How To Use Text Normalization Techniques In NLP With Python [9 Ways]

by Neri Van Otten | Jan 25, 2023 | Data Science, Natural Language Processing

Text normalization is a key step in natural language processing (NLP). It involves cleaning and preprocessing text data to make it consistent and usable for different NLP tasks. The process includes a variety of techniques, such as case normalization, punctuation removal, stop word removal, stemming, and lemmatization. In this article, we will discuss the different text normalization techniques and give examples, advantages, disadvantages, and sample code in Python.

Table of Contents

Steps to carry out text normalization in NLP

1. Case Normalization

Case normalization is converting all text to lowercase or uppercase to standardize the text. This technique is useful when working with text data that contains a mix of uppercase and lowercase letters.

Example text normalization

Input: “The quick BROWN Fox Jumps OVER the lazy dog.”

Output: “the quick brown fox jumps over the lazy dog.”

Advantages

- It eliminates case sensitivity, making text data consistent and easier to process.
- It reduces the dimensionality of the data, which can improve the performance of NLP algorithms.

Disadvantages

- It can lead to loss of information, as capitalization can indicate proper nouns or emphasis.

Text normalization code in Python

```
text = "The quick BROWN Fox Jumps OVER the lazy dog."
text = text.lower()
print(text)
```

2. Punctuation Removal

Punctuation removal is the process of removing special characters and punctuation marks from the text. This technique is useful when working with text

data containing many punctuation marks, which can make the text harder to process.

Example text normalization

Input: “The quick BROWN Fox Jumps OVER the lazy dog!!!”

Output: “The quick BROWN Fox Jumps OVER the lazy dog”

Advantages

- It removes unnecessary characters, making the text cleaner and easier to process.
- It reduces the dimensionality of the data, which can improve the performance of NLP algorithms.

Disadvantages

- It can lead to loss of information, as punctuation marks can indicate sentiment or emphasis.

Text normalization code in Python

```
import string
text = "The quick BROWN Fox Jumps OVER the lazy dog!!!"
text = text.translate(text.maketrans("", "", string.punctuation))
print(text)
```

3. Stop Word Removal

Stop word removal is the process of removing common words with little meaning, such as “the” and “a”. This technique is useful when working with text data containing many stop words, which can make the text harder to process.

> **See also** Dependency Parsing In NLP Explained & 9 Tools With How To Tutorial

Example text normalization

Input: “The quick BROWN Fox Jumps OVER the lazy dog.”

Output: “quick BROWN Fox Jumps OVER lazy dog.”

Advantages

- It removes unnecessary words, making the text cleaner and easier to process.
- It reduces the dimensionality of the data, which can improve the performance of NLP algorithms.

Disadvantages

- It can lead to loss of information, as stop words can indicate context or sentiment.

Text normalization code in Python

```
from nltk.corpus import stopwords
text = "The quick BROWN Fox Jumps OVER the lazy dog."
stop_words = set(stopwords.words("english"))
words = text.split()
filtered_words = [word for word in words if word not in stop_words]
text = " ".join(filtered_words)
print(text)
```

4. Stemming

Stemming is reducing words to their root form by removing suffixes and prefixes, such as “running” becoming “run”. This method is helpful when working with text data that has many different versions of the same word, which can make the text harder to process.

Example text normalization

Input: “running,runner,ran”

Output: “run,run,run”

Advantages

- It reduces the dimensionality of the data, which can improve the performance of NLP algorithms.
- It makes it easier to identify the core meaning of a word.

Disadvantages

- It can lead to loss of information, as the root form of a word may not always be the correct form.
- It may produce non-existent words.

Text normalization code in Python

```
from nltk.stem import PorterStemmer
stemmer = PorterStemmer()
text = "running,runner,ran"
words = text.split(",")
stemmed_words = [stemmer.stem(word) for word in words]
text = ",".join(stemmed_words)
print(text)
```

5. Lemmatization

Lemmatization is reducing words to their base form by considering the context in which they are used, such as “running” becoming “run”. This technique is similar to stemming, but it is more accurate as it considers the context of the word.

Example text normalization

Input: “running,runner,ran”

Output: “run,runner,run”

Advantages

- It reduces the dimensionality of the data, which can improve the performance of NLP algorithms.
- It makes it easier to identify the core meaning of a word while preserving context.

Disadvantages

- It can be more computationally expensive than stemming.
- It may not be able to handle all words or forms.

> See also Multilayer Perceptron Explained And How To Train & Optimise MLPs

Text normalization code in Python

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
text = "running,runner,ran"
words = text.split(",")
lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
text = ",".join(lemmatized_words)
print(text)
```

6. Tokenization

Tokenization is the process of breaking text into individual words or phrases, also known as “tokens”. This technique is useful when working with text data that needs to be analyzed at the word or phrase level, such as in text classification or language translation tasks.

Example text normalization

Input: “The quick BROWN Fox Jumps OVER the lazy dog.”

Output: [“The”, “quick”, “BROWN”, “Fox”, “Jumps”, “OVER”, “the”, “lazy”, “dog.”]

Advantages

- It allows for analysing and manipulating individual words or phrases in the text data.
- It can improve the performance of NLP algorithms that rely on word or phrase-level analysis.

Disadvantages

- It can lead to the loss of information, as the meaning of a sentence or text can change based on the context of words.
- It may not be able to handle all forms of text.

Text normalization code in Python

```
from nltk.tokenize import word_tokenize
text = "The quick BROWN Fox Jumps OVER the lazy dog."
tokens = word_tokenize(text)
print(tokens)
```

7. Replacing synonyms and Abbreviation to their full form to normalize the text in NLP

This technique is useful when working with text data that contains synonyms or abbreviations that need to be replaced by their full form.

Example text normalization

Input: “I’ll be there at 2pm”

Output: “I will be there at 2pm”

Advantages

- It makes text data more readable and understandable.
- It can improve the performance of NLP algorithms that rely on word or phrase-level analysis.

Disadvantages

- It can lead to the loss of information, as the meaning of a sentence or text can change based on the context of words.
- It may not be able to handle all forms of text.

Text normalization code in Python

```
text = "I'll be there at 2pm"
synonyms = {"I'll": "I will", "2pm": "2 pm"}
for key, value in synonyms.items():
    text = text.replace(key, value)
print(text)
```

8. Removing numbers and symbol to normalize the text in NLP

This technique is useful when working with text data that contain numbers and symbols that are not important for the NLP task.

Example text normalization

Input: “I have 2 apples and 1 orange #fruits”

Output: “I have apples and orange fruits”

> **See also** How To Implement Intent Classification In NLP [7 ML & DL Models] With Python Example

Advantages

- It removes unnecessary numbers and symbols, making the text cleaner and easier to process.
- It reduces the dimensionality of the data, which can improve the performance of NLP algorithms.

Disadvantages

- It can lead to loss of information, as numbers and symbols can indicate quantities or sentiments.

Text normalization code in Python

```
import re
text = "I have 2 apples and 1 orange #fruits"
text = re.sub(r"[\d#]", "", text)
print(text)
```

9. Removing any remaining non-textual elements to normalize the text in NLP

Removing any remaining non-textual elements such as HTML tags, URLs, and email addresses This technique is useful when working with text data that contains non-textual elements such as HTML tags, URLs, and email addresses that are not important for the NLP task.

Example text normalization

Input: “Please visit example.com for more information or contact me at info@example.com”

Output: “Please visit for more information or contact me at “

Advantages

- It removes unnecessary non-textual elements, making the text cleaner and easier to process.
- It reduces the dimensionality of the data, which can improve the performance of NLP algorithms.

Disadvantages

- It can lead to loss of information, as non-textual elements can indicate context or sentiment.

Text normalization code in Python

```
import re
text = "Please visit <a href='www.example.com'>example.com</a> for more i
text = re.sub(r"(<[^>]+>)|(http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\)
print(text)
```

It’s important to note that these steps should be applied depending on the specific requirements of the NLP task and the type of text data being processed.

Text normalization is an iterative process, and the steps may be repeated multiple times.

Keyword normalization techniques in NLP

Keyword normalization techniques in NLP are used to standardize and clean keywords or phrases in text data, in order to make them more usable for natural language processing tasks.

Keyword normalisation makes them more useful for further analysis.

The above steps for normalising text in NLP can also all be used on a list of keywords or phrases. They be used to make keywords and phrases more consistent, more easily searchable, and more usable for natural language processing tasks such as text classification, information retrieval, and natural language understanding.

Conclusion for text normalization in NLP

Text normalization techniques are essential for preparing text data for natural language processing (NLP) tasks. Each technique has its advantages and disadvantages, and the appropriate technique depends on the specific requirements of the NLP task and the type of text data being processed.

It is also important to note that text normalization is an iterative process, and the steps may be repeated multiple times depending on the requirements of the NLP task.

About the Author

Neri Van Otten

Neri Van Otten is the founder of Spot Intelligence, a machine learning engineer with over 12 years of experience specialising in Natural Language Processing (NLP) and deep learning innovation. Dedicated to making your projects succeed.

Meet Neri

Neri Van Otten is a machine learning and software engineer with over 12 years of Natural Language Processing (NLP) experience. Dedicated to making your projects succeed.

Popular posts

Top 7 Ways of Implementing Document & Text Similarity
Top 10 Open Source Large Language Models (LLM)
Top 8 Most Useful Anomaly Detection Algorithms For Time Series
Self-attention Made Easy And How To Implement It
Top 3 Easy Ways To Remove Stop Word In Python

Search Posts... Search

Connect with us

Join the NLP Community

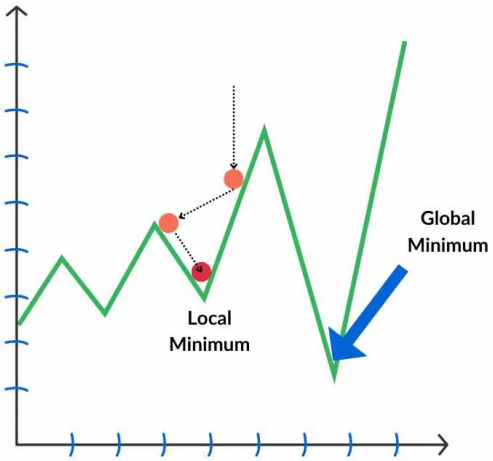
Name

Email

STAY UPDATED WITH OUR NEWSLETTER

Table of Contents

Recent Articles



Stochastic Gradient Descent (SGD) In Machine Learning Explained & How To Implement

Understanding Stochastic Gradient Descent (SGD) In Machine Learning Stochastic Gradient Descent (SGD) is a pivotal optimization algorithm widely utilized in machine...

The BERT [Understa (LLMs) W

What is BERT ir (NLP), the ques human languag

0 Comments

Stay up to date with the latest NLP news

Name

Email

JOIN OUR WEEKLY NEWSLETTER

Building the future by creating innovative products, processing large volumes of text and extracting insights through the use of natural language processing (NLP)

CONNECT WITH US

CONTACT

contact@spotintelligence.com

86-90 Paul Street
EC2A 4NE London
United Kingdom

AWARDS

QUICK LINKS

Home
Extract
API & Applications
NLP Consulting
About
Contact

RESOURCES

Blog
NLP
Machine Learning
Deep Learning