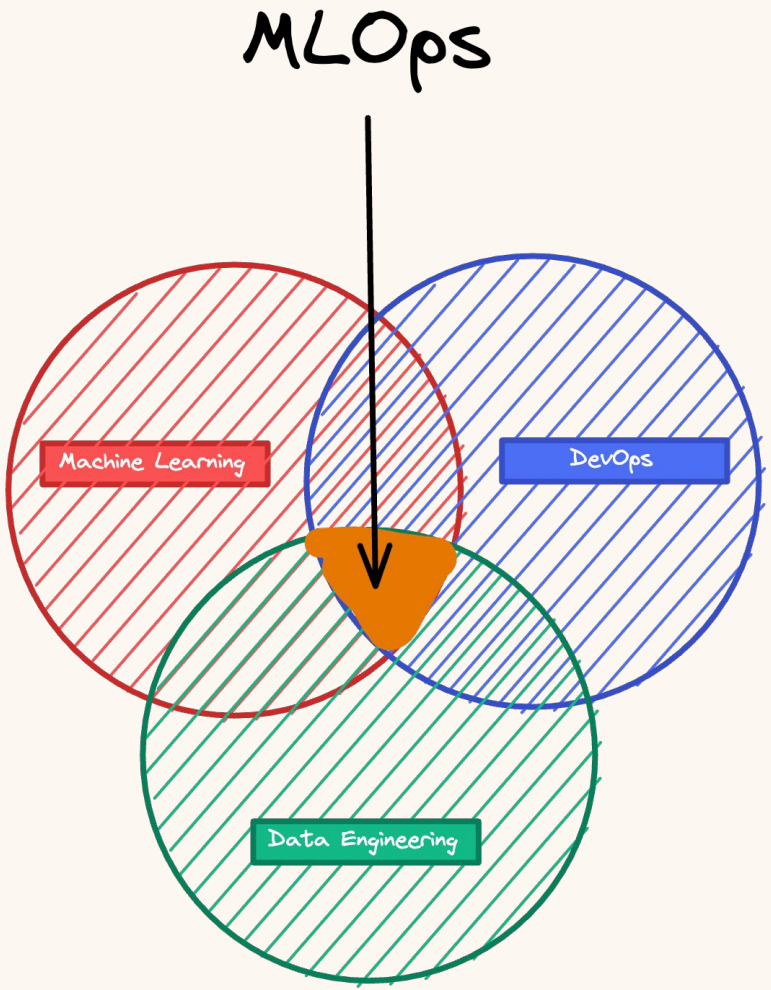
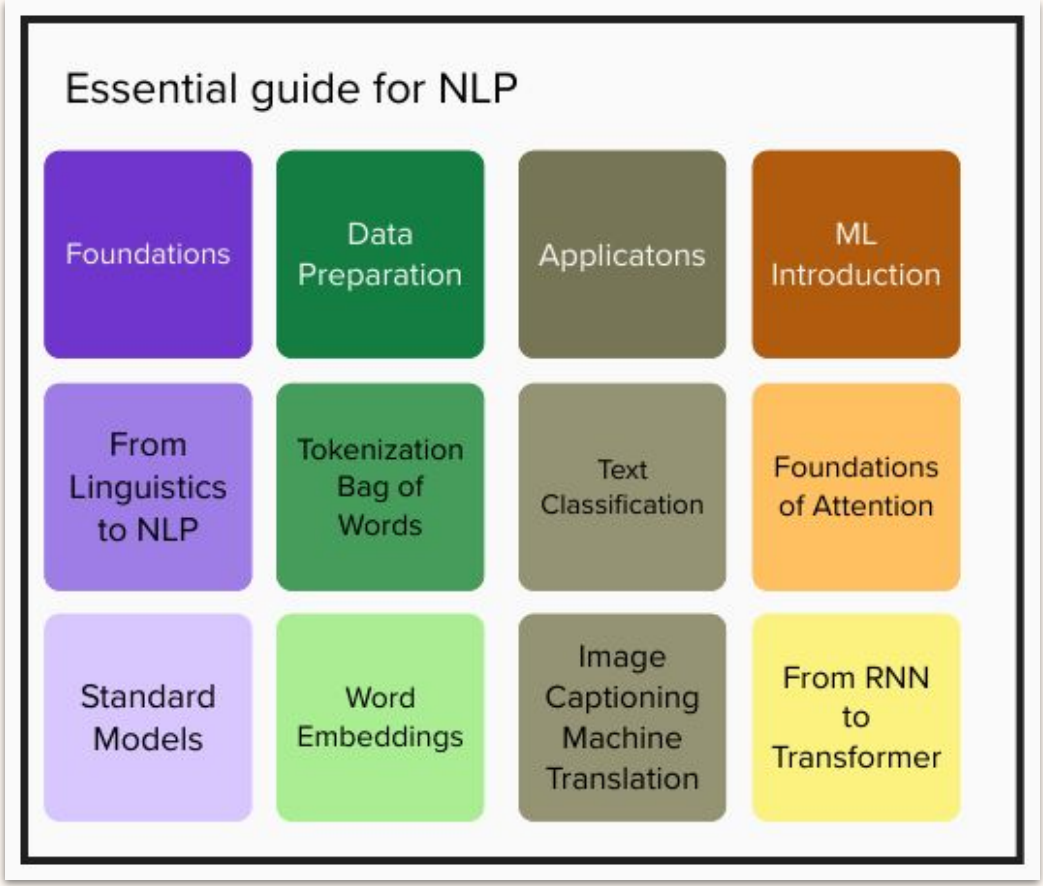
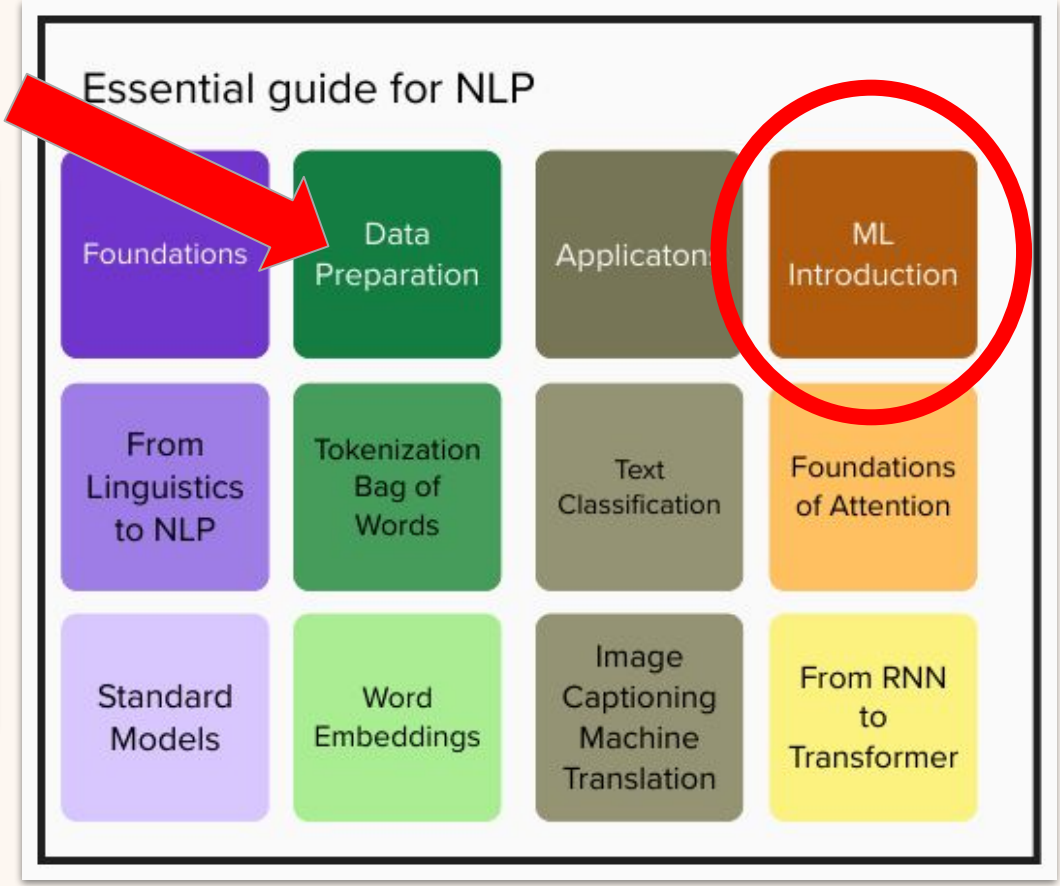
Essential Guide for NLP

DCA0305
ivanovitch.silva@ufrn.br


MLOps
Machine Learning
DevOps
Data Engineering


Essential guide for NLP
Foundations
Data Preparation
Applicatons
ML Introduction
From Linguistics to NLP
Tokenization Bag of Words
Text Classification
Foundations of Attention
Standard Models
Word Embeddings
Image Captioning Machine Translation
From RNN to Transformer

- PDF Machine Learning Fundamentals
  - What is Machine Learning (ML)? Video
  - ML types Video
  - Main challenges of ML
    - Variables, pipeline, and controlling chaos Video
    - Train, dev and test sets Video
    - Bias vs Variance Video
  - Evaluation metrics
    - How to choose an evaluation metric? Video
    - Threshold metrics Video
    - Ranking metrics Video



When dealing with textual data ...

It's essential to master three core domains

Preprocessing/Cleaning

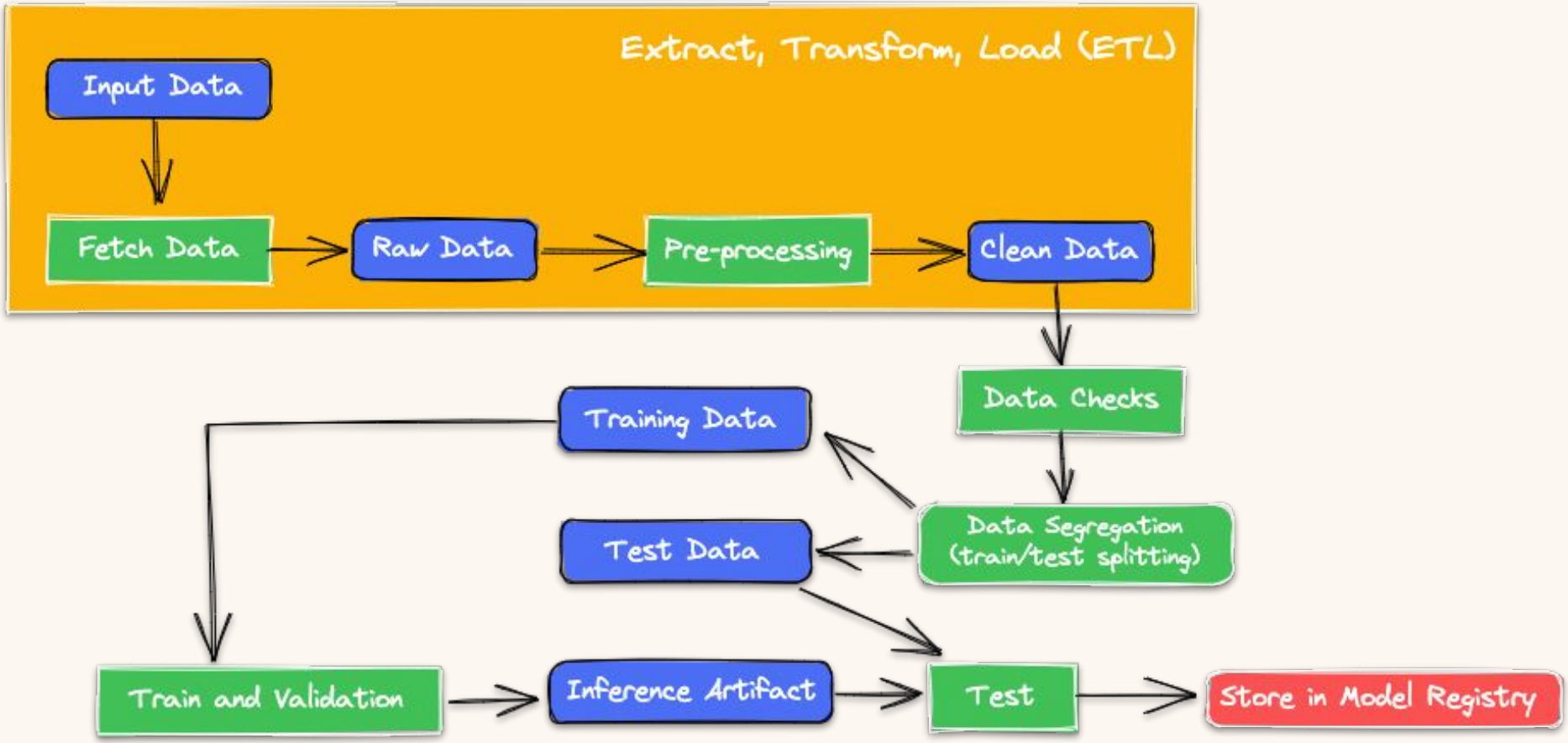
This encompasses the tasks of loading, analyzing, filtering, and refining text data before any computational modeling.

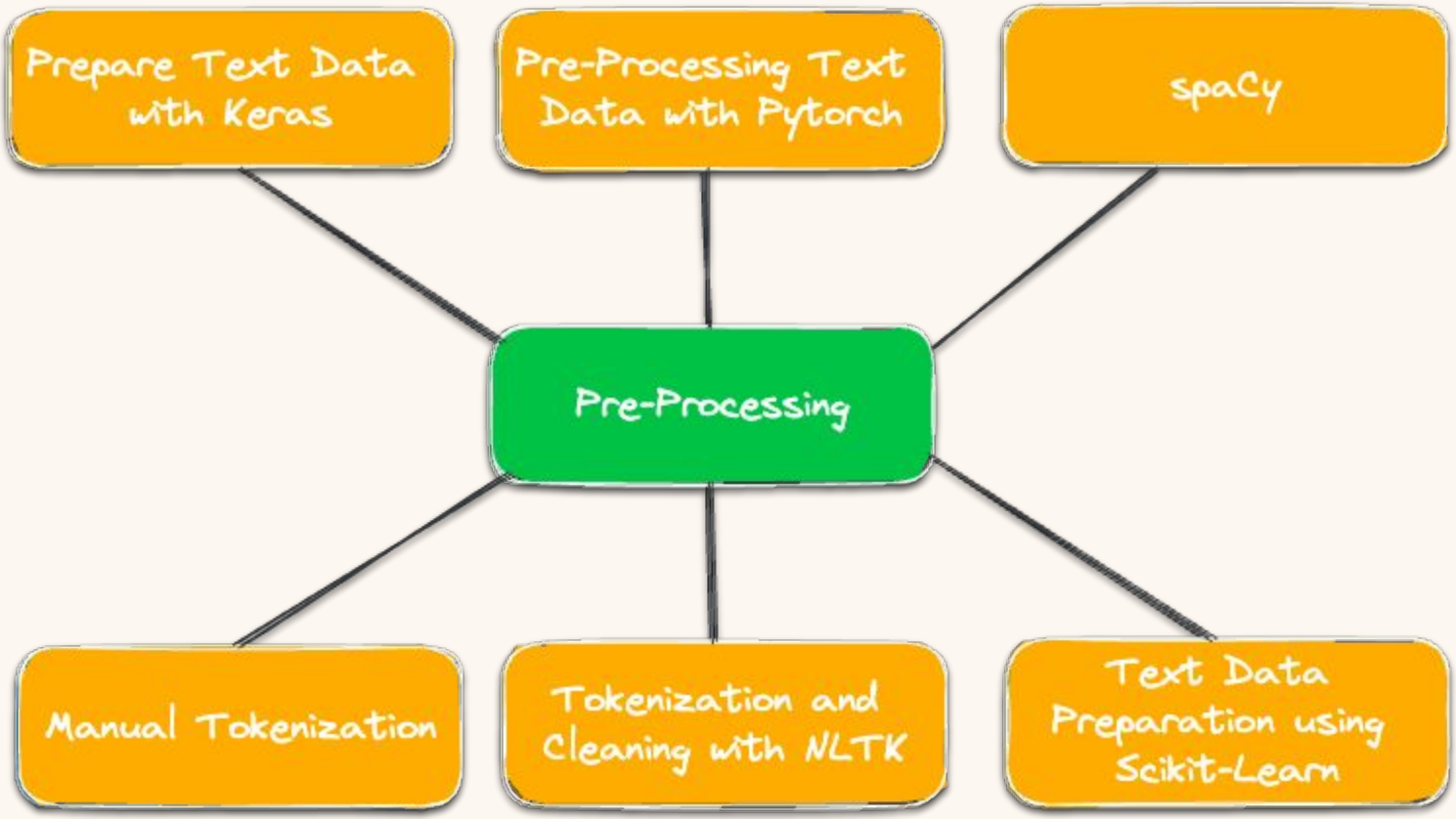
Text Representation

Beyond the traditional bag-of-words model, it's crucial to understand the advanced distributed representations like word embeddings.

Text Generation

This domain covers a spectrum of intriguing challenges, from generating image captions to facilitating machine translation.


Extract, Transform, Load (ETL)
Input Data
Fetch Data
Raw Data
Pre-processing
Clean Data
Data Checks
Training Data
Data Segregation
(train/test splitting)
Test Data
Train and Validation
Inference Artifact
Test
Store in Model Registry


Prepare Text Data with Keras
Pre-Processing Text Data with Pytorch
spaCy
Pre-Processing
Manual Tokenization
Tokenization and Cleaning with NLTK
Text Data Preparation using Scikit-Learn

Project Gutenberg

About ▾ Search and Browse ▾ Help ▾

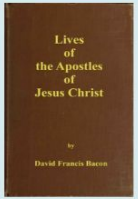
Quick search Go! Donation PayPal

Welcome to Project Gutenberg

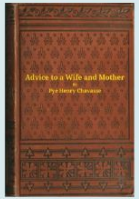
Project Gutenberg is a library of over 70,000 free eBooks

Choose among free epub and Kindle eBooks, download them or read them online. You will find the world's great literature here, with focus on older works for which U.S. copyright has expired. Thousands of volunteers digitized and diligently proofread the eBooks, for you to enjoy.

Chants for the Boer by Joaquin Miller

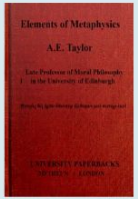


Lives of the apostles of Jesus Christ

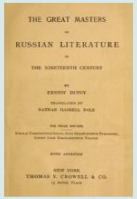


Advice to a wife and mother in two

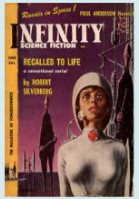
Wings of the phoenix by John Bernard



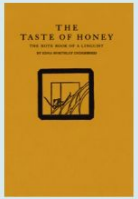
Elements of Metaphysics by A. E. Taylor



The great masters of Russian



The high ones by Poul Anderson



The taste of honey by Edna Worthley

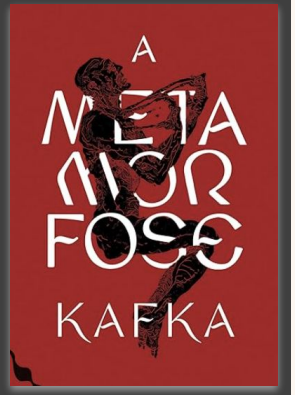
The magazine of history with notes and

Pinocchio under the sea by Gemma

Some of our latest eBooks Click Here for more latest books!

https://www.gutenberg.org

```python
import requests

# URL of the file you want to download
url = "http://www.gutenberg.org/cache/epub/5200/pg5200.txt"

# Send an HTTP GET request to the URL
response = requests.get(url)

# Check if the request was successful (status code 200)
if response.status_code == 200:
    # Open a local file to save the response content
    with open("pg5200.txt", "wb") as file:
        file.write(response.content)
    print("Download completed successfully. The file has been saved as 'pg5200.txt'")
else:
    print(f"Error downloading the file. Status code: {response.status_code}")
```



pg5200.txt

```
1 The Project Gutenberg eBook of Metamorphosis
2
3 This ebook is for the use of anyone anywhere in the United States and
4 most other parts of the world at no cost and with almost no restrictions
5 whatsoever. You may copy it, give it away or re-use it under the terms
6 of the Project Gutenberg License included with this ebook or online
7 at www.gutenberg.org. If you are not located in the United States,
8 you will have to check the laws of the country where you are located
9 before using this eBook.
10
11 *** This is a COPYRIGHTED Project Gutenberg eBook. Details Below. ***
12 ***     Please follow the copyright guidelines in this file.      ***
13
14
15 Title: Metamorphosis
16
17
18 Author: Franz Kafka
19
20 Translator: David Wyllie
21
22 Release date: August 17, 2005 [eBook #5200]
23                    Most recently updated: April 28, 2021
24
25 Language: English
26
27
28
29 *** START OF THE PROJECT GUTENBERG EBOOK METAMORPHOSIS ***
```

```
# rename the file
!mv pg5200.txt metamorphosis.txt

# delete lines 1 to 44
!sed -i '1,44d' metamorphosis.txt

# delete lines 1861 to 2225
!sed -i '1861,2225d' metamorphosis.txt
```

```python
# load text
filename = 'metamorphosis.txt'
file = open(filename, 'rt')
text = file.read()
file.close()
```

Text Cleaning is a task-specific

- Plain text, no markup.
- Translated from German to UK English.
- Text lines break every 70 characters.
- Correct punctuation, hyphens, and names like "Mr. Samsa."

```python
text
```

```
'One morning, when Gregor Samsa woke from troubled dreams, he found\nhimself transformed in his bed into
a horrible vermin. He lay on his\narmour-like back, and if he lifted his head a little he could see his\n
brown belly, slightly domed and divided by arches into stiff sections.\nThe bedding was hardly able to co
ver it and seemed ready to slide off\nany moment. His many legs, pitifully thin compared with the size of
the\nrest of him, waved about helplessly as he looked.\n\n“What’s happened to me?” he thought. It wasn’t
a dream. His room, a\nproper human room although a little too small, lay peacefully between\nits four fam
iliar walls. A collection of textile samples lay spread out\non the table—Samsa was a travelling salesman
—and above it there hung a\npicture that he had recently cut out of an illustrated magazine and\nhoused i
n a nice, gilded frame. It showed a lady fitted out with a fur\nhat and fur boa who sat upright, raising
a heavy fur muff that covered\nthe whole of her lower a...'
```

Manual Tokenization

```
# split into words by white space
words = text.split()
print(words[:100])
```

Punctuation remains intact, as seen in words like "wasn't" and "armour-like"

Punctuation marking the end of sentences remains attached to the last word, as in "sections."

```
['One', 'morning,', 'when', 'Gregor', 'Samsa' 'woke', 'from', 'troubled', 'dreams,', 'he',
'found', 'himself', 'transformed', 'in', 'his' 'bed', 'into', 'a', 'horrible', 'vermin.', 'He',
'lay', 'on', 'his', 'armour-like', 'back,', 'and' 'if', 'he', 'lifted', 'his', 'head', 'a', 'little',
'he', 'could', 'see', 'his', 'brown', 'belly,' 'slightly', 'domed', 'and', 'divided', 'by',
'arches', 'into', 'stiff', 'sections.', 'The' 'bedding', 'was', 'hardly', 'able', 'to', 'cover',
'it', 'and', 'seemed', 'ready', 'to', 'slide' 'off', 'any', 'moment.', 'His', 'many', 'legs',
'pitifully', 'thin', 'compared', 'with', 'the' 'size', 'of', 'the', 'rest', 'of', 'him,', 'waved',
'about', 'helplessly', 'as', 'he', 'looked.', 'What's', 'happened', 'to', 'me?"', 'he',
'thought', 'It', 'wasn't', 'a', 'dream.', 'His', 'room,','a', 'proper', 'human']
```

Manual Tokenization

```python
import re

# split based on words only
words = re.split(r'\W+', text)
print(words[:100])
```

"armour-like" has been split into two separate words: "armour" and "like" (which is satisfactory). However, contractions such as "What's" have also been divided into "What" and "s" (which isn't quite optimal).

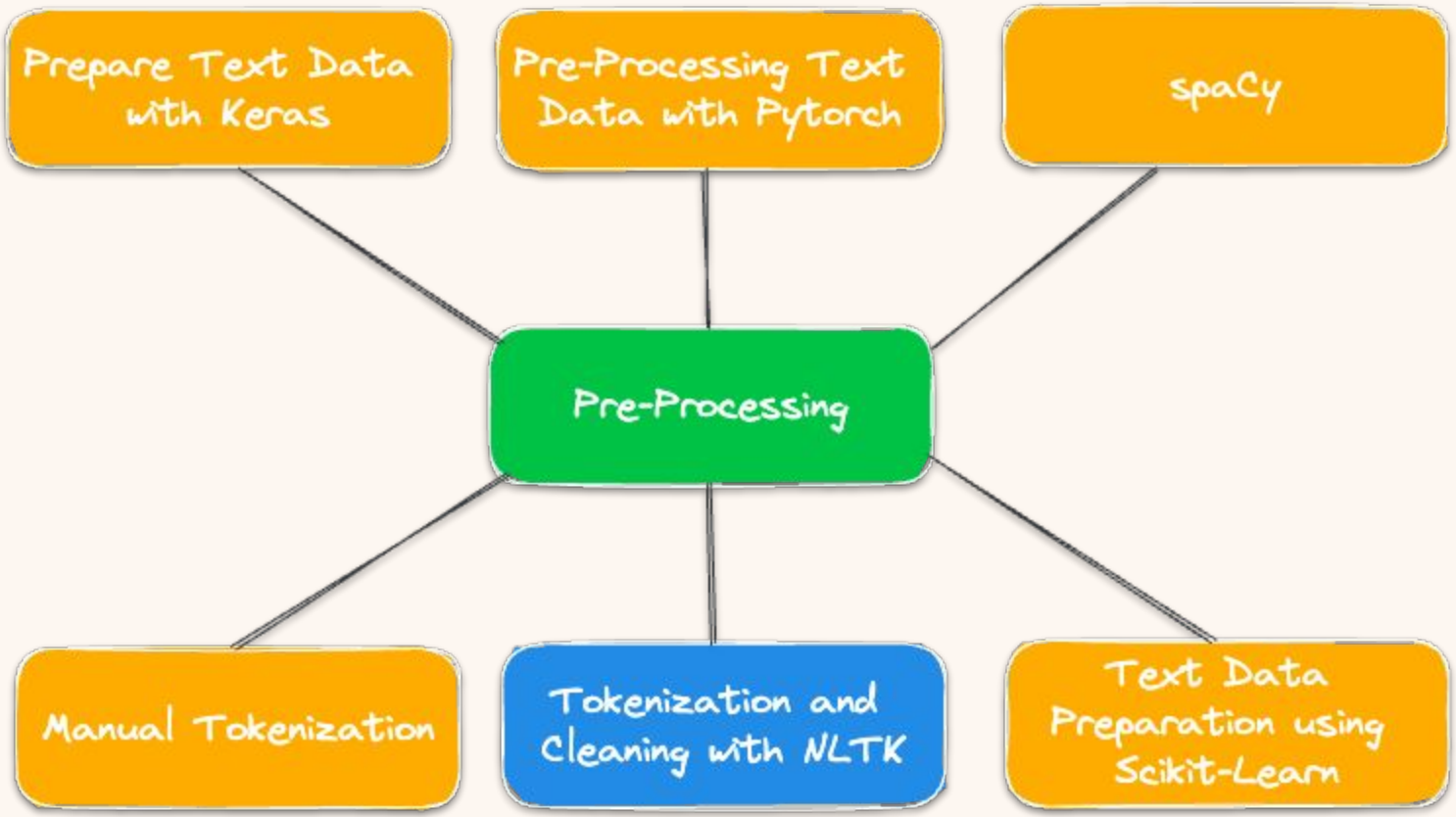
```
['One', 'morning', 'when', 'Gregor', 'Samsa', 'woke', 'from', 'troubled', 'dreams', 'he', 'found', 'himself',
'transformed', 'in', 'his', 'bed', 'into', 'a', 'horrible', 'vermin', 'He', 'lay', 'on', 'his', 'armour', 'like',
'back', 'and', 'if', 'he', 'lifted', 'his', 'head', 'a', 'little', 'he', 'could', 'see', 'his', 'brown', 'belly',
'slightly', 'domed', 'and', 'divided', 'by', 'arches', 'into', 'stiff', 'sections', 'The', 'bedding', 'was',
'hardly', 'able', 'to', 'cover', 'it', 'and', 'seemed', 'ready', 'to', 'slide', 'off', 'any', 'moment', 'His',
'many', 'legs', 'pitifully', 'thin', 'compared', 'with', 'the', 'size', 'of', 'the', 'rest', 'of', 'him', 'waved',
'about', 'helplessly', 'as', 'he', 'looked', 'What', 's', 'happened', 'to', 'me', 'he', 'thought', 'It',
'wasn', 't', 'a', 'dream', 'His', 'room']
```

Manual Tokenization

```
import string

# split into words by white space
words = text.split()

# prepare regex for char filtering
re_punc = re.compile('[%s]' % re.escape(string.punctuation))

# remove punctuation from each word
stripped = [re_punc.sub('', w) for w in words]
print(stripped[:100])
```

```
string.punctuation
!"#$%&'()*+,-./:;<=>?@[\]^_'`{|}~
```

```
# split into words by white space
words = text.split()
# convert to lower case
words = [word.lower() for word in words]
print(words[:100])
```


Prepare Text Data with Keras
Pre-Processing Text Data with Pytorch
spaCy
Pre-Processing
Manual Tokenization
Tokenization and Cleaning with NLTK
Text Data Preparation using Scikit-Learn

nltk / nltk

Type / to search

Code | Issues 249 | Pull requests 22 | Actions | Projects | Wiki | Security 1 | Insights

nltk

Public

Watch 465 | Fork 2.8k | Star 12.4k

develop | 20 branches | 42 tags | Go to file | Add file | Code

sharpblade4 minor fix for wordnet lemmatization pos param documentation ... e2d368e 3 weeks ago 14,447 commits

.github	Add Python 3.11 to CI & documentation	10 months ago
nltk	minor fix for wordnet lemmatization pos param documentation (#3190)	3 weeks ago
tools	Updated Copyright year to 2023 (#3101)	10 months ago
web	Update changelog for NLTK 3.8.1	10 months ago
.gitattributes	Introduce end-of-line normalization	11 years ago
.gitignore	Add .DS_Store to .gitignore for macOS users	last year
.pre-commit-config.yaml	ci: add labeler (#3068)	10 months ago
AUTHORS.md	minor fix for wordnet lemmatization pos param documentation (#3190)	3 weeks ago
CITATION.cff	Add `CITATION.cff` to nltk (#2880)	2 years ago
CONTRIBUTING.md	Improved the language in the file CONTRIBUTING.md (#3115)	9 months ago
ChangeLog	Update changelog for NLTK 3.8.1	10 months ago
LICENSE.txt	Use the full license text and a separate notice file	3 years ago
MANIFEST.in	Update MANIFEST.in with latest files names	3 years ago

About

NLTK Source

www.nltk.org

python | nlp | machine-learning | natural-language-processing | nltk

Readme

Apache-2.0 license

Security policy

Cite this repository

Activity

12.4k stars

465 watching

2.8k forks

Report repository

Releases

42 tags

Tokenization and Cleaning with NLTK

```
sudo pip install -U nltk
```

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

```
!python -m nltk.downloader all
```

```
import nltk
nltk.__version__
3.8.1
```

```
from nltk import sent_tokenize

# load data
filename = 'metamorphosis.txt'
file = open(filename, 'rt')
text = file.read()
file.close()

# split into sentences
sentences = sent_tokenize(text)
print(sentences[0])
```

```
One morning, when Gregor Samsa woke from troubled
dreams, he found himself transformed in his bed into
a horrible vermin.
```

Tokenization and Cleaning with NLTK

```
from nltk.tokenize import word_tokenize

# split into words
tokens = word_tokenize(text)
print(tokens[:100])
```

Segment tokens considering white space and punctuation. For instance, "What's" is tokenized into "What" and "'s"). Quotation marks remain intact, among other features.

```
['One', 'morning', ',', 'when', 'Gregor', 'Samsa', 'woke', 'from', 'troubled', 'dreams', ',', 'he',
'found', 'himself', 'transformed', 'in', 'his', 'bed', 'into', 'a', 'horrible', 'vermin', '.',
'He', 'lay', 'on', 'his', 'armour-like', 'back', ',', 'and', 'if', 'he', 'lifted', 'his', 'head', 'a',
'little', 'he', 'could', 'see', 'his', 'brown', 'belly', ',', 'slightly', 'domed', 'and',
'divided', 'by', 'arches', 'into', 'stiff', 'sections', '.', 'The', 'bedding', 'was', 'hardly',
'able', 'to', 'cover', 'it', 'and', 'seemed', 'ready', 'to', 'slide', 'off', 'any', 'moment', '.',
'His', 'many', 'legs', ',', 'pitifully', 'thin', 'compared', 'with', 'the', 'size', 'of', 'the',
'rest', 'of', 'him', ',', 'waved', 'about', 'helplessly', 'as', 'he', 'looked', '.', '“',
'What', '’', 's', 'happened']
```

Tokenization and Cleaning with NLTK

```python
from nltk.tokenize import word_tokenize

# split into words
tokens = word_tokenize(text)

# remove all tokens that are not alphabetic
words = [word for word in tokens if word.isalpha()]
print(words[:100])
```

```
['One', 'morning', 'when', 'Gregor', 'Samsa', 'woke', 'from', 'troubled', 'dreams', 'he', 'found', 'himself',
 'transformed', 'in', 'his', 'bed', 'into', 'a', 'horrible', 'vermin', 'He', 'lay', 'on', 'his', 'back', 'and',
'if', 'he', 'lifted', 'his', 'head', 'a', 'little', 'he', 'could', 'see', 'his', 'brown', 'belly', 'slightly',
'domed', 'and', 'divided', 'by', 'arches', 'into', 'stiff', 'sections', 'The', 'bedding', 'was', 'hardly',
'able', 'to', 'cover', 'it', 'and', 'seemed', 'ready', 'to', 'slide', 'off', 'any', 'moment', 'His', 'many', 'legs',
'pitifully', 'thin', 'compared', 'with', 'the', 'size', 'of', 'the', 'rest', 'of', 'him', 'waved', 'about',
'helplessly', 'as', 'he', 'looked', 'What', 's', 'happened', 'to', 'me', 'he', 'thought', 'It', 'wasn', 't', 'a',
 'dream', 'His', 'room', 'a', 'proper']
```

Tokenization and Cleaning with NLTK

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

```
['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',
'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", 'no',
'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"]
```

```python
# load data
filename = 'metamorphosis.txt'
file = open(filename, 'rt')
text = file.read()
file.close()

# split into words
tokens = word_tokenize(text)

# convert to lower case
tokens = [w.lower() for w in tokens]

# prepare regex for char filtering
re_punc = re.compile('[%s]' % re.escape(string.punctuation))

# remove punctuation from each word
stripped = [re_punc.sub('', w) for w in tokens]

# remove remaining tokens that are not alphabetic
words = [word for word in stripped if word.isalpha()]

# filter out stop words
stop_words = set(stopwords.words('english'))
words = [w for w in words if not w in stop_words]
print(words[:100])
```

1. Load the raw text.
2. Tokenize the text.
3. Convert tokens to lowercase.
4. Strip each token of punctuation.
5. Retain only alphabetic tokens.
6. Exclude tokens identified as stop words.

Tokenization and Cleaning with NLTK

```
from nltk.tokenize import word_tokenize
from nltk.stem.porter import PorterStemmer

# load data
filename = 'metamorphosis.txt'
file = open(filename, 'rt')
text = file.read()
file.close()

# split into words
tokens = word_tokenize(text)

# stemming of words
porter = PorterStemmer()

stemmed = [porter.stem(word) for word in tokens]
print(stemmed[:100])
```

Stemming is the process of truncating words to their fundamental form or base. For instance, fishing, fished, and fisher all stem to fish.

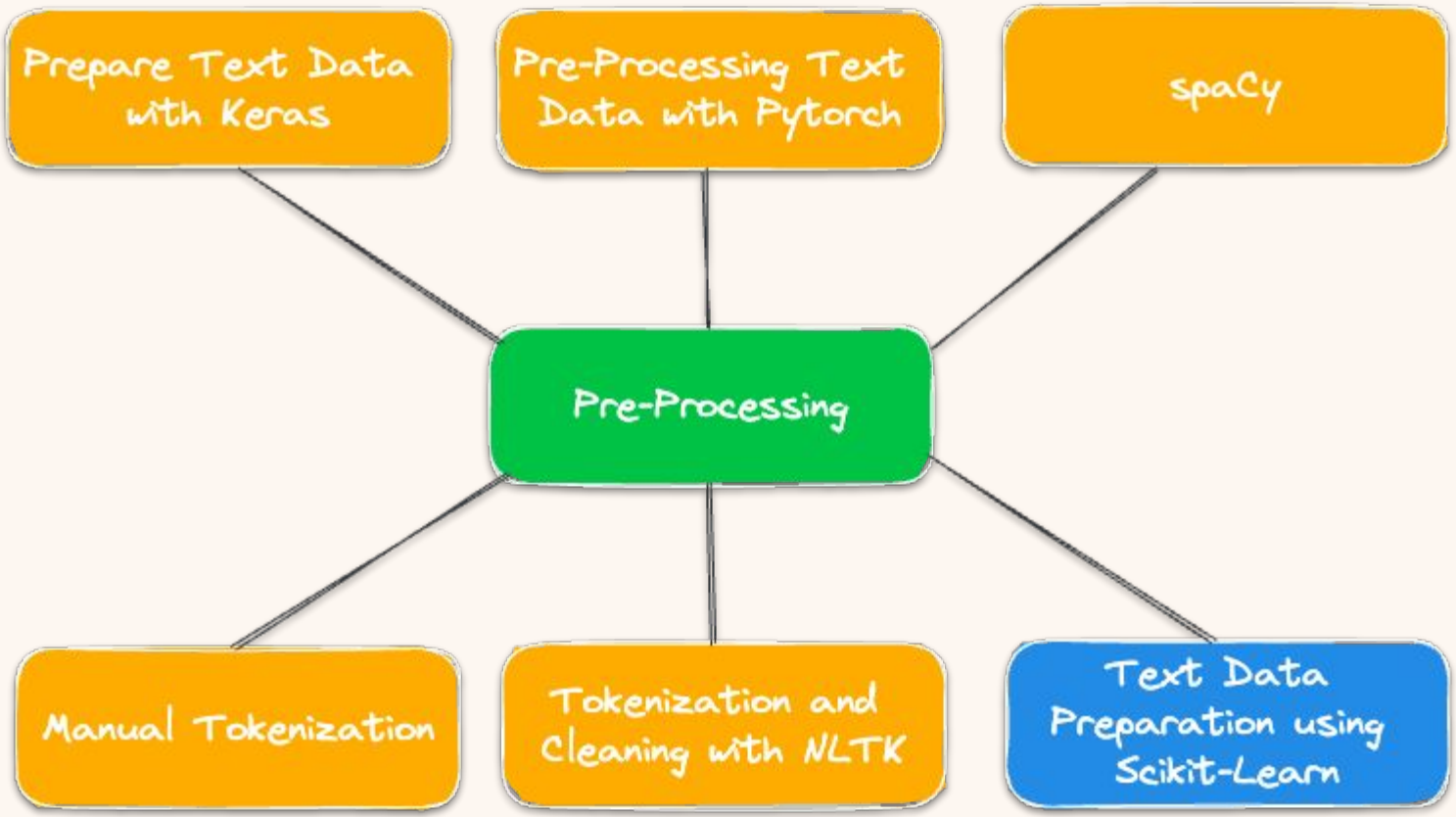
Various stemming techniques exist, with the Porter Stemming algorithm being one of the most renowned and enduring.

Word Tokenizer

```
['One', 'morning', ',', 'when', 'Gregor', 'Samsa', 'woke', 'from', 'troubled', 'dreams', ',', 'he',
'found', 'himself', 'transformed', 'in', 'his', 'bed', 'into', 'a', 'horrible', 'vermin', '.',
'He', 'lay', 'on', 'his', 'armour-like', 'back', ',', 'and', 'if', 'he', 'lifted', 'his', 'head', 'a',
'little', 'he', 'could', 'see', 'his', 'brown', 'belly', ',', 'slightly', 'domed', 'and',
'divided', 'by', 'arches', 'into', 'stiff', 'sections', '.', 'The', 'bedding', 'was', 'hardly',
'able', 'to', 'cover', 'it', 'and', 'seemed', 'ready', 'to', 'slide', 'off', 'any', 'moment', '.',
'His', 'many', 'legs', ',', 'pitifully', 'thin', 'compared', 'with', 'the', 'size', 'of', 'the',
'rest', 'of', 'him', ',', 'waved', 'about', 'helplessly', 'as', 'he', 'looked', '.', '“',
'What', '’', 's', 'happened']
```

Stemming

```
['one', 'morn', ',', 'when', 'gregor', 'samsa', 'woke', 'from', 'troubl', 'dream', ',', 'he', 'found', 'himself',
 'transform', 'in', 'hi', 'bed', 'into', 'a', 'horribl', 'vermin', '.', 'he', 'lay', 'on', 'hi', 'armour-lik',
'back', ',', 'and', 'if', 'he', 'lift', 'hi', 'head', 'a', 'littl', 'he','could', 'see', 'hi', 'brown', 'belli',
',', 'slightli', 'dome', 'and', 'divid', 'by', 'arch', 'into', 'stiff', 'section', '.', 'the', 'bed', 'wa', 'hardli',
'abl', 'to', 'cover', 'it', 'and', 'seem', 'readi', 'to', 'slide', 'off', 'ani', 'moment', '.', 'hi', 'mani', 'leg',
',', 'piti', 'thin', 'compar', 'with', 'the', 'size', 'of', 'the', 'rest', 'of', 'him', ',', 'wave', 'about',
'helplessli', 'as', 'he', 'look', '.', '“', 'what', '’', 's', 'happen']
```


Prepare Text Data with Keras
Pre-Processing Text Data with Pytorch
spaCy
Pre-Processing
Manual Tokenization
Tokenization and Cleaning with NLTK
Text Data Preparation using Scikit-Learn

Install User Guide API Examples Community More

Go

scikit-learn

Machine Learning in Python

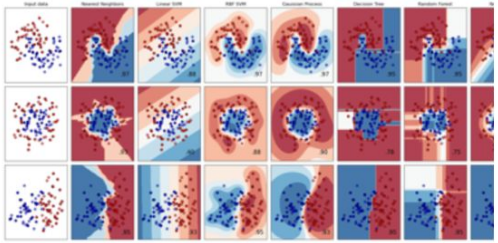
Getting Started | Release Highlights for 1.3 | GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: Gradient boosting, nearest neighbors, random forest, logistic regression, and more...

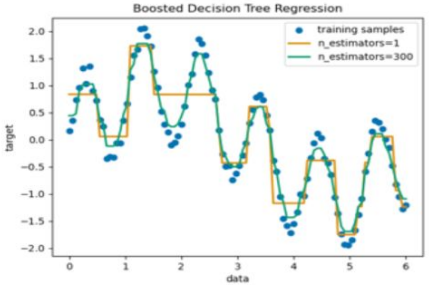


Examples

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: Gradient boosting, nearest neighbors, random forest, ridge, and more...

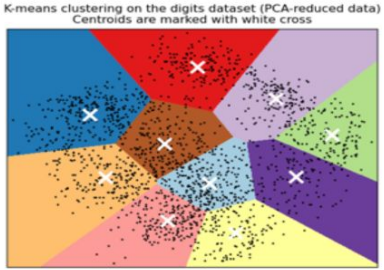


Examples

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, HDBSCAN, hierarchical clustering, and more...



Examples

Text Data Preparation Using Scikit-learn

Textual data necessitates unique pre-processing steps to make it suitable for predictive modeling.

Initially, it involves parsing the text to separate and identify individual words, a process known as **tokenization**.

- Transforming text into word count vectors using CountVectorizer.
- Transitioning text into word frequency vectors via TfidfVectorizer.

Text Data Preparation Using Scikit-learn

```python
from sklearn.feature_extraction.text import CountVectorizer

# list of text documents
text = ["The quick brown fox jumped over the lazy dog."]

# create the transform
vectorizer = CountVectorizer()

# tokenize and build vocab
vectorizer.fit(text)

# summarize
print(vectorizer.vocabulary_)

# encode document
vector = vectorizer.transform(text)

# summarize encoded vector
print(vector.shape)
print(type(vector))
print(vector.toarray())
```

```python
# encode another document
text2 = ["the puppy"]
vector = vectorizer.transform(text2)
print(vector.toarray())
```

```
[[0 0 0 0 0 0 0 1]]
```

```
{'the': 7, 'quick': 6, 'brown': 0, 'fox': 2,
'jumped': 3, 'over': 5, 'lazy': 4, 'dog': 1}

(1, 8)

<class 'scipy.sparse._csr.csr_matrix'>

[[1 1 1 1 1 1 1 2]]
```

Text Data Preparation Using Scikit-learn (Cont.)

Mathematical Breakdown of TF-IDF

1. Term Frequency (TF)

The term frequency for a term `t` in a document `d` is defined as:

$$\mathrm{TF}(t, d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d}$$

2. Inverse Document Frequency (IDF)

The inverse document frequency for a term `t` is defined as:

$$\mathrm{IDF}(t) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing term } t}\right) + 1$$

3. TF-IDF: The TF-IDF score is the product of TF and IDF.

$$TFIDF(t, d) = TF(t, d) \times IDF(t)$$

In the realm of text data analysis, word counts offer a rudimentary technique for representing textual information. However, a salient challenge with relying solely on raw word counts is that commonplace words—often referred to as stop words like the—typically emerge frequently.