How to work with Text

Natural Language Processing

NLP = Natural Language Processing

- NLP is a field in machine learning with the ability of a computer to understand, analyze, manipulate, and potentially generate human language.

- Useful to all big data applications
- Especially useful for mining knowledge about people's behavior, attitude and opinions
- Express directly knowledge about our world: Small text data are also useful!

Natural Language Processing

A lot of applications

- Information Retrieval (Google finds relevant and similar results).
- Information Extraction (Gmail structures events from emails).
- Machine Translation (Google Translate translates language from one language to another).
- Text Simplification (Rewordify simplifies the meaning of sentences). Shashi Tharoor tweets could be used(pun intended).
- Sentiment Analysis (Hater News gives us the sentiment of the user).
- Text Summarization (Smmry or Reddit's autotldr gives a summary of sentences).
- Spam Filter (Gmail filters spam emails separately).
- Auto-Predict (Google Search predicts user search results).
- Auto-Correct (Google Keyboard and Grammarly correct words otherwise spelled wrong).
- Speech Recognition (Google WebSpeech or Vocalware).
- Question Answering (IBM Watson's answers to a query).
- Natural Language Generation (Generation of text from image or video data.)

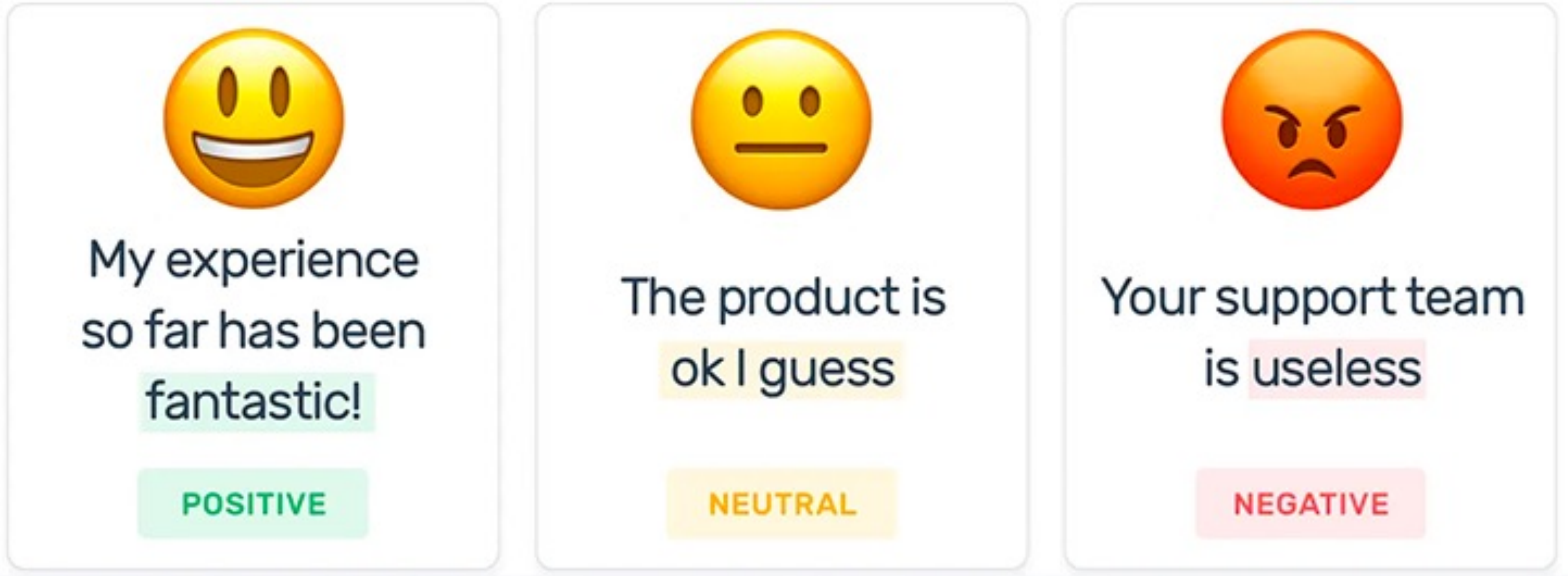
Main NLP Task

- Document classification
  - Associate a label to a document

- Sentiment analysis
  - Associate a label to a sentence

  - Sentiment analysis is the automated process of analyzing text data and classifying opinions as negative, positive or neutral. Usually, besides identifying the opinion, these systems extract attributes of the expression e.g.:
    - Polarity: if the speaker express a positive or negative opinion,
    - Subject: the thing that is being talked about,
    - Opinion holder: the person, or entity that expresses the opinion

Sentiment analysis



My experience so far has been fantastic!

POSITIVE

The product is ok I guess

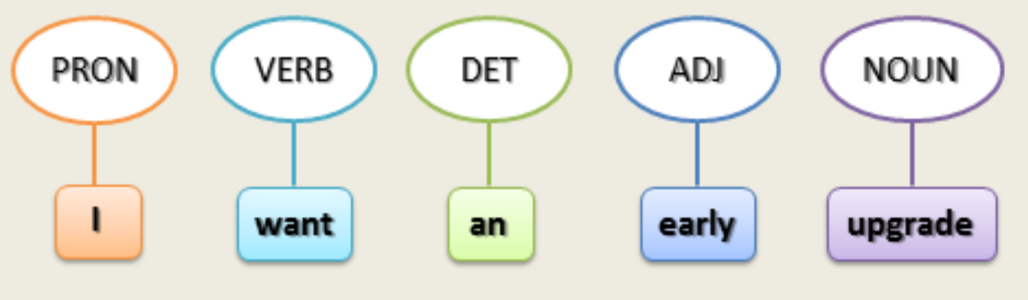
NEUTRAL

Your support team is useless

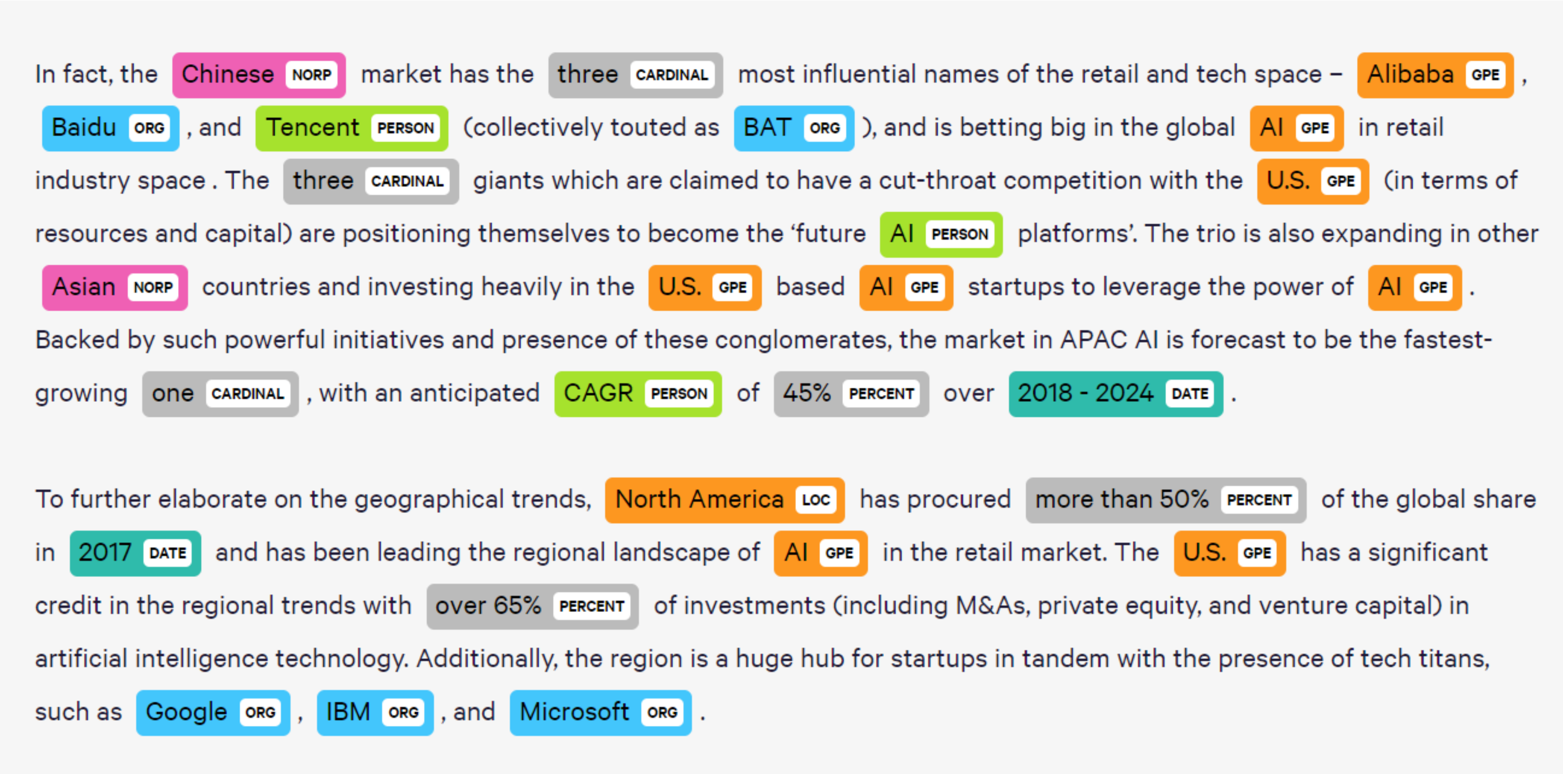
NEGATIVE

Main NLP Task 2
Associate a label to a word

- Part of speech tagging



- Naming entity recognition

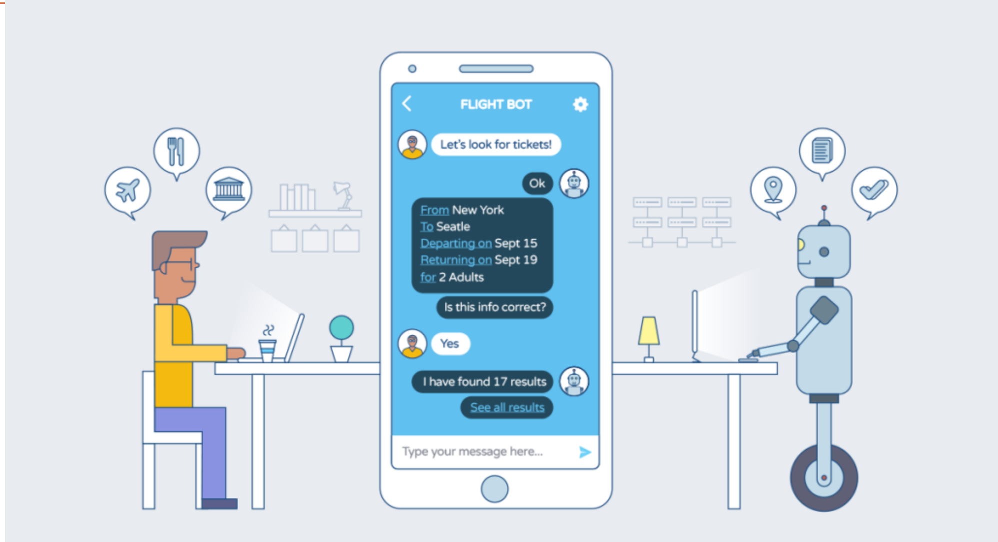


In fact, the Chinese NORP market has the three CARDINAL most influential names of the retail and tech space – Alibaba GPE , Baidu ORG , and Tencent PERSON (collectively touted as BAT ORG), and is betting big in the global AI GPE in retail industry space . The three CARDINAL giants which are claimed to have a cut-throat competition with the U.S. GPE (in terms of resources and capital) are positioning themselves to become the 'future AI PERSON platforms'. The trio is also expanding in other Asian NORP countries and investing heavily in the U.S. GPE based AI GPE startups to leverage the power of AI GPE . Backed by such powerful initiatives and presence of these conglomerates, the market in APAC AI is forecast to be the fastest-growing one CARDINAL , with an anticipated CAGR PERSON of 45% PERCENT over 2018 - 2024 DATE .

To further elaborate on the geographical trends, North America LOC has procured more than 50% PERCENT of the global share in 2017 DATE and has been leading the regional landscape of AI GPE in the retail market. The U.S. GPE has a significant credit in the regional trends with over 65% PERCENT of investments (including M&As, private equity, and venture capital) in artificial intelligence technology. Additionally, the region is a huge hub for startups in tandem with the presence of tech titans, such as Google ORG , IBM ORG , and Microsoft ORG .

Main NLP Task 3
Transform a Sentence to another Sentence

- Question & Answer (Chatbot)
- Translation



FLIGHT BOT

Let's look for tickets!

Ok

From New York
To Seatle
Departing on Sept 15
Returning on Sept 19
for 2 Adults

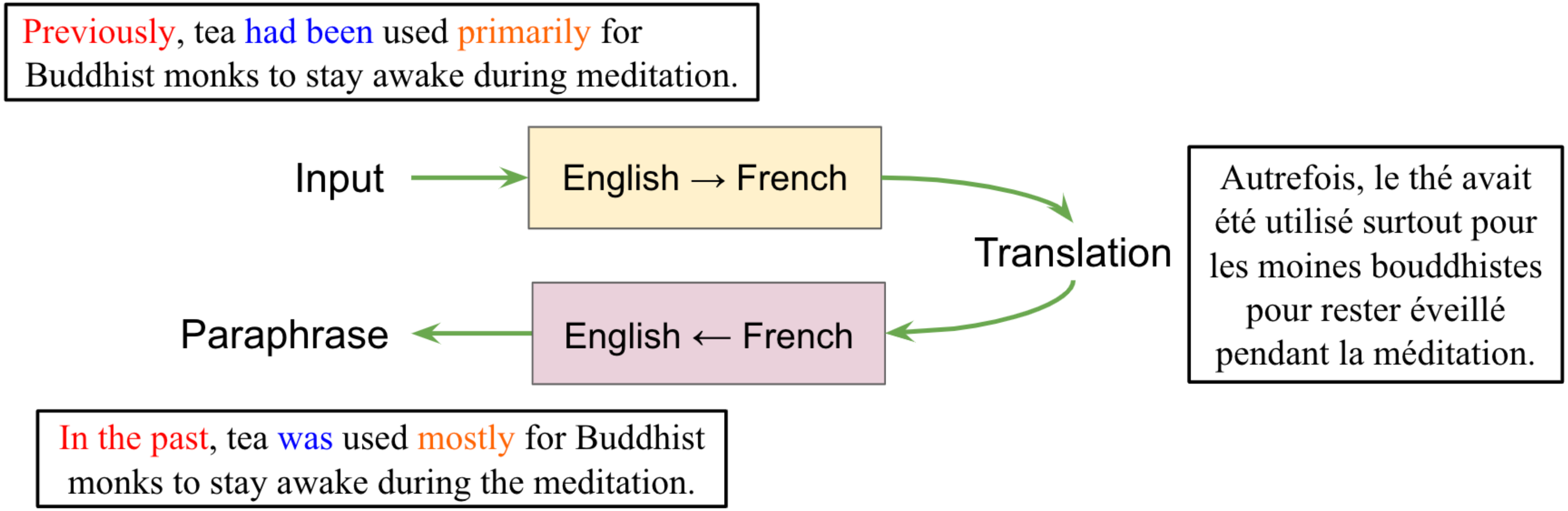
Is this info correct?

Yes

I have found 17 results

See all results

Type your message here...



Previously, tea had been used primarily for Buddhist monks to stay awake during meditation.

Input

English → French

Translation

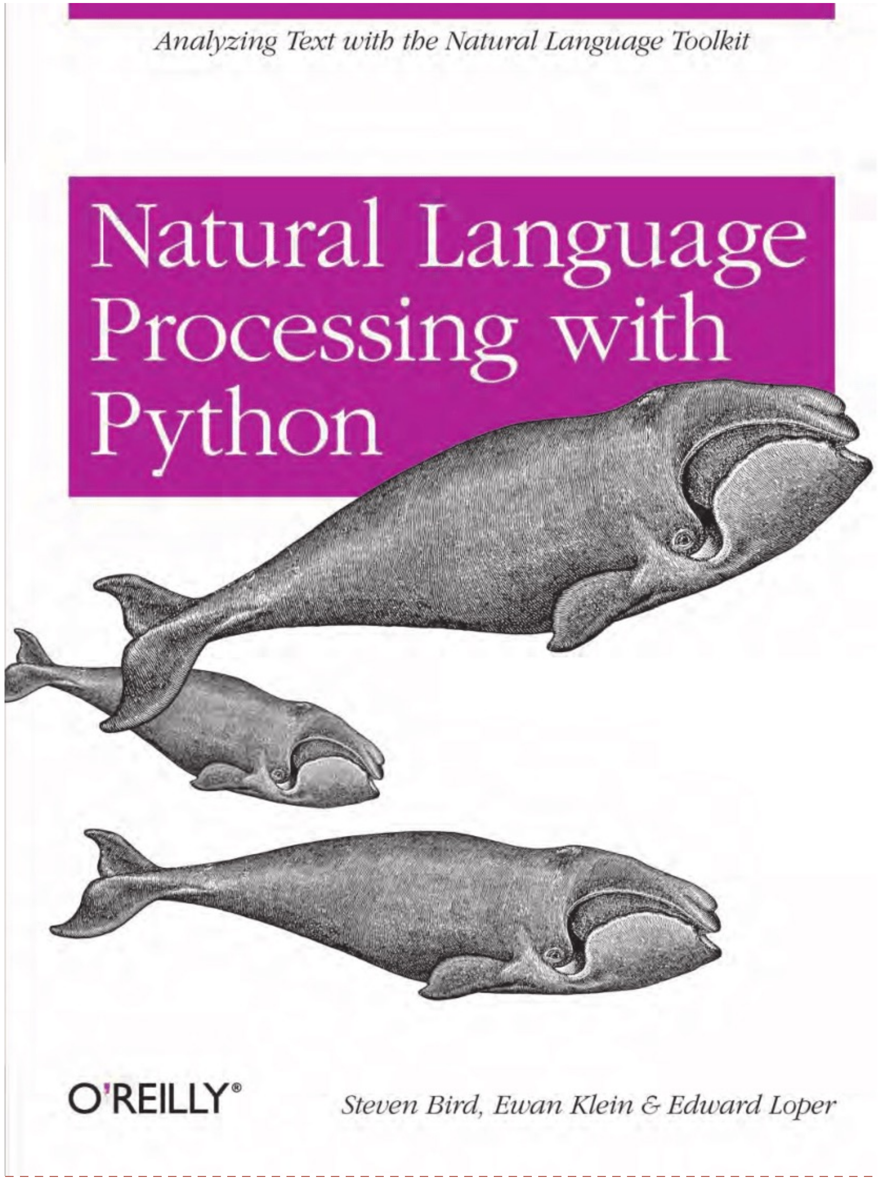
Autrefois, le thé avait été utilisé surtout pour les moines bouddhistes pour rester éveillé pendant la méditation.

English ← French

Paraphrase

In the past, tea was used mostly for Buddhist monks to stay awake during the meditation.

Two books…

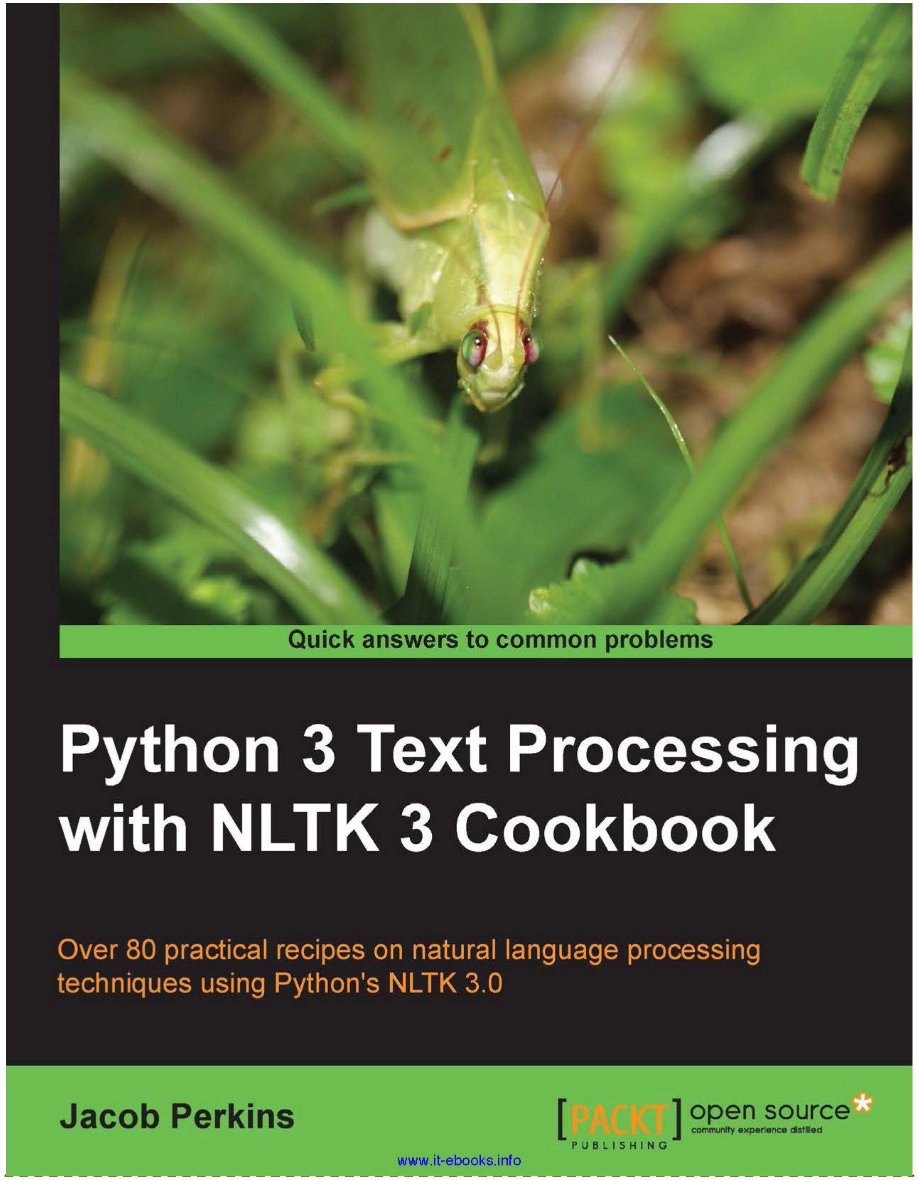


Analyzing Text with the Natural Language Toolkit

Natural Language Processing with Python

O'REILLY®

Steven Bird, Ewan Klein & Edward Loper



Quick answers to common problems

Python 3 Text Processing with NLTK 3 Cookbook

Over 80 practical recipes on natural language processing techniques using Python's NLTK 3.0

Jacob Perkins

[PACKT] PUBLISHING open source* community experience distilled

www.it-ebooks.info

And one main library

- Natural Language ToolKit (NLTK)
  - http://www.nltk.org/
  - A comprehensive Python library for natural language processing and text analytics
  - Originally designed for teaching
  - also adopted in the industry for research and development due to its usefulness and breadth of coverage
- NLTK is often used for rapid prototyping of text processing programs
- Demos of select NLTK functionality and production-ready APIs are available at http://text-processing.com
- In Python: use nltk library (http://www.nltk.org/book/)
  - `!pip install -U nltk` in the Jupyter Notebook
  - or `conda install -c conda-forge nltk`

The NLTK Pipeline

Main step for an NLP pipeline

- Text normalization
  - The set of operation depend on the task

- Main goal
  - Replace the list of chars (the original text) by a list of tokens
  - Normalize some representation : data, phone number
  - Try to reduce the vocabulary size
    - Use only lower character
    - Spelling Correction
    - Lemmatization (or Stemming)
    - Replace by synonyms (semantic reduction)

Tokenization

- Tokenization: process of splitting a string into a list of pieces (tokens).
  - A token is a piece of whole
    - A char is a token in a word
    - A word is a token in a sentence
    - A sentence is a token in paragraph
- Token != Words
  - Tokens
    - Substrings
    - Only structural
    - Data
  - Words
    - Objects
    - Contains a ‘sense’
    - Meaning
- Not always an easy task
  - ‘ ‘ between space ? One or two words
    - cats!
    - San Francisco

Python tokenization

- By default, work with english language
  - from nltk.tokenize import sent_tokenize
  - sent_tokenize(a_text)
    - Return a list of sentences

  - from nltk import word_tokenize
  - word_tokenize(a_text)
    - Return a list of word
    - Ponctuation is a word

- For other language, you have to load specific tokenizer
  - from nltk.data import load
  - spanish_tokenizer = load("tokenizers/punkt/PY3/spanish.pickle")
  - spanish_tokenizer.tokenize(a_text)
- Avalaible tokenizers are on ~/nltk_data/tokenizers/punkt/PY3

Text normalization

- Text normalization is the process of transforming text into a single canonical form
- Text normalization requires being aware of
  - What type of text is to be normalized
  - how it is to be processed afterwards
  - There is no all-purpose normalization procedure
- Easy part
  - Put the text in lower case
    - lower_text = text.lower()
      - □ Text → "This is the first sentence. A gallon of milk in the U.S. …"
      - □ Lower text → "this is the first sentence. a gallon of milk in the u.s. …"
  - Suppress '.' in acronym:
    - U.S.A → USA
  - Negation handling → depend the language
    - Replace "don't XX" by "not_XX" for example
- Difficult part
  - Phone number: +33 6 10 20 30 40 or +33.(0)6.10.20.30.40
  - Date: 11/01/2018 or 2018-01-11
  - Spelling correction
  - Etc.

Correct misspelled words

- It's a really difficult task and there's no specific approach.
  - Remove repeating character
    - I looove it
  - Spelling correction using distance between current word and a dictionary
  - Specific Neural Network (same approach than text translation)

- Use for example auto correct library

```
from autocorrect import spell


spell(‘looove’))
```

Stop word removal

- Stopwords are common words that generally do not contribute to the meaning of a sentence
  - Examples: the, as, a
- Most search engines will filter out stopwords fom search queries in order to save space in their index
- NLTK comes with a stopword corpus
  - from nltk.corpus import stopwords
  - stopwords.words('english')
    - ['i', 'me', 'my', 'myself', 'we', 'our', …
  - stopwords.words('french')
    - ['au', 'aux', 'avec', 'ce', 'ces', …
- General use
  - tokens = word_tokenize(text)
    - ['this', 'is', 'the', 'first', 'sentence', '.', 'a', …
  - [t for t in tokens if t not in english_stopwords]
    - ['first', 'sentence', '.',

Some experiment with stop word removal

- from nltk.corpus import stopwords
- print(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', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until',
'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through',
'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on',
'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where',
'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such',
'no', 'nor', **'not'**, 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will',
'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain',
**'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't",
'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma',
'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan',
"shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't",
'won', "won't", 'wouldn', "wouldn't"**]

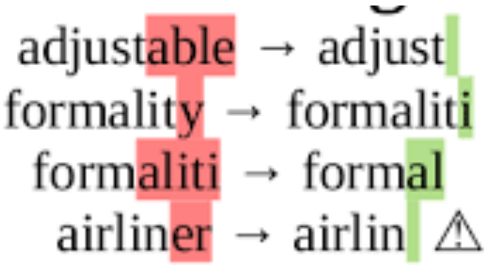
Some experiment with stop word removal

- Let's imagine you are asked to create a model that does sentiment analysis of product reviews. The dataset is fairly small that you label it your self. Consider a few reviews from the dataset.
  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*
- You performed preprocessing on data and removed all stopwords. Now, let us look what happens to the sample we selected above.
  1. *product really good. — POSITIVE*
  2. *products seems good. — POSITIVE*
  3. *Good product. really liked. — POSITIVE*
  4. *like product. — NEGATIVE ?*
  5. *product good. — NEGATIVE ?*
- Scary, right?

Reduce word forms
Stemming and Lemmatisation

Stemming

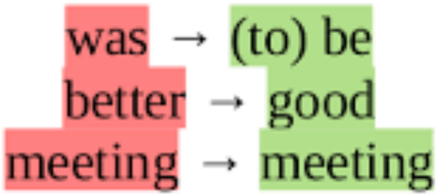
- The term "stem" generally refers to a crude heuristic process that cuts off the end of words in the hope of achieving a reduction in the forms of a word
- This often results in the removal of suffixes and sometimes prefixes
  - cats, cat → cat
  - looked → look
- May result in an unknown word



adjustable → adjust
formality → formaliti
formaliti → formal
airliner → airlin ⚠

Lemmatization

- Lemmatization generally involves doing things correctly using vocabulary and morphological analysis of words
- Reduce inflections or variant forms to base form
  - am, are, is → be
  - Jack's → Jack
- Always ends up with a known word



was → (to) be
better → good
meeting → meeting

Reduce word forms
Stemming and Lemmatisation

Stemming :

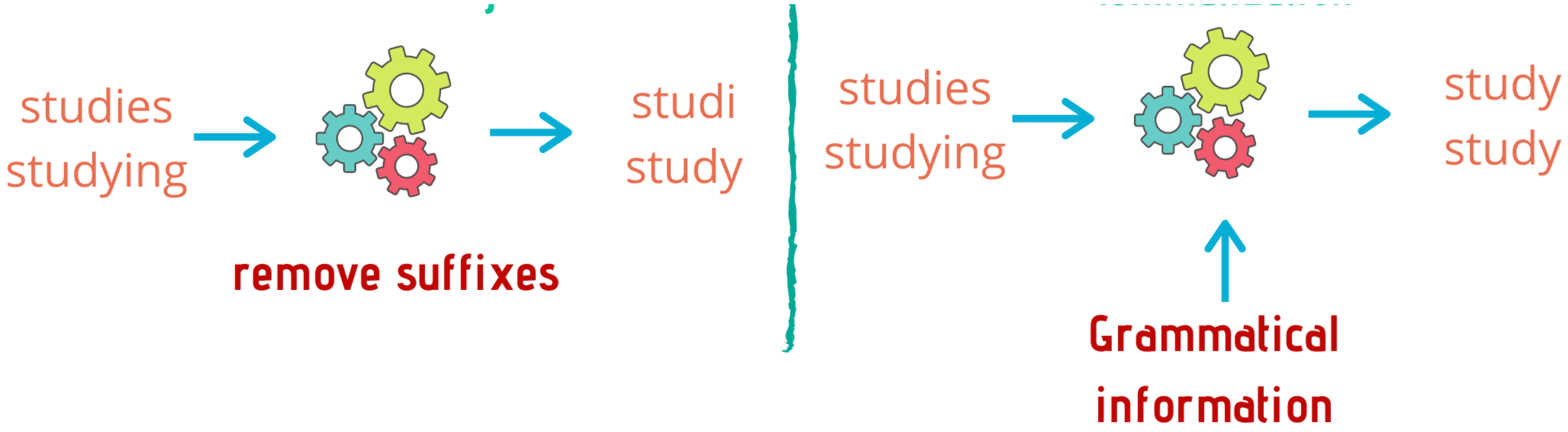
Porter algorithm

- porter = nltk.PorterStemmer()
- stemming_form = porter.stem(token)

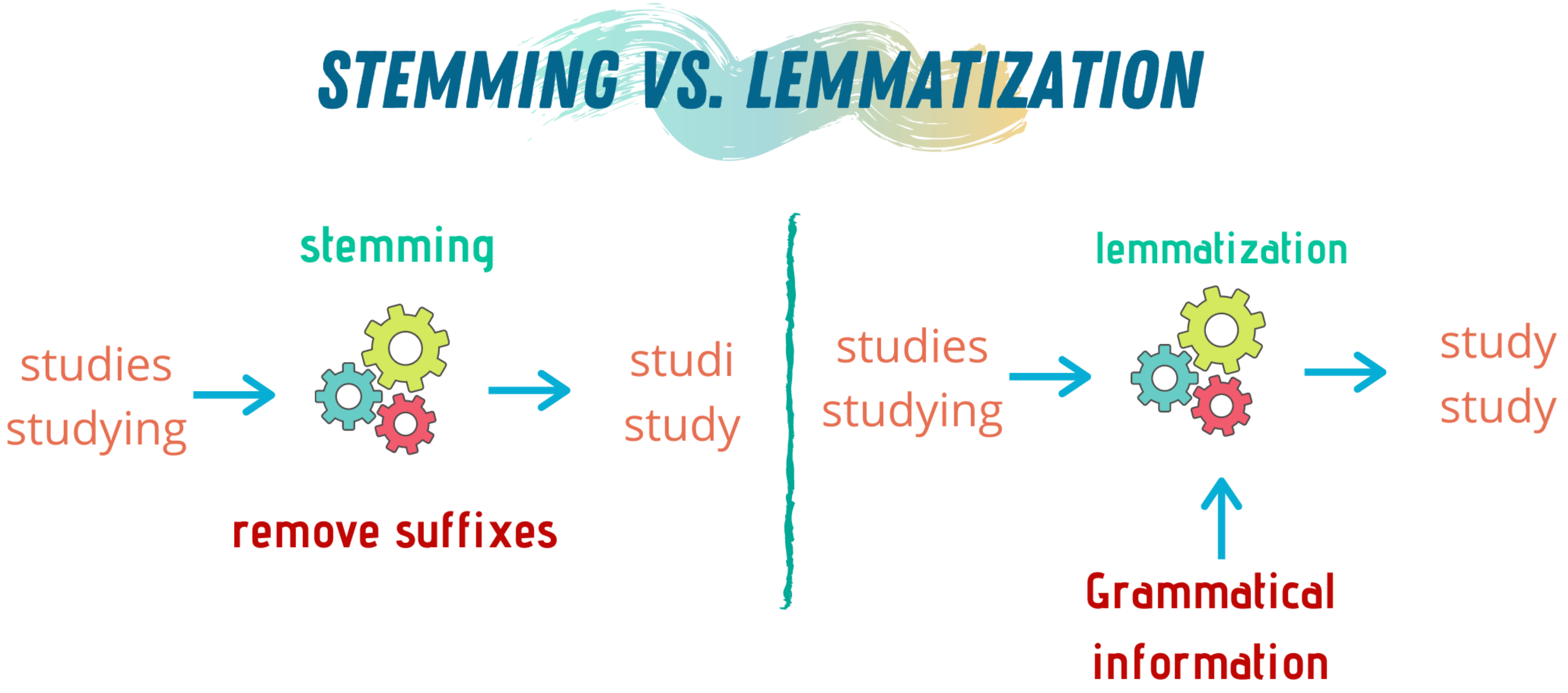
Lemmatization

Word net lemmatizer

- WNlemma = nltk.WordNetLemmatizer()
- Lemma_form = WNlemma.lemmatize(token)



Stem vs Lem


STEMMING VS. LEMMATIZATION
stemming
studies
studying
studi
study
remove suffixes
lemmatization
studies
studying
study
study
Grammatical information

Summary
Text normalization in Python

Tokenization

- Usually depends on the language, sometimes on the task
  - tokens = nltk.word_tokenize(sentence)
  - sentences = nltk.sent_tokenize(paragraph)

Normalization

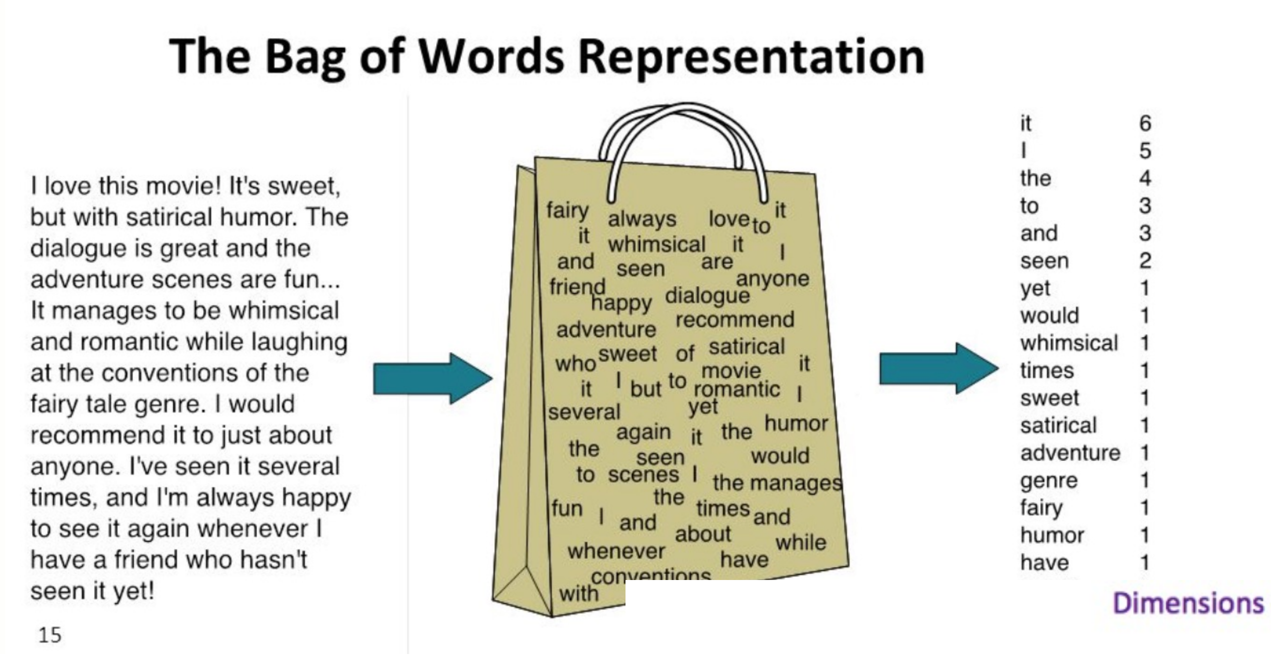
- Use only one form: lowercase for example
  - lower_text = text.lower()

Reduce vocabulary

- Stop word removal
  - from nltk.corpus import stopwords
  - tokens = [t for t in tokens if t not in stopwords.words('english')]
- Stemming: user Porter algorithm - Several algorithms available
  - porter = nltk.PorterStemmer()
  - stemming_form = porter.stem(token)
- Lemmatization - Several algorithms available
  - WNlemma = nltk.WordNetLemmatizer()
  - lemma_form = WNlemma.lemmatize(token)

Features extraction

Main approaches



The Bag of Words Representation

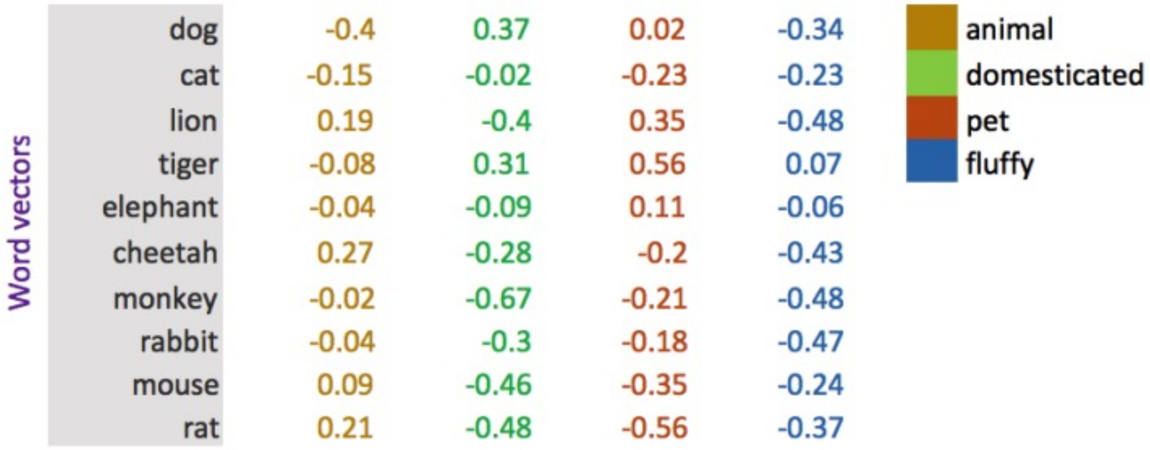
I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1

15

Bag of Words representation Today lecture

Dimensions



Word vectors					
dog	-0.4	0.37	0.02	-0.34	animal
cat	-0.15	-0.02	-0.23	-0.23	domesticated
lion	0.19	-0.4	0.35	-0.48	pet
tiger	-0.08	0.31	0.56	0.07	fluffy
elephant	-0.04	-0.09	0.11	-0.06	
cheetah	0.27	-0.28	-0.2	-0.43	
monkey	-0.02	-0.67	-0.21	-0.48	
rabbit	-0.04	-0.3	-0.18	-0.47	
mouse	0.09	-0.46	-0.35	-0.24	
rat	0.21	-0.48	-0.56	-0.37	

Vectorized representation Another lecture

Bag Of Words (BOW) - Binary

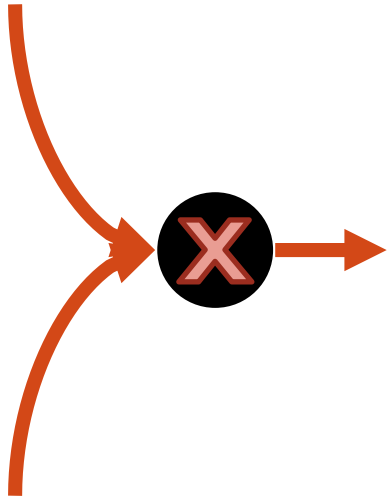
	About	Bird	Heard	Is	The	word	You
About the bird, the bird, bird, bird, bird	1	1	0	0	1	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	1	1	0

Bag Of Words (BOW) - Count

	About	Bird	Heard	Is	The	word	You
About the bird, the bird, bird, bird, bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

Bag Of Words (BOW) – Tf * Idf

TF	About	Bird	Heard	Is	The	word	You
About the bird, the bird, bird, bird, bird	1/8	5/8	0	0	2/8	0	0
You heard about the bird	1/5	1/5	1/5	0	1/5	0	1/5
The bird is the word	0	1/5	0	1/5	2/5	1/5	0



	About	Bird	Heard	Is	The	word	You
sent1	0,04	0	0	0	0	0	0
sent2	0,06	0	0,09	0	0	0	0,09
sent3	0	0	0	0,09	0	0,09	0

IDF	About	Bird	Heard	Is	The	word	You
Log(nb doc/nb word)	Log(3/2)	Log(3/3)	Log(3/1)	Log(3/1)	Log(3/3)	Log(3/1)	Log(3/1)

BOW

- Sklearn

```
from sklearn.feature_extraction.text import CountVectorizer
bow = CountVectorizer(binary=True or False)

From sklearn.feature_extraction.text import TfidfVectorizer
bow = TfidfVectorizer()

bow.fit(X_train)
X_train_features = Bow.transform(X_train)
```

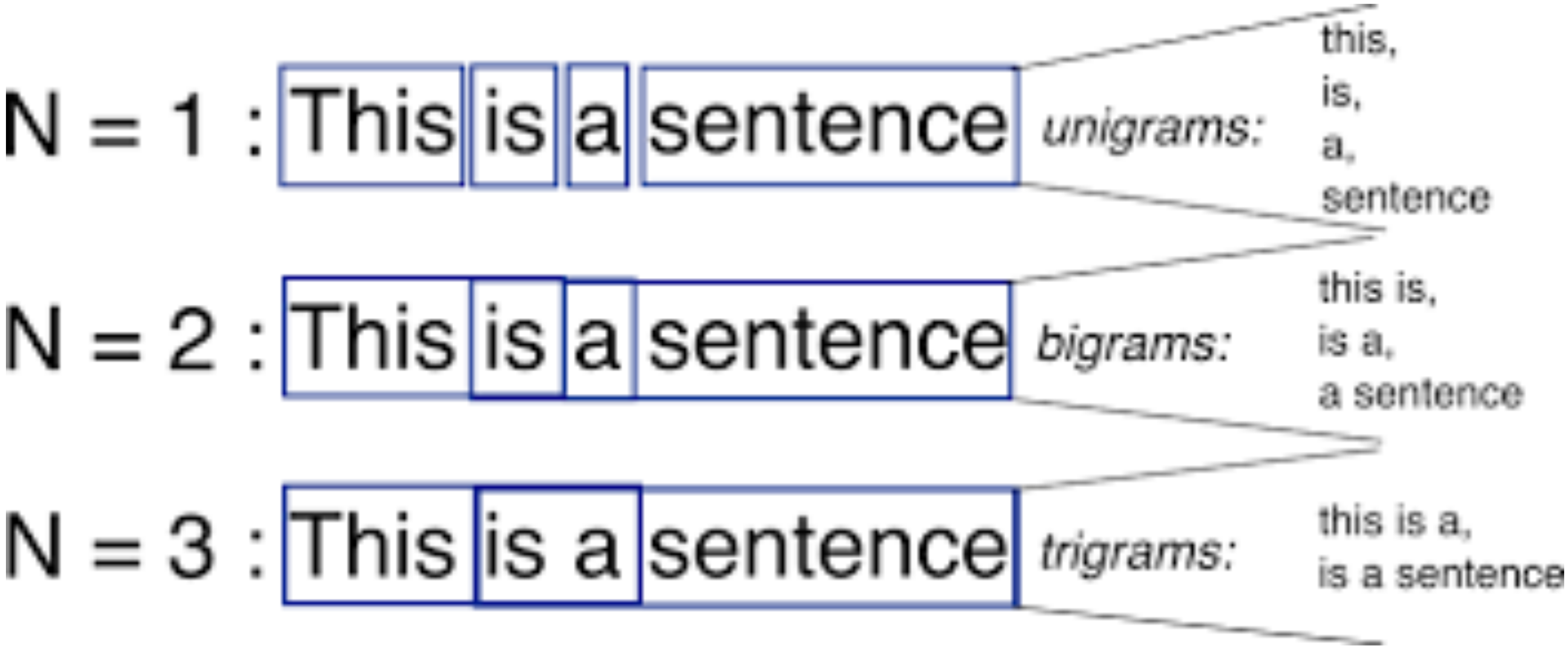
- Tf.keras

```
from tf.keras.layers import TextVectorization
bow = TextVectorization(output_mode=‘count’ or ‘tf_idf’)

bow.adapt(X_train)
Then use like a tf.keras layer
```

- A lot of others parameters → see the doc

N-gram


N = 1 :
This
is
a
sentence
unigrams:
this,
is,
a,
sentence
N = 2 :
This
is
a
sentence
bigrams:
this is,
is a,
a sentence
N = 3 :
This
is a
sentence
trigrams:
this is a,
is a sentence

N-grams

- *N*-gram of size
  - 1 is referred to as a "unigram";
  - 2 is a "bigram" (or, less commonly, a "digram
  - 3 is a "trigram"
- N-grams in NKTK:
  - from nltk import ngrams
  - For the word "hello"
    - set(ngrams("hello", 2))
      - □ {('e', 'l'), ('h', 'e'), ('l', 'l'), ('l', 'o')}
  - For the sentence "The cow jumps over the moon"
    - set(ngrams(nltk.word_tokenize("The cow jumps over the moon"), 2))
      - □ {('The', 'cow'), ('cow', 'jumps'), ('jumps', 'over'), ('over', 'the'), ('the', 'moon')})
- N-gram is also included in sklearn or keras function
  - Sklearn: `ngram_range = (1, 3)` from unigram to three-grams
  - Keras `ngrams = (1,3)` unigram and three-grams

Control the size of the feature vector

- Sklearn, a lot of parameters
  - strip_accents
  - lowercase
  - preprocessor: your own preprocessor
  - stop_words
  - max_df=0.8
  - min_df=5

- Keras, mainly 2 parameters
  - standardize: "lower_and_strip_punctuation » or "lower" or "strip_punctuation" or callable
  - max_token: None or integer, Maximum size of the vocabulary, select the most frequent token

Summary

- Text normalization
  - Reduce the form of each word
    - lower, spelling, stemming or lemming
    - Normalize acronym, date, phone number
  - Try to reduce the number of selected words
    - Fix vocabulary size
    - Remove stopwords,
    - Remove punctuation
    - Suppress unfrequent tokens (min_df)
    - Suppres to frequent tokens (max_df)
- Two possible approaches:
  - Sklearn: allows very fine tuning of the preprocessing
  - Keras: allows easy integration into a neural network