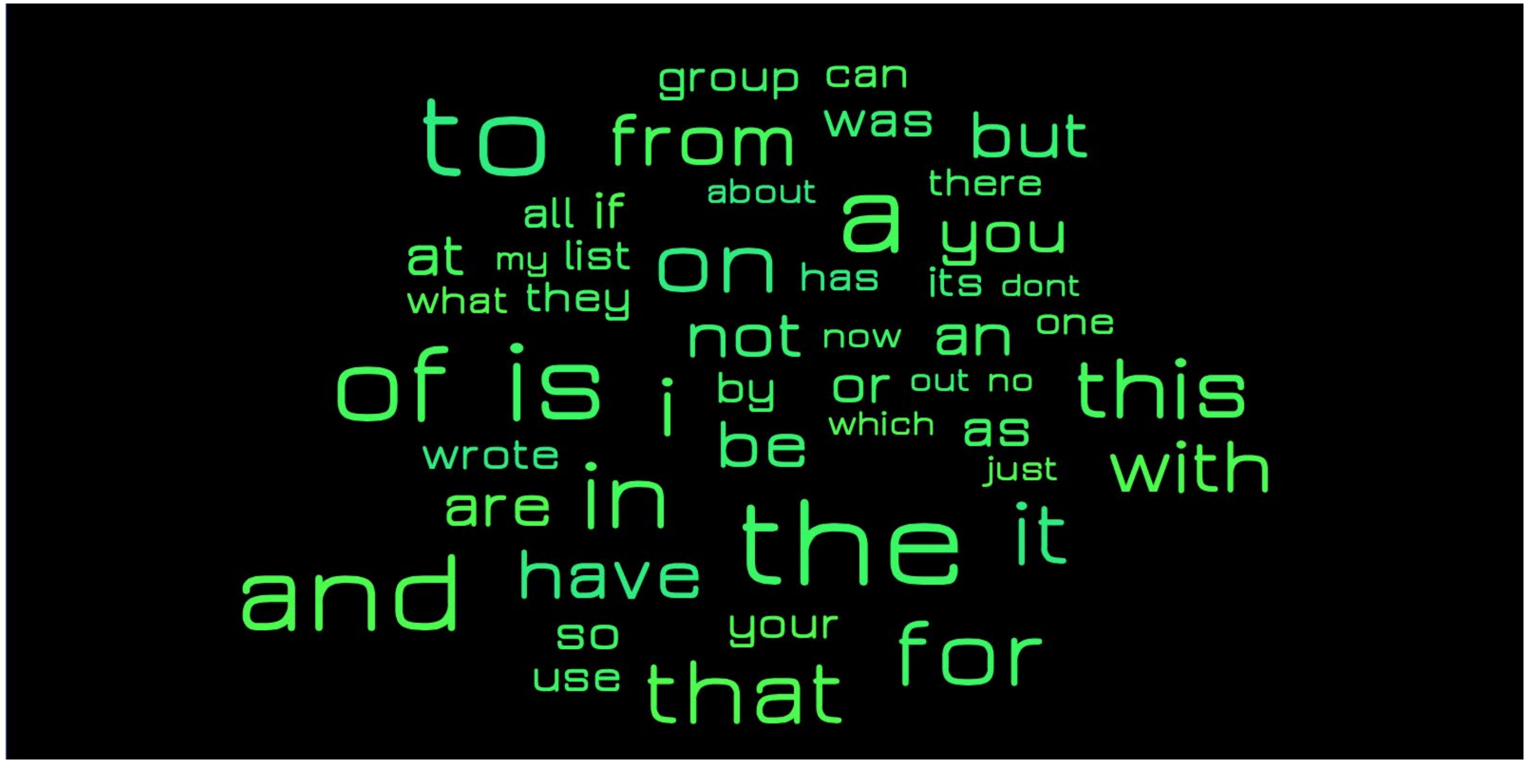
```python
print (tokens_no_punctuation)
```

```
['open', 'a', 'bank', 'account', 'the', 'nearest', 'dbs', 'bank', 'is', 'besides', 'the', 'ri
ver', 'bank', 'on', 'your', 'way', 'there', 'you', 'may', 'wish', 'to', 'donate', 'some', 'bl
ood', 'at', 'the', 'nearby', 'blood', 'bank', 'will', 'you', 'be', 'willing', 'to', 'donate',
'it', 'is', 'open', 'from', '9:00', 'a.m', 'to', '4:00', 'p.m']
```

Refer to nltk-example-6-punctuation-lowercase.ipynb

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	那麼

Example: Text Without Stop Words

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

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

Discussion:

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

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."

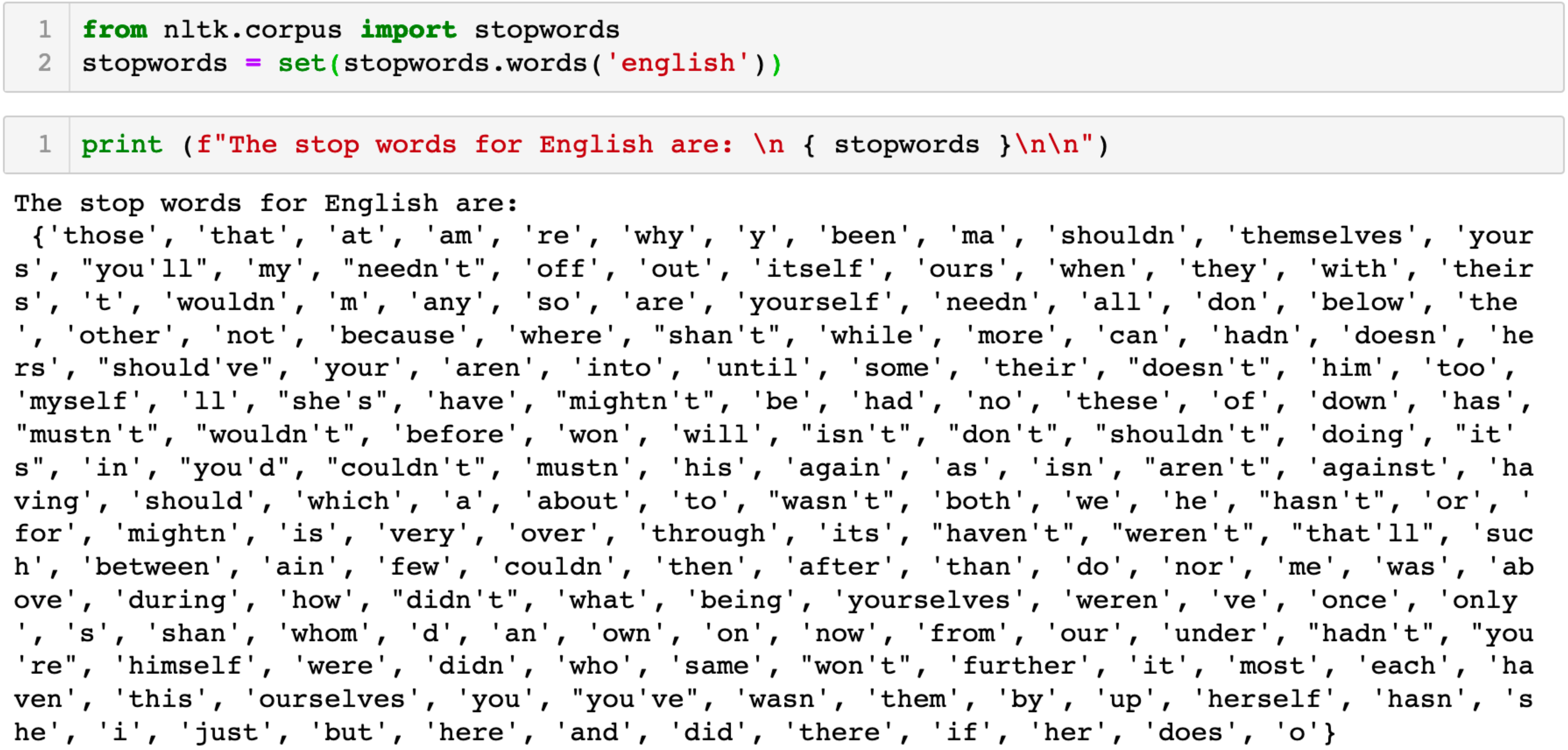
source: https://en.wikipedia.org/wiki/History_of_Humanity

Stop Words Corpus in NLTK

NLTK provides a corpus of stop words for the following languages:

Arabic, Azerbaijani, Danish, Dutch, English, Finnish
French, German, Greek, Hungarian, Indonesian, Italian
Kazakh, Nepali, Norwegian, Portuguese, Romanian
Russian, Slovene, Spanish, Swedish, Tajik

English Stop Words



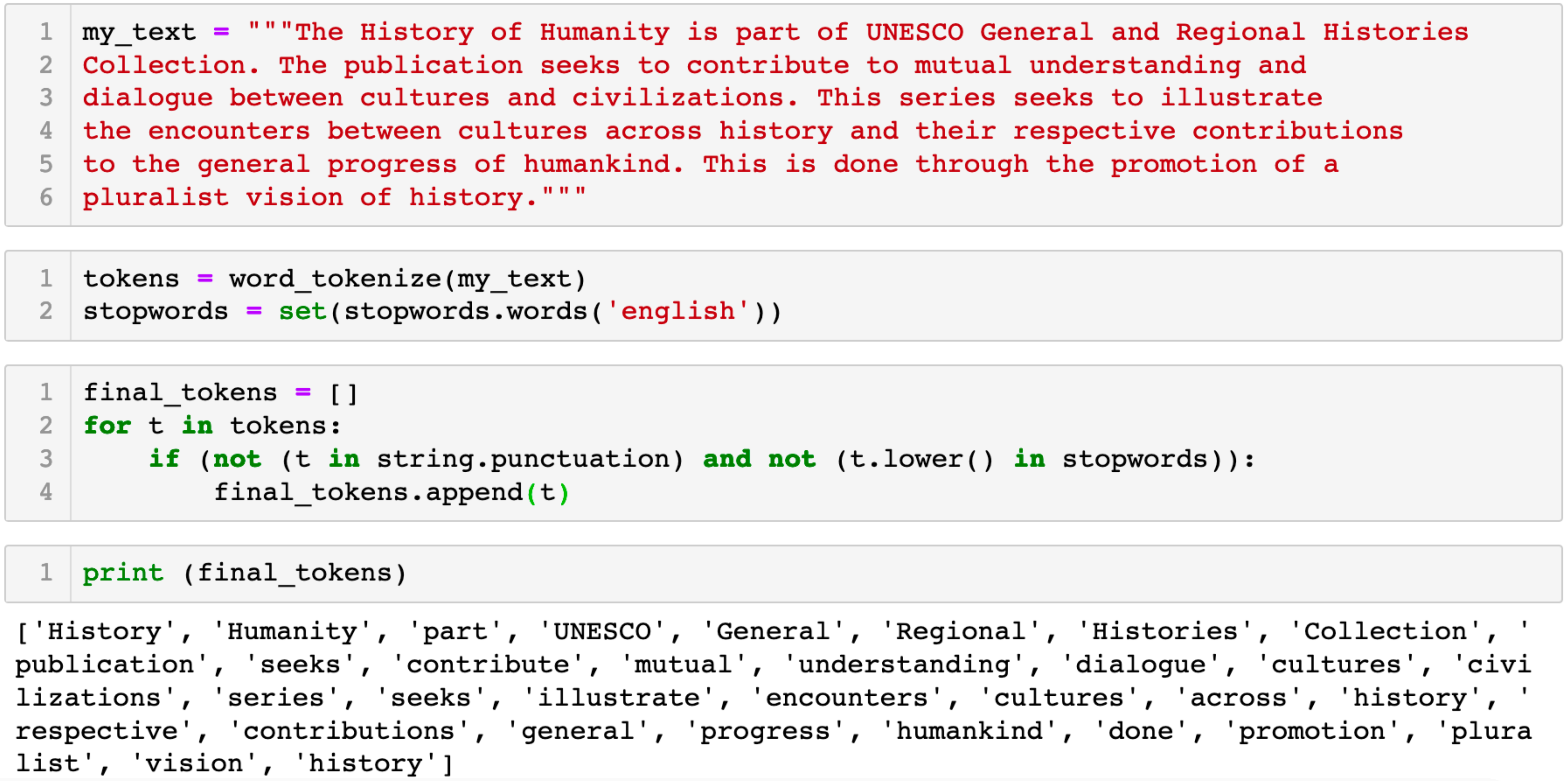
```python
from nltk.corpus import stopwords
stopwords = set(stopwords.words('english'))
```

```python
print (f"The stop words for English are: \n { stopwords }\n\n")
```

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

Refer to nltk-example-7-stopwords-corpus.ipynb

Code: Remove Stop Words from Text



```python
my_text = """The History of Humanity is part of UNESCO 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."""
```

```python
tokens = word_tokenize(my_text)
stopwords = set(stopwords.words('english'))
```

```python
final_tokens = []
for t in tokens:
    if (not (t in string.punctuation) and not (t.lower() in stopwords)):
        final_tokens.append(t)
```

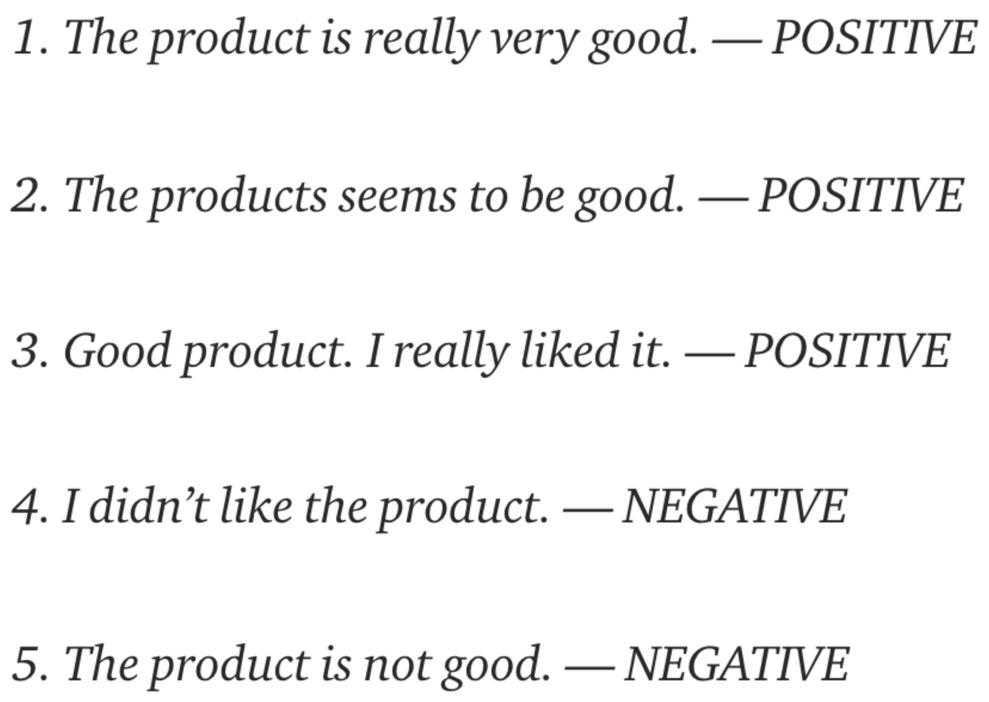
```python
print (final_tokens)
```

```
['History', 'Humanity', 'part', 'UNESCO', 'General', 'Regional', 'Histories', 'Collection', '
publication', 'seeks', 'contribute', 'mutual', 'understanding', 'dialogue', 'cultures', 'civi
lizations', 'series', 'seeks', 'illustrate', 'encounters', 'cultures', 'across', 'history', '
respective', 'contributions', 'general', 'progress', 'humankind', 'done', 'promotion', 'plura
list', 'vision', 'history']
```

Refer to nltk-example-8-using-stopwords.ipynb

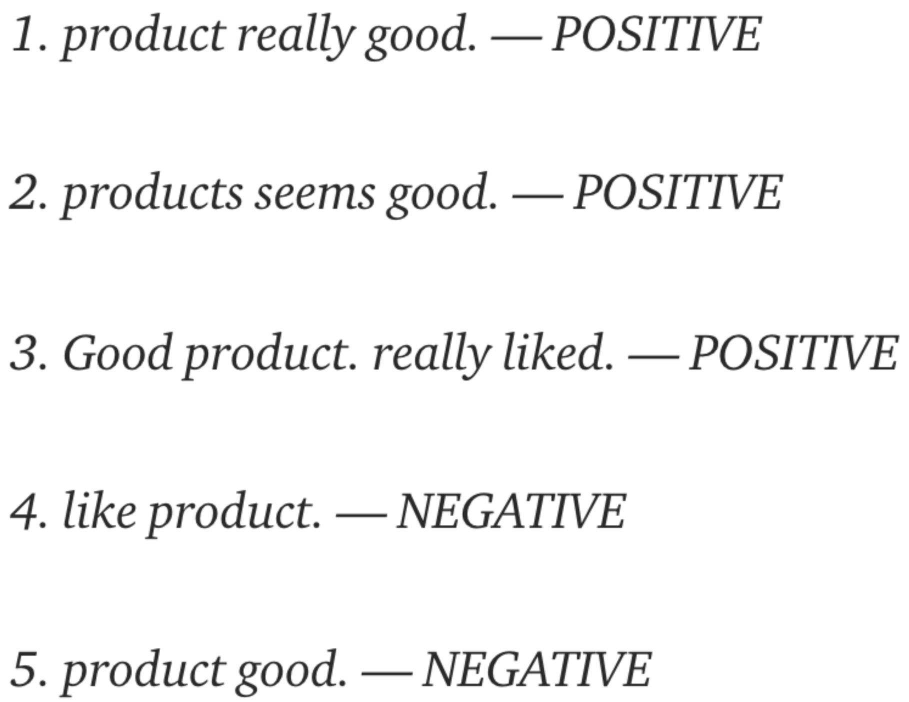
Danger: removing stop words may affect meaning

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*

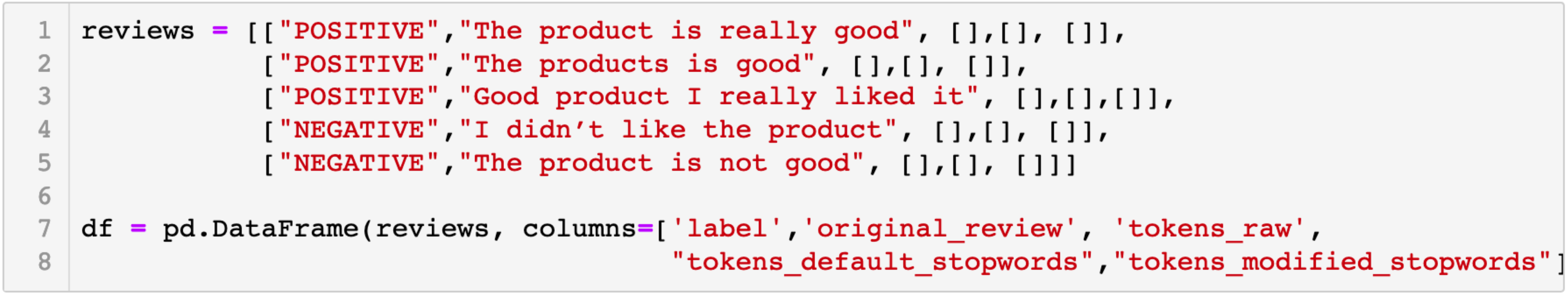
Text and labels for sentiment analysis of product reviews – after stopword removal



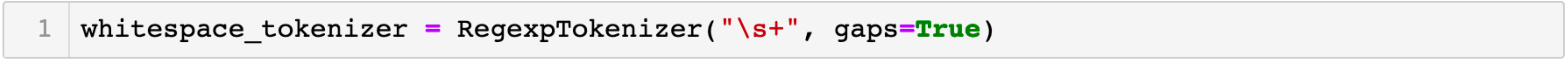
1. *product really good. — POSITIVE*
2. *products seems good. — POSITIVE*
3. *Good product. really liked. — POSITIVE*
4. *like product. — NEGATIVE*
5. *product good. — NEGATIVE*

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

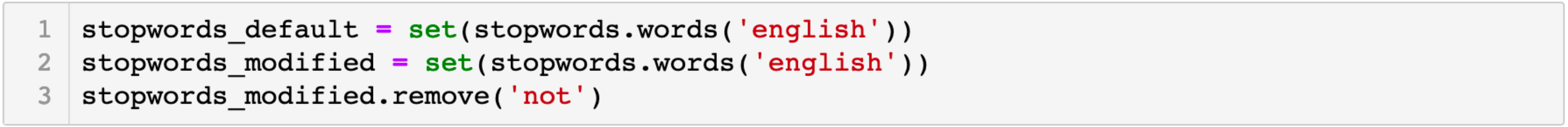
Code: Remove "not" from the Stop Words



```python
reviews = [["POSITIVE","The product is really good", [],[], []],
           ["POSITIVE","The products is good", [],[], []],
           ["POSITIVE","Good product I really liked it", [],[],[]],
           ["NEGATIVE","I didn’t like the product", [],[], []],
           ["NEGATIVE","The product is not good", [],[], []]]

df = pd.DataFrame(reviews, columns=['label','original_review', 'tokens_raw',
                                    "tokens_default_stopwords","tokens_modified_stopwords"]
```



```python
whitespace_tokenizer = RegexpTokenizer("\s+", gaps=True)
```



```python
stopwords_default = set(stopwords.words('english'))
stopwords_modified = set(stopwords.words('english'))
stopwords_modified.remove('not')
```

Refer to nltk-example-9-using-stopwords-customisation.ipynb

Continue next page →

```python
for index, row in df.iterrows():
    original_review = row['original_review'].lower()
    row['tokens_raw'] = whitespace_tokenizer.tokenize(original_review)

    for t in row['tokens_raw']:
        if not(t in stopwords_default):
            row['tokens_default_stopwords'].append(t)
        if not(t in stopwords_modified):
            row['tokens_modified_stopwords'].append(t)
```

```python
df.head()
```

	label	original_review	tokens_raw	tokens_default_stopwords	tokens_modified_stopwords
0	POSITIVE	The product is really good	[the, product, is, really, good]	[product, really, good]	[product, really, good]
1	POSITIVE	The products is good	[the, products, is, good]	[products, good]	[products, good]
2	POSITIVE	Good product I really liked it	[good, product, i, really, liked, it]	[good, product, really, liked]	[good, product, really, liked]
3	NEGATIVE	I didn’t like the product	[i, didn’t, like, the, product]	[didn’t, like, product]	[didn’t, like, product]
4	NEGATIVE	The product is not good	[the, product, is, not, good]	[product, good]	[product, not, good]

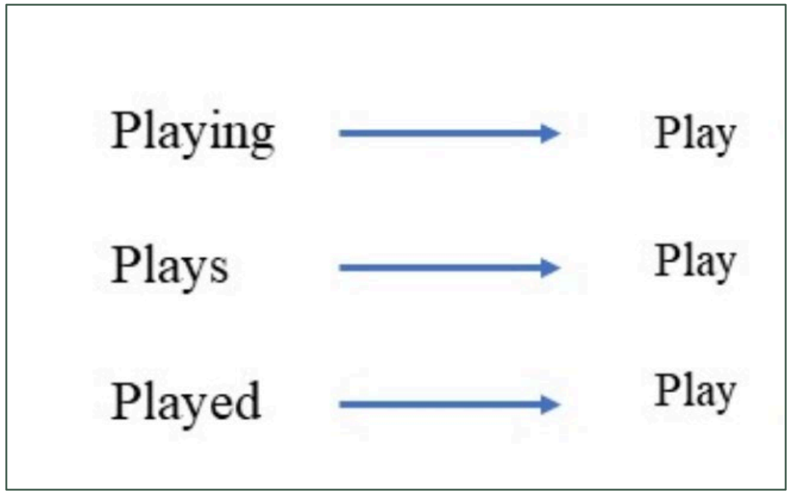
Preprocessing Techniques

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

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

Normalization – Key Idea

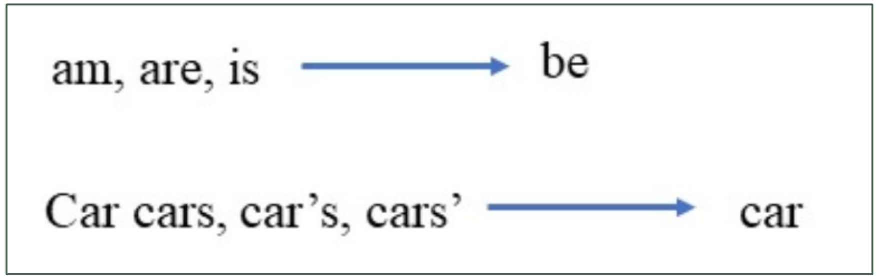
What we speak and write are made up of several words often derived from one another. Some words are derived from root words.



Playing $\rightarrow$ Play

Plays $\rightarrow$ Play

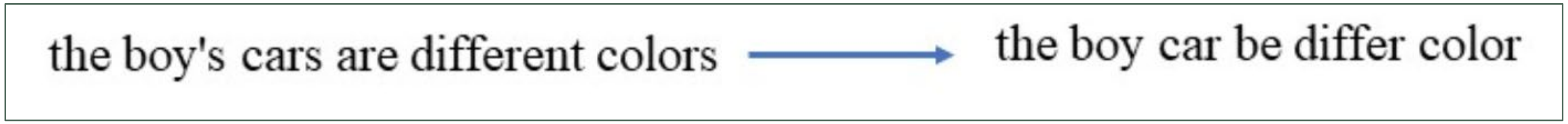
Played $\rightarrow$ Play



am, are, is $\rightarrow$ be

Car cars, car's, cars' $\rightarrow$ car

Normalized sentence



the boy's cars are different colors $\rightarrow$ the boy car be differ color

If we can cut down the number of unique words, we can reduce the size of the words in the text.

Stemming

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

Stemming refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes.

Problem
- Stemmed word does not have similar meaning to the original
- Stemmed word may not be a proper word found in the dictionary

Lemmatization

Careful , Caring → Care
Cars , car → Car
runs, running, ran → run

Lemmatization uses vocabulary knowledge and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

Code: Lemmatization

```python
lemmatizer = WordNetLemmatizer()

text = """The COVID-19 virus spreads primarily through droplets of
saliva or discharge from the nose when an infected person coughs
or sneezes, so it is important that you also practice respiratory etiquette"""

tokens_word = word_tokenize(text)
```

```python
tokens_word_lem = []

for t in tokens_words:
   tokens_word_lem.append(lemmatizer.lemmatize(t, pos='v'))
```

Treats word as verb

```python
for i in range(len(tokens_words)):
    if tokens_words[i] != tokens_word_lem[i]:
        print (f" {tokens_words[i]}  ===> {tokens_word_lem[i]}")
```

```
spreads  ===> spread
infected  ===> infect
coughs  ===> cough
sneezes  ===> sneeze
```

refer to nltk-example-10-lemmatization.ipynb

Code: Lemmatization

```python
lemmatizer = WordNetLemmatizer()

text = """The COVID-19 virus spreads primarily through droplets of
saliva or discharge from the nose when an infected person coughs
or sneezes, so it is important that you also practice respiratory etiquette.
"""

tokens_word = word_tokenize(text)
```

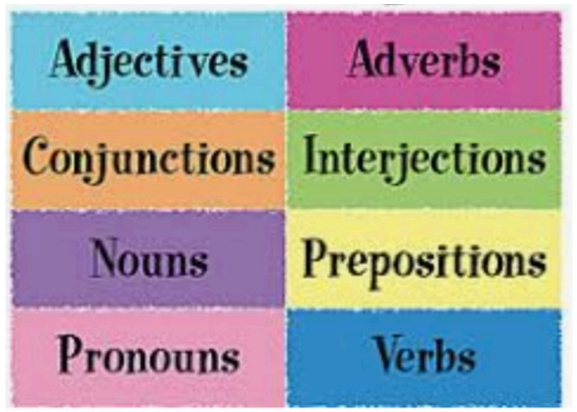
```python
tokens_word_lem = []

for t in tokens_words:
   tokens_word_lem.append(lemmatizer.lemmatize(t, pos='n'))
```

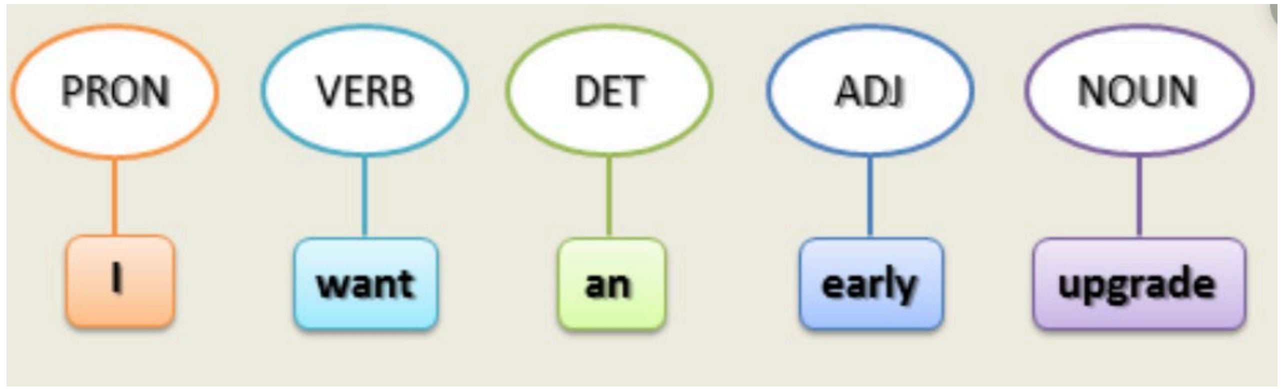
Treats word as noun

```python
for i in range(len(tokens_words)):
    if tokens_words[i] != tokens_word_lem[i]:
        print (f" {tokens_words[i]}  ===> {tokens_word_lem[i]}")
```

```
spreads  ===> spread
droplets  ===> droplet
coughs  ===> cough
sneezes  ===> sneeze
```

Preprocessing: Parts of Speech (POS) Tagging





- Sentences contain nouns, verbs, adjectives, ...
- Parts of speech tagging generates labels (grammar information) for each word
- Having tags on nouns can reveal names, entities, places etc that are mentioned in the text
- For lemmatization to work well, we need to pass the POS tags information

Code: POS Tags from Treebank tags

```python
text = """The COVID-19 virus spreads primarily through droplets of
saliva or discharge from the nose when an infected person coughs
or sneezes, so it is important that you also practice respiratory etiquette.
"""
```

```python
from nltk.tag import pos_tag
tokens_word_pos = pos_tag(word_tokenize(text))
```

```python
print (tokens_word_pos)
```

```
[('The', 'DT'), ('COVID-19', 'NNP'), ('virus', 'NN'), ('spreads', 'NNS'), ('primarily', 'RB
'), ('through', 'IN'), ('droplets', 'NNS'), ('of', 'IN'), ('saliva', 'NN'), ('or', 'CC'), ('d
ischarge', 'NN'), ('from', 'IN'), ('the', 'DT'), ('nose', 'NN'), ('when', 'WRB'), ('an', 'DT
'), ('infected', 'JJ'), ('person', 'NN'), ('coughs', 'NNS'), ('or', 'CC'), ('sneezes', 'NNS
'), (',', ','), ('so', 'IN'), ('it', 'PRP'), ('is', 'VBZ'), ('important', 'JJ'), ('that', 'IN
'), ('you', 'PRP'), ('also', 'RB'), ('practice', 'NN'), ('respiratory', 'NN'), ('etiquette',
'NN'), ('.', '.')]
```

NLTK uses the set of tags from the Penn Treebank Project
(https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html)

Refer to nltk-example-11-POSTagger.ipynb

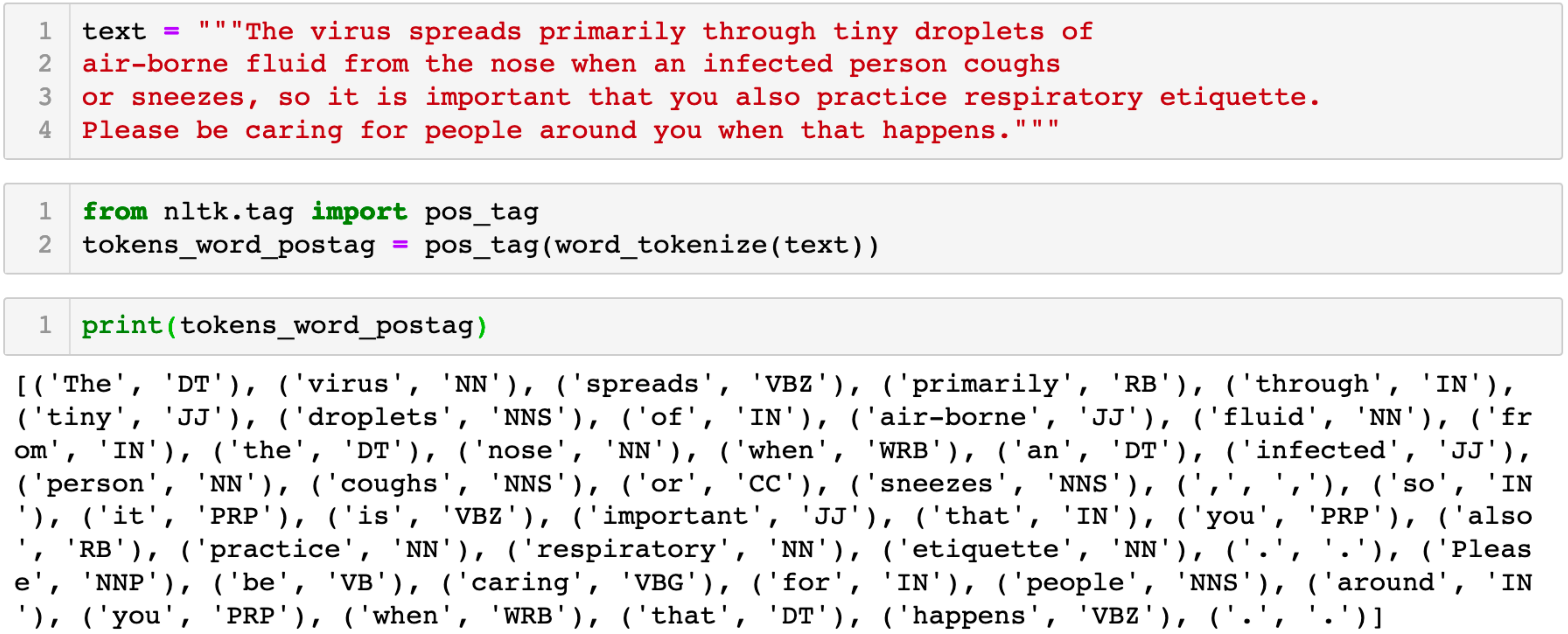
Convert between Wordnet and Treebank POS Tags

Lemmatizer uses tags follows WordNet convention - v – verb, n – noun, r – adverb, a - adjective

```python
def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return "a"
    elif treebank_tag.startswith('V'):
        return "v"
    elif treebank_tag.startswith('N'):
        return "n"
    elif treebank_tag.startswith('R'):
        return "r"
    else:
        return "n"
```

Refer to nltk-example-12-POSTagger-Lemmatizer.pynb
Continue next page →

Code: Lemmatization with POS



```
text = """The virus spreads primarily through tiny droplets of
air-borne fluid from the nose when an infected person coughs
or sneezes, so it is important that you also practice respiratory etiquette.
Please be caring for people around you when that happens."""
```

```
from nltk.tag import pos_tag
tokens_word_postag = pos_tag(word_tokenize(text))
```

```
print(tokens_word_postag)
```

```
[('The', 'DT'), ('virus', 'NN'), ('spreads', 'VBZ'), ('primarily', 'RB'), ('through', 'IN'),
('tiny', 'JJ'), ('droplets', 'NNS'), ('of', 'IN'), ('air-borne', 'JJ'), ('fluid', 'NN'), ('fr
om', 'IN'), ('the', 'DT'), ('nose', 'NN'), ('when', 'WRB'), ('an', 'DT'), ('infected', 'JJ'),
('person', 'NN'), ('coughs', 'NNS'), ('or', 'CC'), ('sneezes', 'NNS'), (',', ','), ('so', 'IN
'), ('it', 'PRP'), ('is', 'VBZ'), ('important', 'JJ'), ('that', 'IN'), ('you', 'PRP'), ('also
', 'RB'), ('practice', 'NN'), ('respiratory', 'NN'), ('etiquette', 'NN'), ('.', '.'), ('Pleas
e', 'NNP'), ('be', 'VB'), ('caring', 'VBG'), ('for', 'IN'), ('people', 'NNS'), ('around', 'IN
'), ('you', 'PRP'), ('when', 'WRB'), ('that', 'DT'), ('happens', 'VBZ'), ('.', '.')]
```

Refer to nltk-example-12-POSTagger-Lemmatizer.pynb

Continue next page →

Code: Lemmatization with POS

```python
lemmatizer = WordNetLemmatizer()
tokens_word_lem = []
for t in tokens_word_postag:
   tokens_word_lem.append(lemmatizer.lemmatize(t[0], get_wordnet_pos(t[1])))
```

```python
for i in range(len(tokens_word_postag)):
    if tokens_word_postag[i][0] != tokens_word_lem[i]:
        print (f" {tokens_word_postag[i]}  ===> {tokens_word_lem[i]}")
```

```
('spreads', 'VBZ')  ===> spread
('droplets', 'NNS')  ===> droplet
('coughs', 'NNS')  ===> cough
('sneezes', 'NNS')  ===> sneeze
('is', 'VBZ')  ===> be
('caring', 'VBG')  ===> care
('happens', 'VBZ')  ===> happen
```

Refer to nltk-example-12-POSTagger-Lemmatizer.pynb

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 words, multi-words phrases

Preprocessing: Chunking

What is the right way to tokenize **open compound words**?

We need to invest in **data science** and **artificial intelligence** capabilities.

"data", "science“ **or** "data science"

"artificial" , "intelligence" **or** "artificial intelligence"

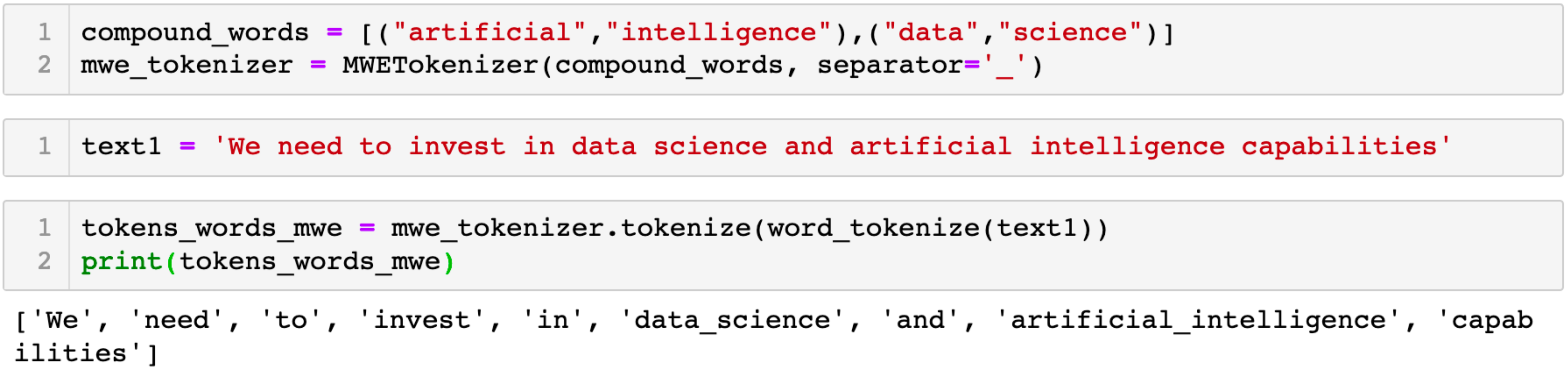
Mr **Heng Swee Keat** will deliver a Ministerial Statement on additional support measures for COVID-19 pandemic.

"heng", "swee“ **,** "keat"

or "heng swee keat"

or “heng_swee_keat”

Code: Chunking using MWE Tokenizer



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

```python
text1 = 'We need to invest in data science and artificial intelligence capabilities'
```

```python
tokens_words_mwe = mwe_tokenizer.tokenize(word_tokenize(text1))
print(tokens_words_mwe)
```

```
['We', 'need', 'to', 'invest', 'in', 'data_science', 'and', 'artificial_intelligence', 'capab
ilities']
```

Refer to nltk-example-13-Chunking-MWE.ipynb

Other cleansing issues

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

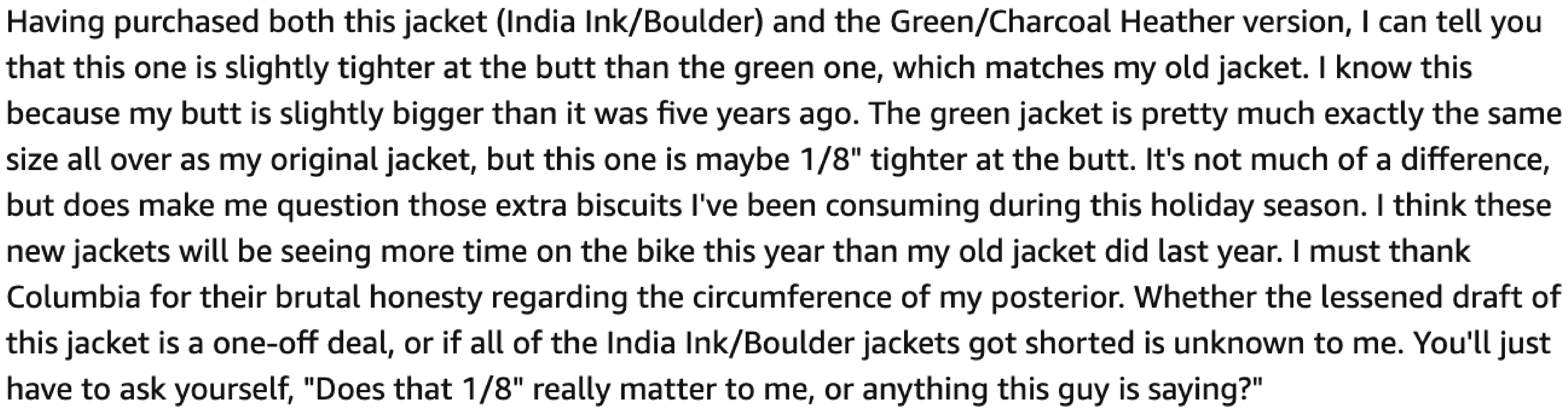
Special characters in scraped text from web pages and PDF files

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

Spelling correction - misspelled errors are encountered especially if you are processing text from crowd-sourced or informal sources

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

Optional Learning Video

NLP - Text Preprocessing and Text Classification (using Python)

- https://youtu.be/nxhCyeRR75Q

Exercise

Refer to nltk-exercise-2-text-preprocessing.ipynb

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