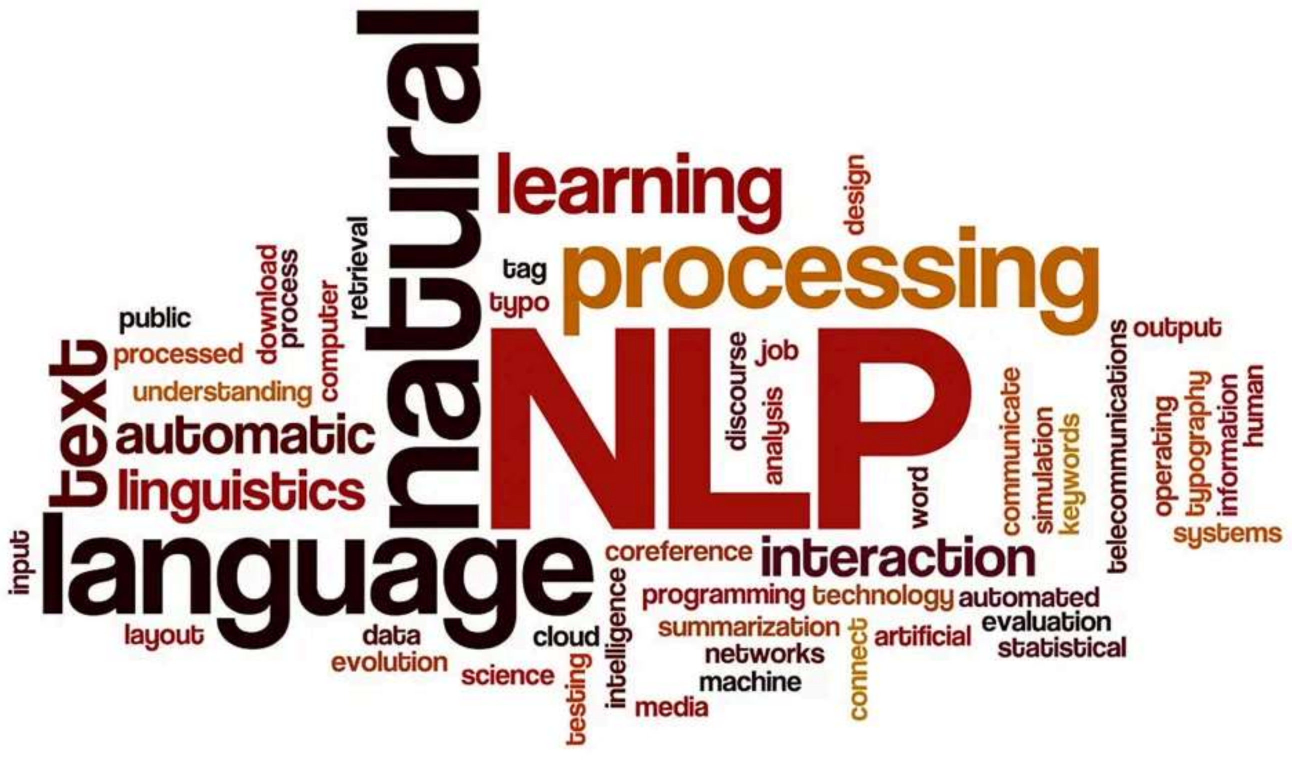
Preprocessing Steps for Natural Language Processing (NLP): A Beginner's Guide

Maleesha De Silva · Follow
5 min read · Apr 30, 2023

Listen Share More



Machine Learning heavily relies on the quality of the data fed into it, and thus, data preprocessing plays a crucial role in ensuring the accuracy and efficiency of the model. In this article, we will discuss the main text preprocessing techniques used in NLP.

1. Text Cleaning

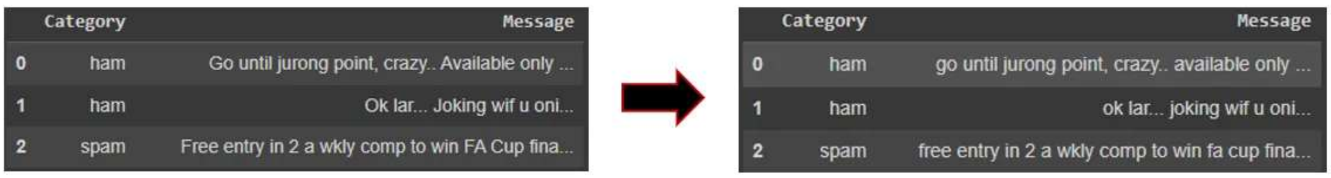
In this step, we will perform fundamental actions to clean the text. These actions involve transforming all the text to lowercase, eliminating characters that do not qualify as words or whitespace, as well as removing any numerical digits present.

I. Converting to lowercase

Python is a case sensitive programming language. Therefore, to avoid any issues and ensure consistency in the processing of the text, we convert all the text to lowercase.

This way, "Free" and "free" will be treated as the same word, and our data analysis will be more accurate and reliable.

```
df = df.applymap(lambda x: x.lower() if isinstance(x, str) else x)
```



	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...

	Category	Message
0	ham	go until jurong point, crazy.. available only ...
1	ham	ok lar... joking wif u oni...
2	spam	free entry in 2 a wkly comp to win fa cup fina...

Dataset before and after converting all text into lowercase

II. Removing URLs

When building a model, URLs are typically not relevant and can be removed from the text data.

For removing URLs we can use 'regex' library.

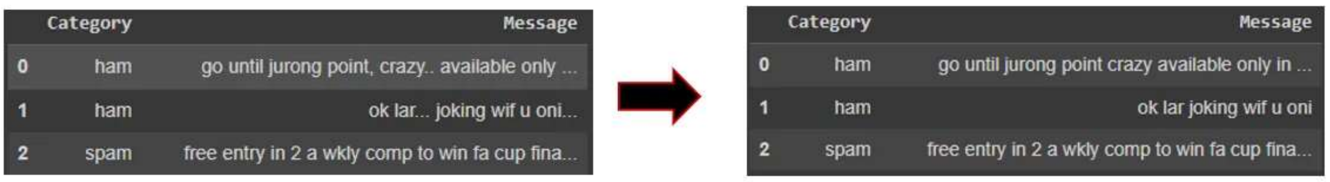
```
import pandas as pd
import re

# Define a regex pattern to match URLs
url_pattern = re.compile(r'https?://\S+')

# Define a function to remove URLs from text
def remove_urls(text):
    return url_pattern.sub('', text)
    # Apply the function to the 'text' column and create a new column 'clean_text'
    df['Message'] = df['Message'].apply(remove_urls)
```

III. Removing remove non-word and non-whitespace characters

It is essential to remove any characters that are not considered as words or whitespace from the text dataset.

These non-word and non-whitespace characters can include punctuation marks, symbols, and other special characters that do not provide any meaningful information for our analysis.

```
df = df.replace(to_replace=r'[^\w\s]', value='', regex=True)
```



	Category	Message
0	ham	go until jurong point, crazy.. available only ...
1	ham	ok lar... joking wif u oni...
2	spam	free entry in 2 a wkly comp to win fa cup fina...

	Category	Message
0	ham	go until jurong point crazy available only in ...
1	ham	ok lar joking wif u oni
2	spam	free entry in 2 a wkly comp to win fa cup fina...

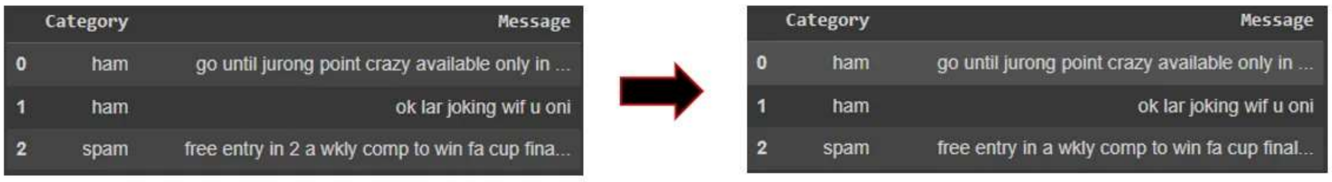
Dataset before and after removing all non-word and non-whitespace characters

IV. Removing digits

It is important to remove all numerical digits from the text dataset. This is because, in most cases, numerical values do not provide any significant meaning to the text analysis process.

Moreover, they can interfere with natural language processing algorithms, which are designed to understand and process text-based information.

```
df = df.replace(to_replace=r'\d', value='', regex=True)
```



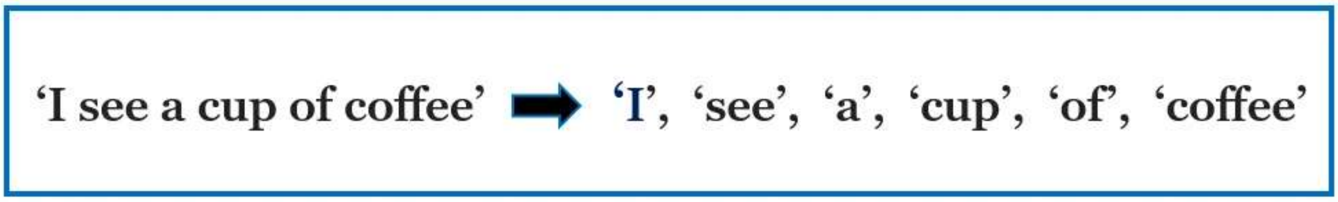
	Category	Message
0	ham	go until jurong point crazy available only in ...
1	ham	ok lar joking wif u oni
2	spam	free entry in 2 a wkly comp to win fa cup fina...

	Category	Message
0	ham	go until jurong point crazy available only in ...
1	ham	ok lar joking wif u oni
2	spam	free entry in a wkly comp to win fa cup final...

Dataset before and after removing digits

2. Tokenization

Tokenization is the process of breaking down large blocks of text such as paragraphs and sentences into smaller, more manageable units.

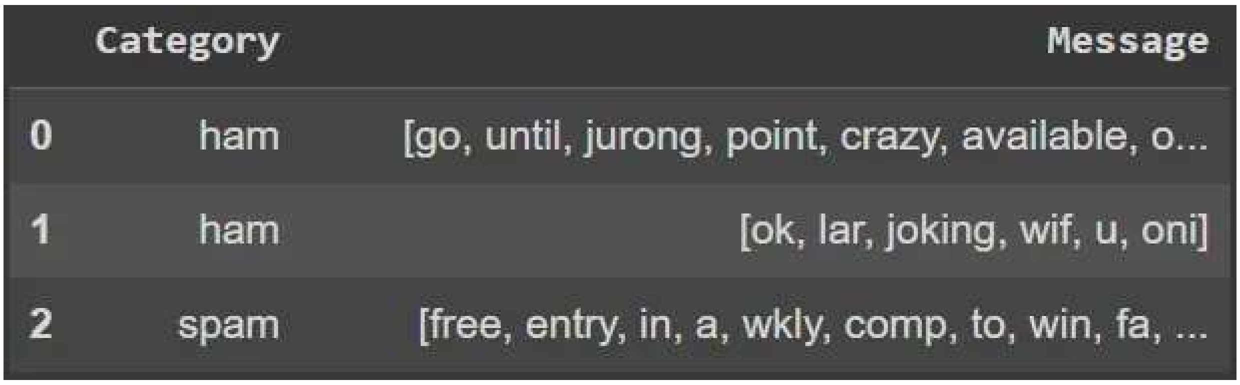


Tokenization of text

In this step, we will be applying word tokenization to split the data in the 'Message' column into words.

By performing word tokenization, we can obtain a more accurate representation of the underlying patterns and trends present in the text data.

```
import nltk
from nltk.tokenize import word_tokenize

df['Message'] = df['Message'].apply(word_tokenize)
```



	Category	Message
0	ham	[go, until, jurong, point, crazy, available, o...
1	ham	[ok, lar, joking, wif, u, oni]
2	spam	[free, entry, in, a, wkly, comp, to, win, fa, ...

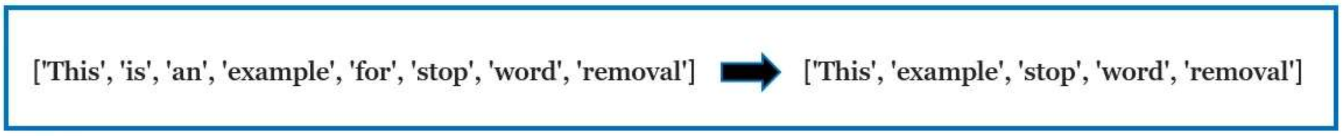
Dataset after tokenization

3. Stopword Removal

Stopwords refer to the most commonly occurring words in any natural language.

For the purpose of analyzing text data and building NLP models, these stopwords might not add much value to the meaning of the document. Therefore, removing stopwords can help us to focus on the most important information in the text and improve the accuracy of our analysis.

One of the advantages of removing stopwords is that it can reduce the size of the dataset, which in turn reduces the training time required for natural language processing models.

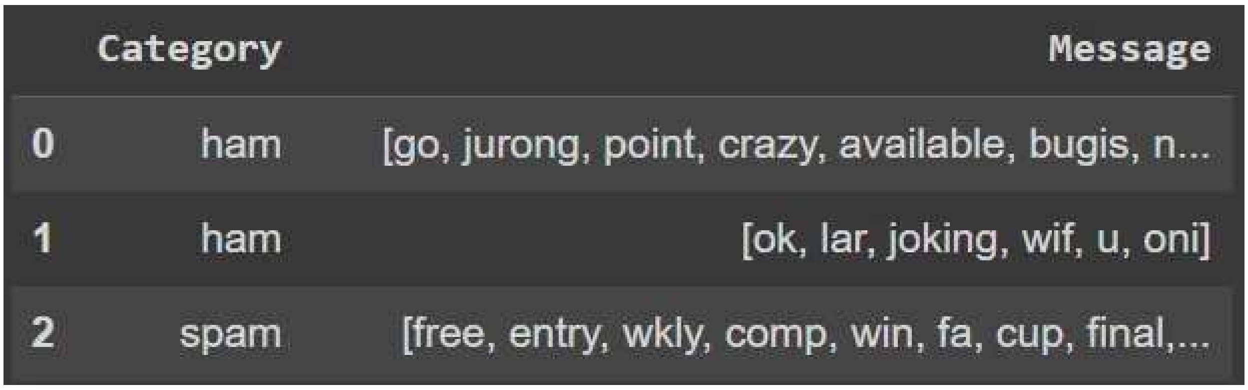

['This', 'is', 'an', 'example', 'for', 'stop', 'word', 'removal'] ➡ ['This', 'example', 'stop', 'word', 'removal']

Stopword removal

Various libraries such as 'Natural Language Toolkit' (NLTK), 'spaCy', and 'Scikit-Learn' can be used to remove stopwords.

In this example, we will use the NLTK library to remove stopwords in the 'Message' column of our dataset.

```
import nltk
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))
df['Message'] = df['Message'].apply(lambda x: [word for word in x if word not i
```

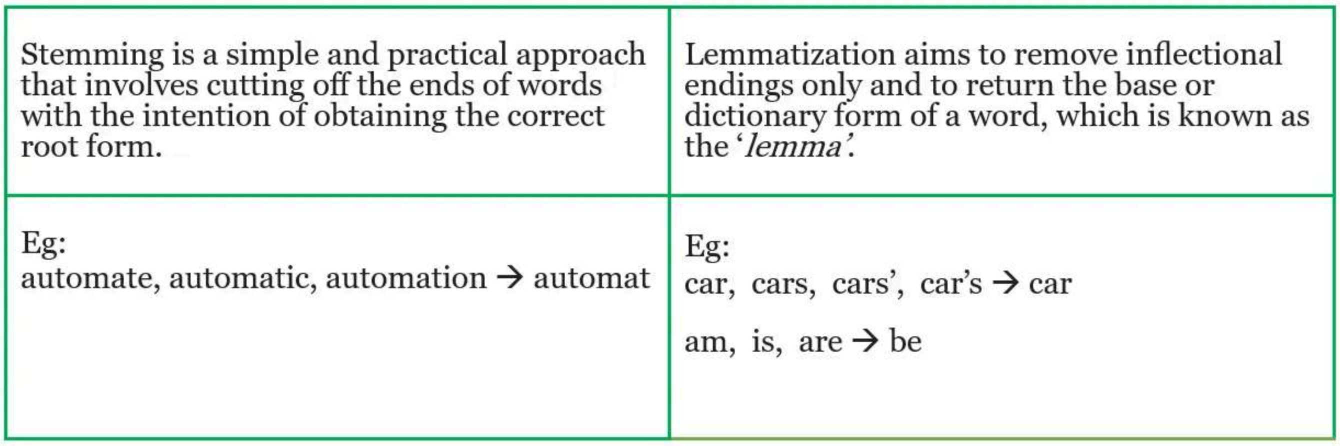


	Category	Message
0	ham	[go, jurong, point, crazy, available, bugis, n...
1	ham	[ok, lar, joking, wif, u, oni]
2	spam	[free, entry, wkly, comp, win, fa, cup, final,...

Dataset after removal of stopwords

4. Stemming/Lemmatization

What's the difference between Stemming and Lemmatization?



Stemming is a simple and practical approach that involves cutting off the ends of words with the intention of obtaining the correct root form.	Lemmatization aims to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the '*lemma*'.
Eg: automate, automatic, automation → automat	Eg: car, cars, cars', car's → car am, is, are → be

Stemming vs Lemmatization

There are various algorithms that can be used for stemming,

· Porter Stemmer algorithm

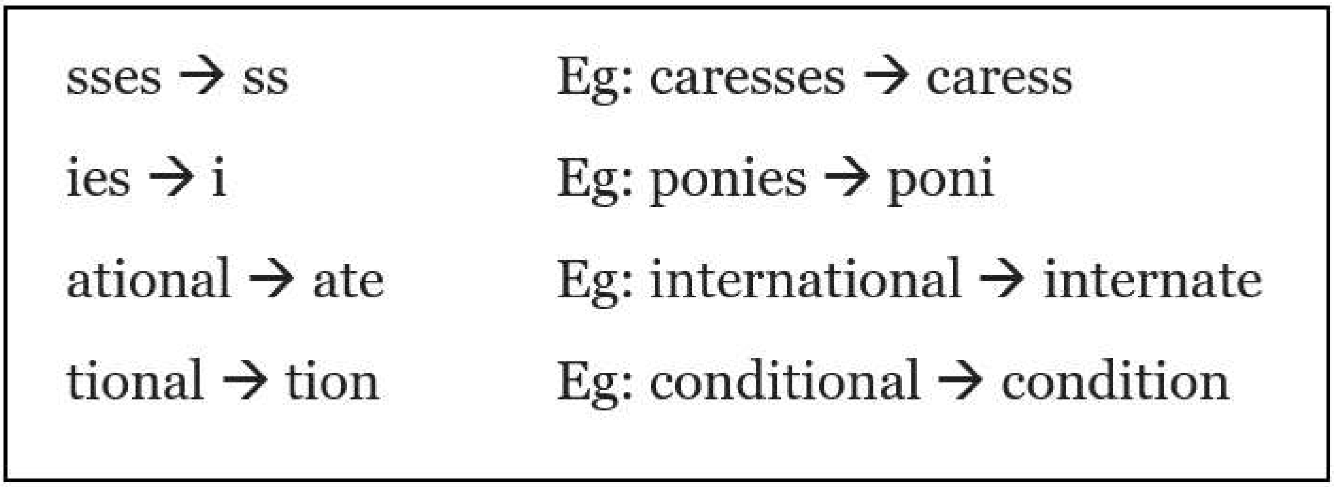
· Snowball Stemmer algorithm

· Lovins Stemmer algorithm

Stemming

Let's take a look at how we can use 'Porter Stemmer' algorithm on our dataset.

Some basic rules defined under the Porter Stemmer algorithm are,



sses → ss	Eg: caresses → caress
ies → i	Eg: ponies → poni
ational → ate	Eg: international → internate
tional → tion	Eg: conditional → condition

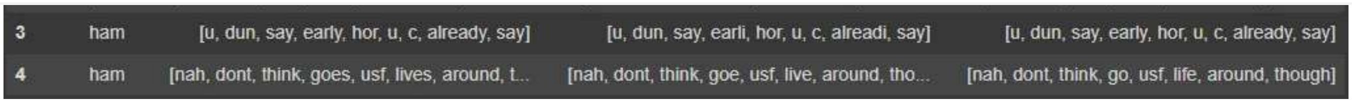
```
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import pandas as pd

# Initialize the Porter Stemmer
stemmer = PorterStemmer()

# Define a function to perform stemming on the 'text' column
def stem_words(words):
    return [stemmer.stem(word) for word in words]

# Define a function to perform stemming on the 'text' column
def stem_words(words):
    return [stemmer.stem(word) for word in words]

# Apply the function to the 'text' column and create a new column 'stemmed_text
df['stemmed_messages'] = df['Message'].apply(stem_words)
```

Lemmatization

Next, let's take a look at how we can implement Lemmatization for the same dataset.

```
import nltk
nltk.download('averaged_perceptron_tagger')
import nltk
nltk.download('wordnet')

import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
import pandas as pd

# initialize lemmatizer
lemmatizer = WordNetLemmatizer()

# define function to lemmatize tokens
def lemmatize_tokens(tokens):
    # convert POS tag to WordNet format
    def get_wordnet_pos(word):
        tag = nltk.pos_tag([word])[0][1][0].upper()
        tag_dict = {"J": wordnet.ADJ,
                    "N": wordnet.NOUN,
                    "V": wordnet.VERB,
                    "R": wordnet.ADV}
        return tag_dict.get(tag, wordnet.NOUN)

    # lemmatize tokens
    lemmas = [lemmatizer.lemmatize(token, get_wordnet_pos(token)) for token in

    # return lemmatized tokens as a list
    return lemmas

# apply lemmatization function to column of dataframe
df['lemmatized_messages'] = df['Message'].apply(lemmatize_tokens)
```

The above code segments will produce outputs as shown below.



3	ham	[u, dun, say, early, hor, u, c, already, say]	[u, dun, say, earli, hor, u, c, alreadi, say]	[u, dun, say, early, hor, u, c, already, say]
4	ham	[nah, dont, think, goes, usf, lives, around, t...	[nah, dont, think, goe, usf, live, around, tho...	[nah, dont, think, go, usf, life, around, though]

Stemmed and Lemmatized dataset

Note that, we only use either Stemming or Lemmatization on our dataset based on the requirement.

Conclusion

In this article we discussed main preprocessing steps in building an NLP model, which include text cleaning, tokenization, stopword removal, and stemming/lemmatization. Implementing these steps can help improve model accuracy by reducing the noise in the text data and converting it into a structured format that can be easily analyzed by the model.

References

Stemming and lemmatization

Next: Faster postings list intersection Up: Determining the vocabulary of Previous: Other languages. Contents Index For...

nlp.stanford.edu

NLP Preprocessing Steps in Easy Way

This article was published as a part of the Data Science Blogathon . Hello friends, In this article, we will discuss...

www.analyticsvidhya.com

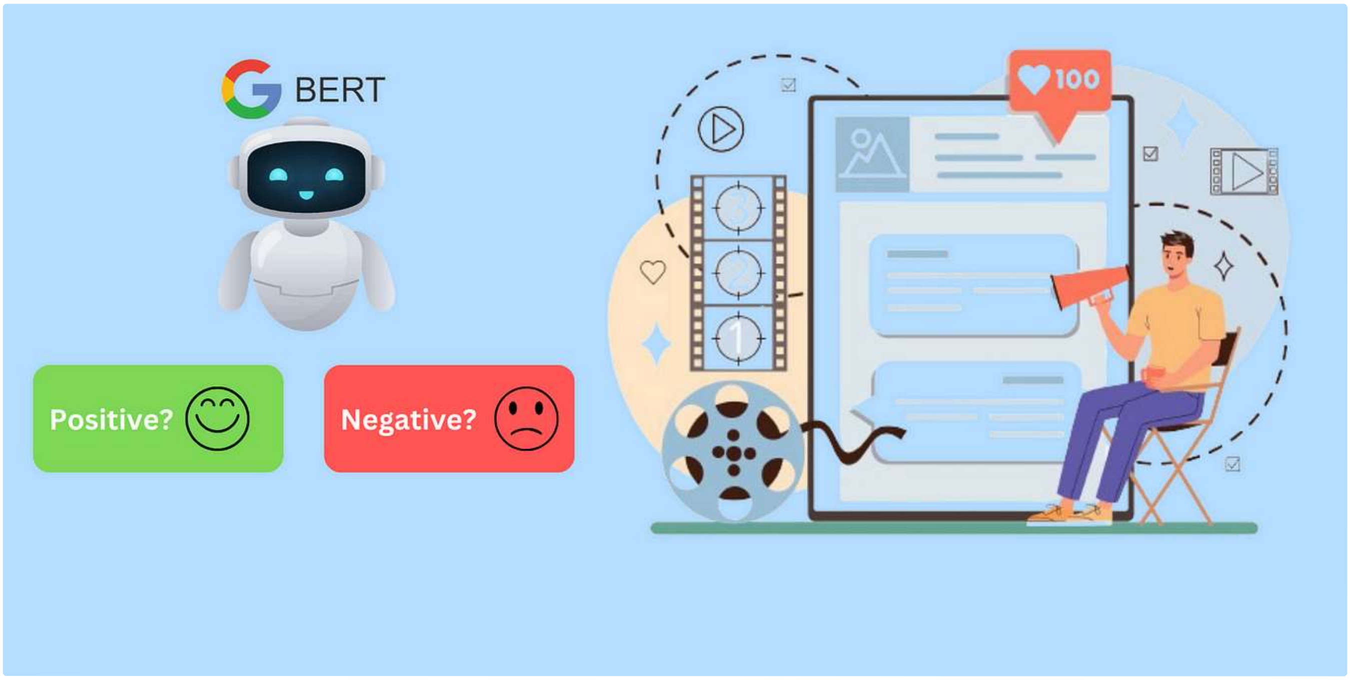
Follow

Written by Maleesha De Silva

35 Followers

Final Year Data Science Undergraduate | Beta Microsoft Learn Student Ambassador

More from Maleesha De Silva



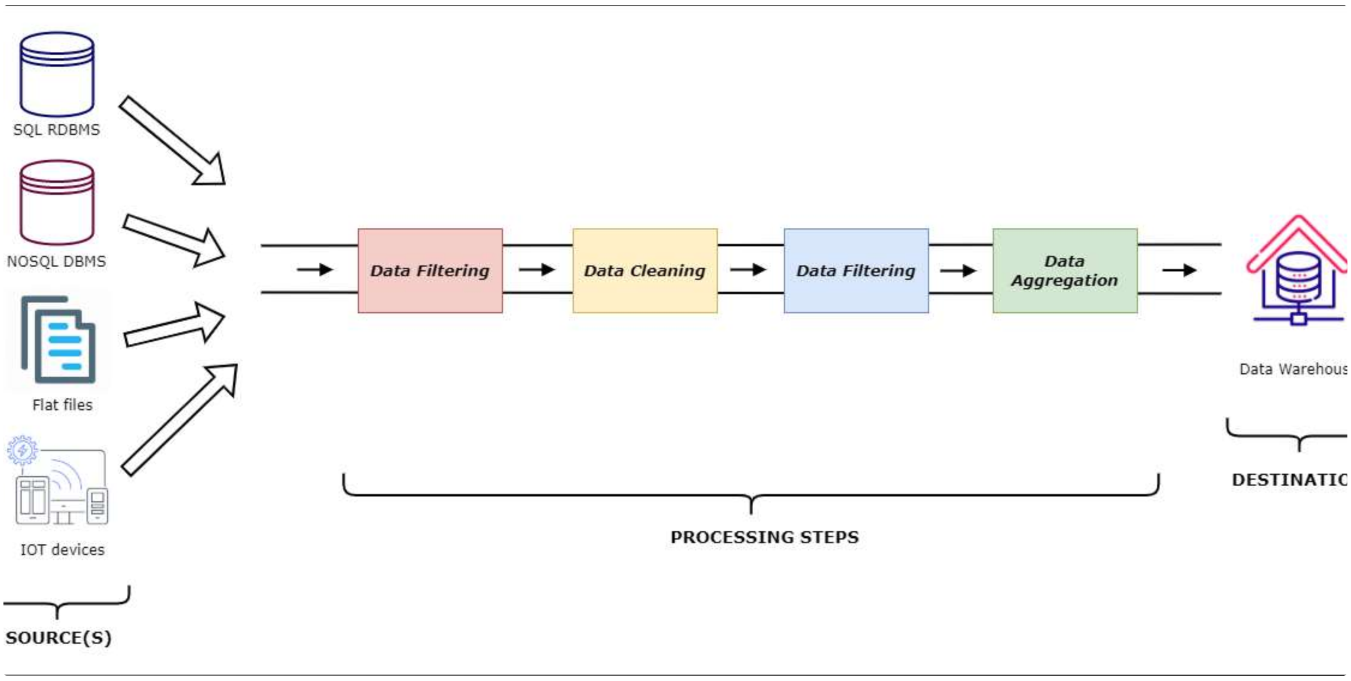
Maleesha De Silva

Text Classification using BERT: A Complete Guide

What is BERT?

6 min read · Feb 22, 2024

14



Maleesha De Silva

An Introduction to Data Pipelines

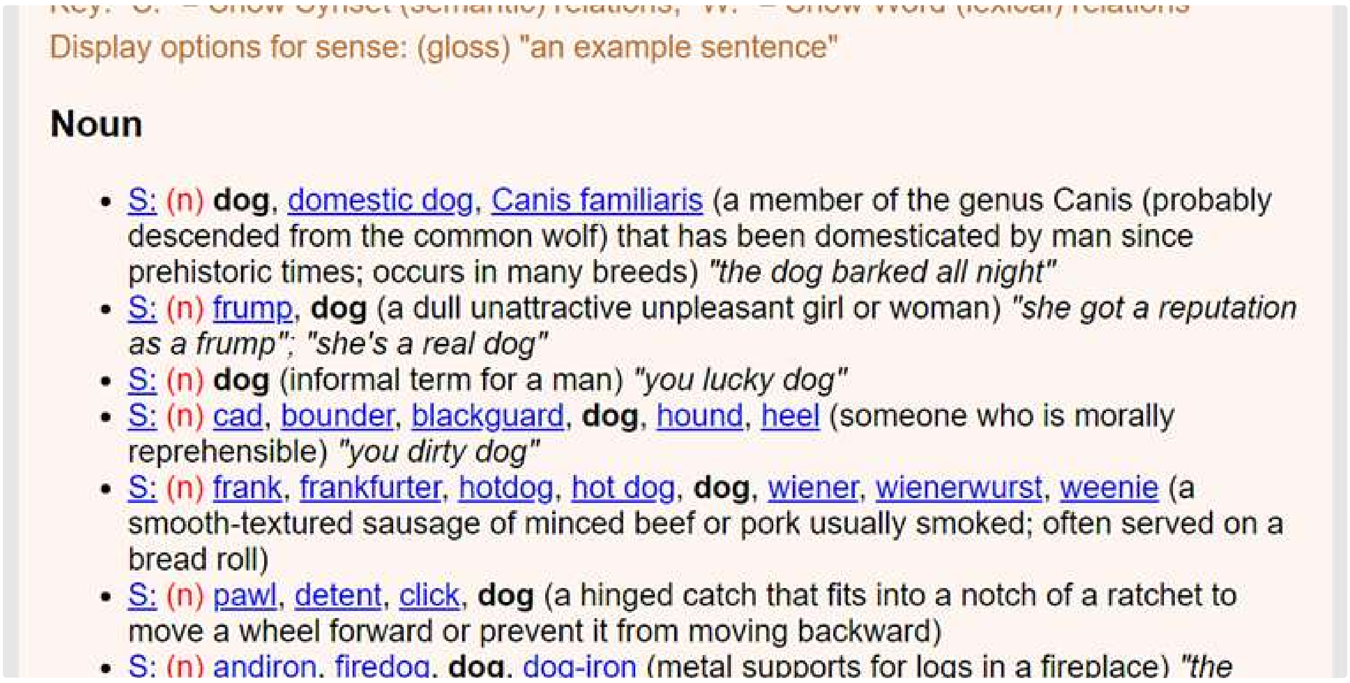
What is a data pipeline?

5 min read · Jan 18, 2023

1

See all from Maleesha De Silva

Recommended from Medium


Display options for sense: (gloss) "an example sentence"

Noun

- S: (n) **dog**, domestic dog, Canis familiaris (a member of the genus Canis (probably descended from the common wolf) that has been domesticated by man since prehistoric times; occurs in many breeds) *"the dog barked all night"*
- S: (n) frump, **dog** (a dull unattractive unpleasant girl or woman) *"she got a reputation as a frump"; "she's a real dog"*
- S: (n) **dog** (informal term for a man) *"you lucky dog"*
- S: (n) cad, bounder, blackguard, **dog**, hound, heel (someone who is morally reprehensible) *"you dirty dog"*
- S: (n) frank, frankfurter, hotdog, hot dog, **dog**, wiener, wienerwurst, weenie (a smooth-textured sausage of minced beef or pork usually smoked; often served on a bread roll)
- S: (n) pawl, detent, click, **dog** (a hinged catch that fits into a notch of a ratchet to move a wheel forward or prevent it from moving backward)
- S: (n) andiron, firedog, **dog**, dog-iron (metal supports for logs in a fireplace) "the

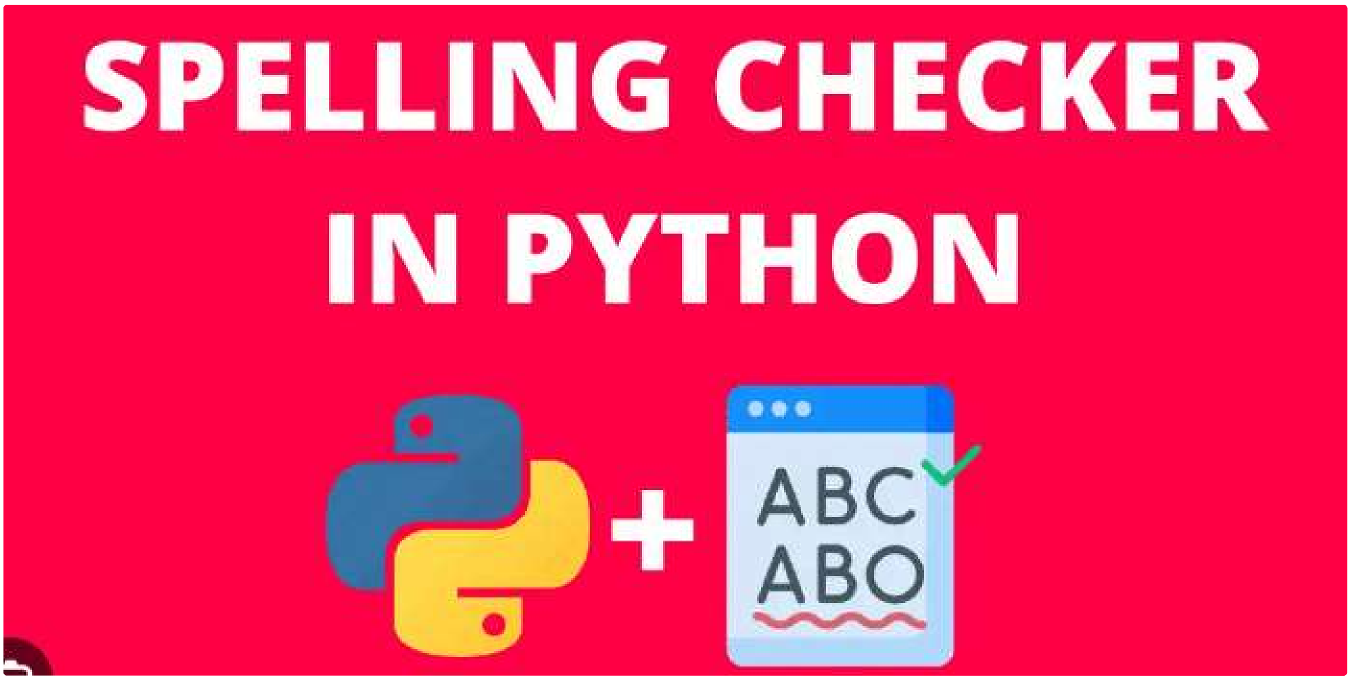
om pramod

WordNet in Action: Enhancing NLP Capabilities Through NLTK

Words might seem distinct at first glance but often share similarities within certain contexts. Consider the words "big" and "large." At a...

2 min read · Dec 14, 2023

609



NitinKumar Sharma

Spell Checker in Python: Create your own spell checker

Spell checking is a crucial component in natural language processing and text analysis. In this blog post, we'll explore a Python...

8 min read · Dec 31, 2023

6

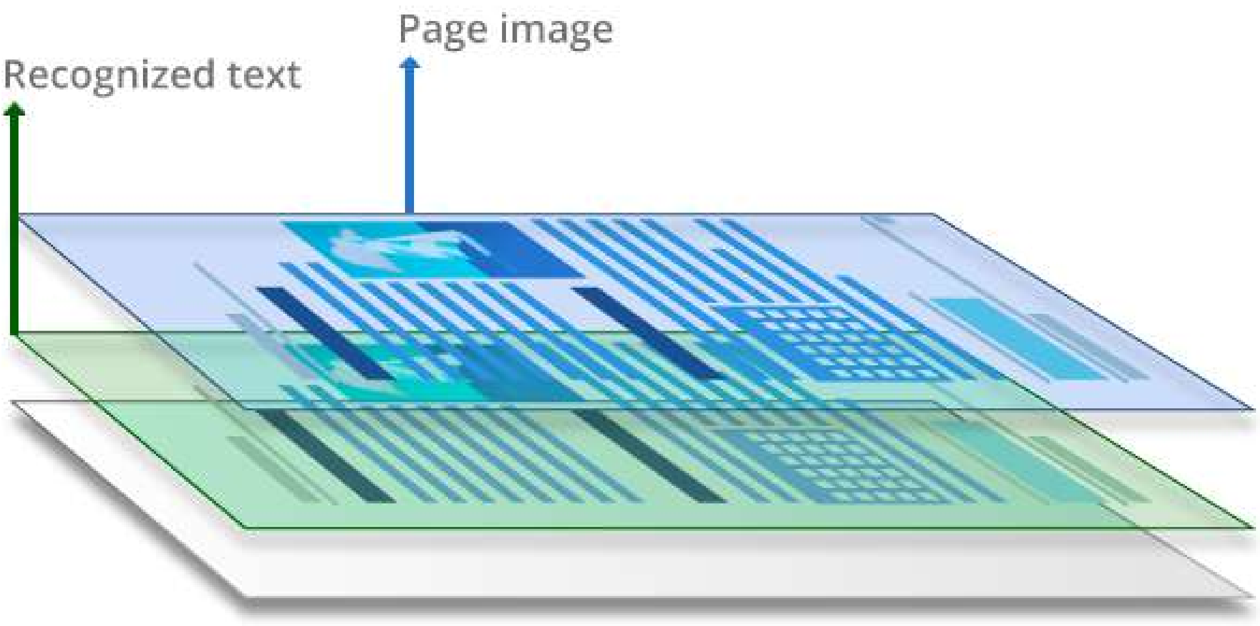
Lists

Staff Picks
636 stories · 953 saves

Stories to Help You Level-Up at Work
19 stories · 599 saves

Self-Improvement 101
20 stories · 1759 saves

Productivity 101
20 stories · 1623 saves



Sasha Korovkina in Dev Genius

Building a High Precision Financial PDF Extraction Tool. Part 1.

Parsing Text from PDF Files

10 min read · Apr 29, 2024

97

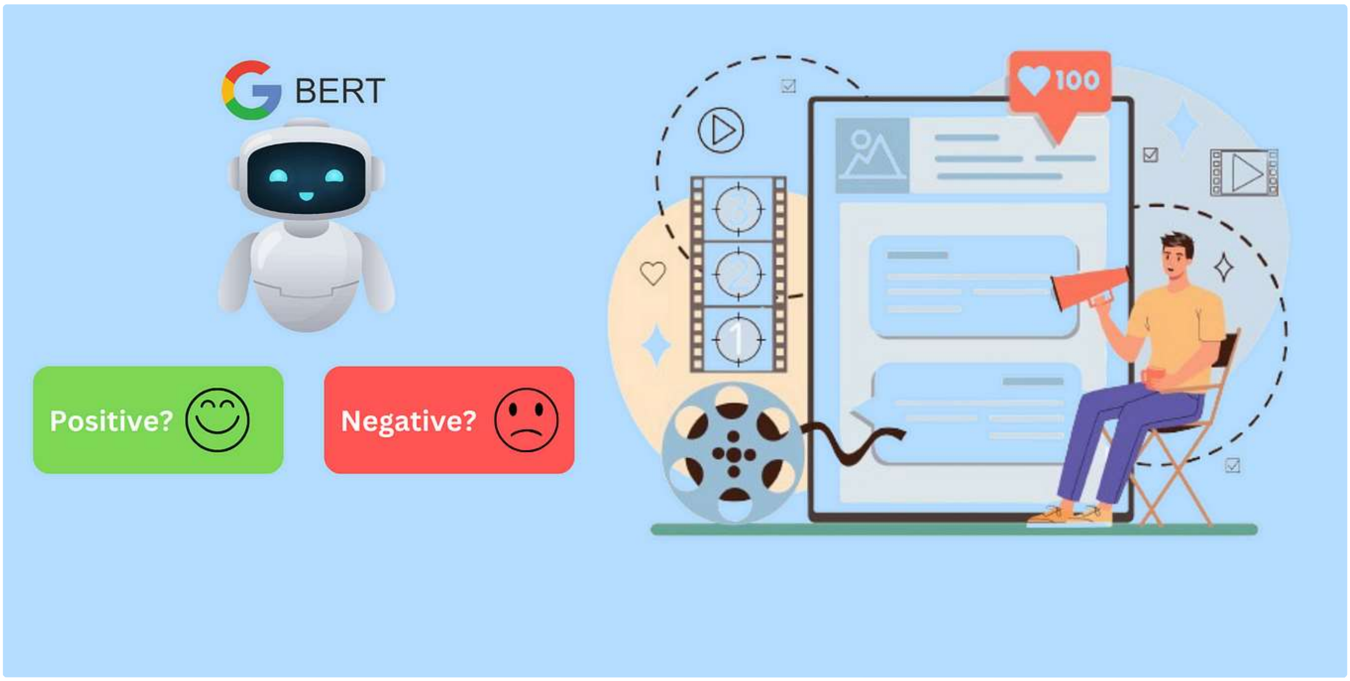
Jason Roell in Stackademic

Ultimate Python Cheat Sheet: Practical Python For Everyday Tasks

(My Other Ultimate Guides)

34 min read · Jan 30, 2024

4.4K 38



Maleesha De Silva

Text Classification using BERT: A Complete Guide

What is BERT?

6 min read · Feb 22, 2024

14

Furkan Ayık

Mastering Text Classification with BERT: A Comprehensive Guide

Understand how to build advanced classifiers with fine-tuning BERT and its variants.

10 min read · Dec 17, 2023

62

See more recommendations