Automate your Text Processing workflow in a single line of Python Code

Process your text data for NLP tasks using the CleanText library

Satyam Kumar May 5 · 4 min read ★

Photo by Windows on Unsplash

that can understand and implement natural language task usage. A typical NLP project follows various aspects of the pipeline to train a model. Various steps in the pipeline include text cleaning, tokenization, stemming, encoding to numerical vector, etc followed by model training.

The dataset derived for NLP tasks is textual data, mainly derived from the internet. Most of the time, the textual data used for NLP modeling is dirty and needs to be cleaned in the early stage of data processing. A data scientist spends most of the time in data preprocessing which includes cleaning the textual data.

In this article, we will discuss an interesting library CleanText, that eases the process of cleaning textual data and speeds up the data preprocessing pipeline.

What is CleanText?

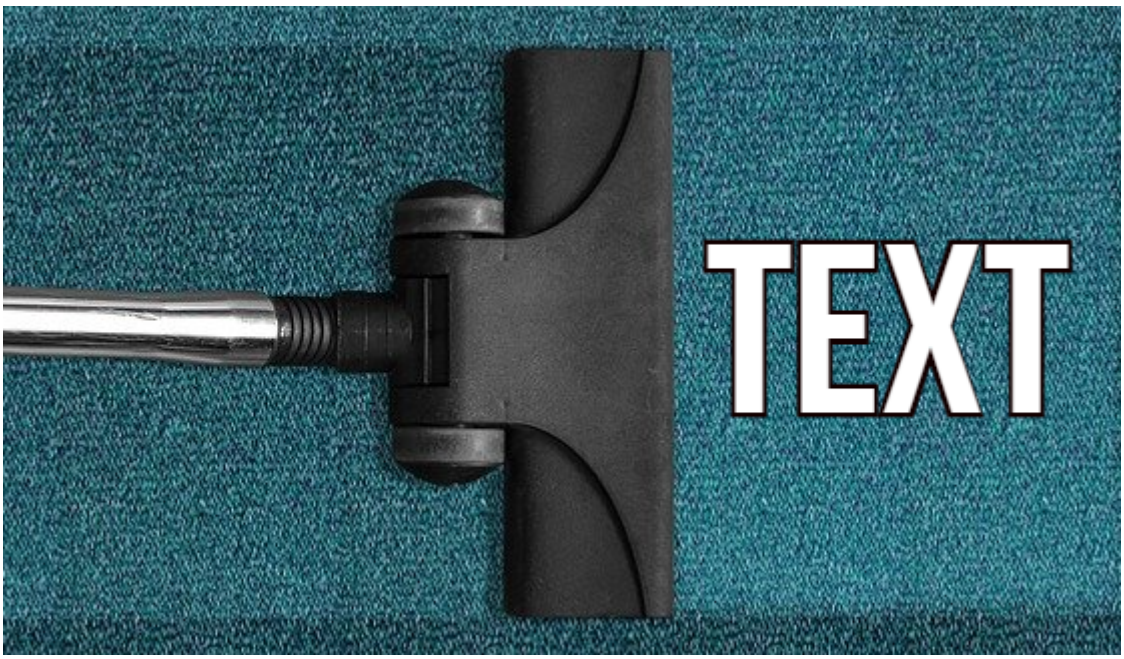

Image by Michal Jarmoluk from Pixabay

CleanText is an open-source Python library that enables to clean of the textual data scraped from the web or social media. CleanText enables developers to create a normalized text representation. CleanText uses ftfy, unidecode, and various other hard-coded rules including RegEx to convert a corrupted or dirty input text into a clean text, that can be further processed to train an NLP model.

Installation:

```
pip install clean-text
```

Post-installation, you can import the library by importing it using:

```
from cleantext import clean
```

Usage:

The library CleanText comes up with just one function 'Clean' that takes various parameters that can be tuned to perform cleaning of text. The clean function can perform 11 types of cleaning including:

Unicode:

It fixes various Unicode errors.

```
s1 = 'Zürich'
clean(s1, fix_unicode=True)

# Output: zurich
```

ASCII:

It translated the text to the nearest ASCII representation.

```
s2 = "ko\u017eu\u0161\u010dek"
clean(s2, to_ascii=True)

# Output: kozuscek
```

Lower:

Convert the text data into lower case.

```
s3 = "My Name is SATYAM"
clean(s3, lower=True)
```

Replace URLs / Emails / Phone Numbers:

Replaces all the URLs or Emails or Phone numbers present in the text data with a special token.

```
s4 = "https://www.Google.com and https://www.Bing.com are popular
seach engines. You can mail me at satkr7@gmail.com. If not replied
call me at 9876543210"

clean(s4, no_urls=True, replace_with_url="URL",
no_emails=True, replace_with_email="EMAIL"
no_phone_numbers=True, replace_with_email="PHONE")

# Output: url and url are popular search engines. You can mail me at
EMAIL. If not replied call me at PHONE
```

Replace Currency:

Replaces all the currency present in the text data with a special token.

```
s5 = "I want ₹ 40"
clean(s5, no_currency_symbols = True)
clean(s5, no_currency_symbols = True,
replace_with_currency_symbol="Rupees")

# Output: i want <cur> 40
# Output: i want rupees 40
```

Remove Numbers:

Replacing or removing all numbers with a special token.

```
s7 = 'abc123def456ghi789zero0'

clean(s7, no_digits = True)
clean(s7, no_digits = True, replace_with_digit="")

# Output: abc000def000ghi000zero0
# Output: abcdefghizero
```

Replace Punctuations:

```
s6 = "40,000 is greater than 30,000."
clean(s6, no_punct = True)

# Output: 40000 is greater than 30000
```

Combining all the parameters:

We have discussed all the parameters individually above. Now let's combine all of them in the Clean function, call it for a dirty sample text and observe the clean text result.

```
from cleantext import clean

text = """
Zürich has a famous website https://www.zuerich.com/
WHICH ACCEPTS 40,000 € and adding a random string, :
abc123def456ghi789zero0 for this demo. Also remove punctions ,.
my phone number is 9876543210 and mail me at satkr7@gmail.com.'
     """

clean_text = clean(s8,
      fix_unicode=True,
      to_ascii=True,
      lower=True,
      no_line_breaks=True,
      no_urls=True,
      no_numbers=True,
      no_digits=True,
      no_currency_symbols=True,
      no_punct=True,
      replace_with_punct="",
      replace_with_url="<URL>",
      replace_with_number="<NUMBER>",
      replace_with_digit="",
      replace_with_currency_symbol="<CUR>",
      lang='en')

print(clean_text)

# Output: zurich has a famous website <url> which accepts <number> <cur> and adding a r
```

CleanText.py hosted with ❤️ by GitHub view raw

(Code by Author), CleanText

Conclusion:

CleanText is an efficient library that can process or clean your scraped dirty data to get a normalized clean text output just in a single line of code. The developer just needs to tune the parameters as per his/her needs. It eases data scientist's work, as now he/she doesn't have to write many lines of a complex regular expression code to clean the text. CleanText not only works with English language input text but can handle German, just by setting `lang='de'` .

CleanText library does only covers some of the text cleaning parameters and has room for improvement. Still, the developer can use it for some cleaning tasks and then proceed with manual coding to complete the remaining.

> *Read the below-mentioned article to know about AutoNLP — an automated NLP library.*

AutoNLP: Sentiment Analysis in 5 Lines of Python Code
AutoNLP — AutoML of Natural Language Processing
medium.com

References:

[1] Clean-Text Repository: https://github.com/jfilter/clean-text

Thank You for Reading