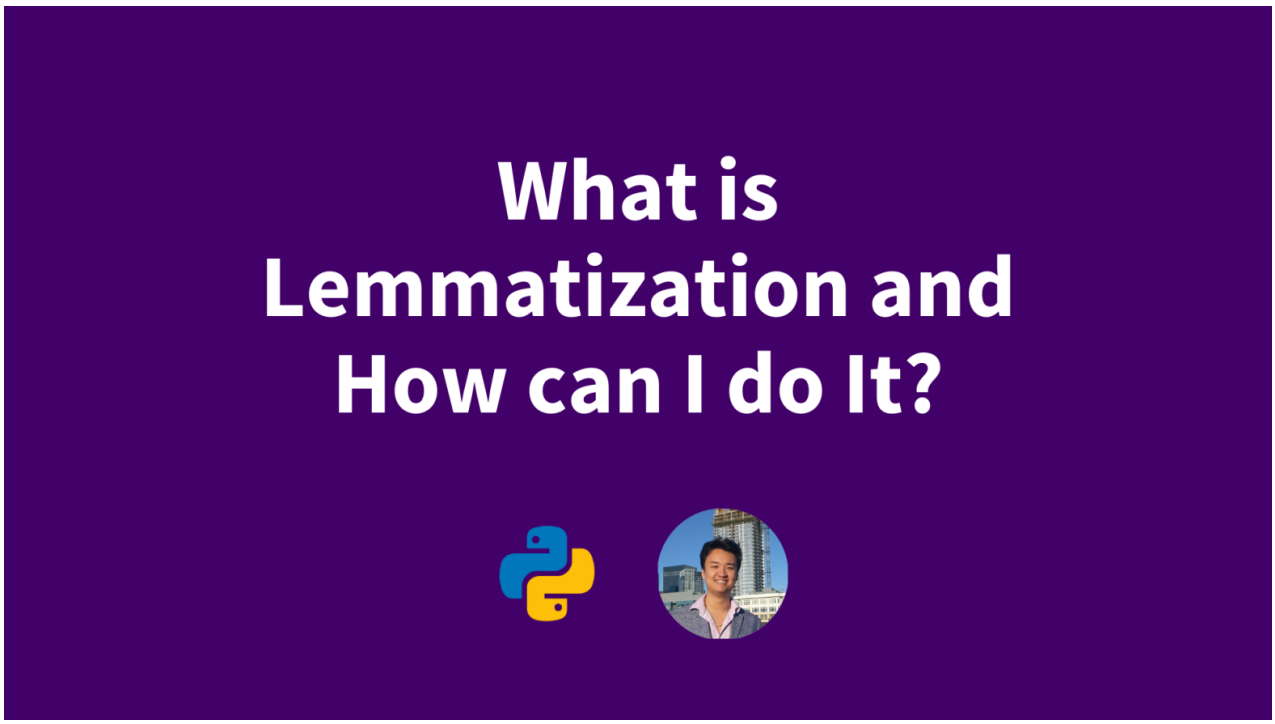
PythonAlgos
The Best and Fastest Way to Learn Python

What is Lemmatization and How can I do It?



Lemmatization is an important part of Natural Language Processing. Other NLP topics we've covered include **Text Polarity**, **Named Entity Recognition**, and **Summarization**. Lemmatization is the process of turning a word into its lemma. A lemma is the "**canonical form**" of a word. A lemma is usually the dictionary version of a word, it's picked by convention. Let's look at some examples to make more sense of this.

The words "playing", "played", and "plays" all have the same lemma of the word "play". The words "win", "winning", "won", and "wins" all have the same lemma of the word "win". Let's take a look at one more example before we move on to how you can do lemmatization in Python. The words "programming", "programs", "programmed", and "programmatic" all have the same lemma of the word "program". Another way to think about it is to think of the lemma as the "root" of the word.

How Can I Do Lemmatization with Python?

Python has many well known Natural Language Processing libraries, and we're going to make use of two of them to do lemmatization. The first one we'll look at is **spaCy** and the second one we'll use is Natural Language Toolkit (**NLTK**).

Lemmatization with spaCy

This is pretty cool, we're going to lemmatize our text in under 10 lines of code. To get started with spaCy we'll install the `spacy` library and download a model. We can do this in the terminal with the following commands:

```
pip install spacy
python -m spacy download en_core_web_sm
```

To start off our program, we'll import `spacy` and load the language model.

```
import spacy

nlp = spacy.load("en_core_web_sm")
```

Once we have the model, we'll simply make up a text, turn it into a spaCy `Doc`, and that's basically it. To get the lemma of each word, we'll just print out the `lemma_` attribute. Note that printing out the `lemma` attribute will get you a number corresponding to the lemma's representation.

```
text = "This is an example program for showing how lemmatization works in s

doc = nlp(text)

for token in doc:
    print(token.lemma_)
```

Our output should look like the following:

```
$ python lemmatization.py
this
be
an
example
program
for
show
how
lemmatization
work
in
spaCy
.
```

Sounds like a pirate!

Lemmatization with NLTK

Cool, lemmatization with spaCy wasn't that hard, let's check it out with NLTK. For NLTK, we'll need to install the library and install the `wordnet` submodule before we can write the program. We can do that in the terminal with the below commands.

```
pip install NLTK
python
>>> import nltk
>>> nltk.download('wordnet')
>>> exit()
```

Why are we running a Python script in shell and not just downloading `wordnet` at the start of our program? We only need to download it once to be able to use it, so we don't want to put it in a program we'll be running multiple times. As always, we'll start out our program by importing the libraries we need. In this case, we're just going to be importing `nltk` and the `WordNetLemmatizer` object from `nltk.stem`.

```
import nltk
from nltk.stem import WordNetLemmatizer
```

First we'll use `word_tokenize` from `nltk` to tokenize our text. Then we'll loop through the tokenized text and use the lemmatizer to lemmatize each token and print it out.

```
lemmatizer = WordNetLemmatizer()
tokenized = nltk.word_tokenize("This is an example program for showing how
for t in tokenized:
    print(lemmatizer.lemmatize(t))
```

We'll end up with something like the image below.

```
$ python lemmatization.py
This
is
an
example
program
for
showing
how
lemmatization
work
in
NLTK
.
Let
's
play
ball
!
```

As you can see, using NLTK returns a different lemmatization than using spaCy. It doesn't seem to do lemmatization as well. NLTK and spaCy are made for different purposes, so I am usually impartial. However, spaCy definitely wins for built in lemmatization. NLTK can be customized because it's highly used for research purposes, but that's out of the scope for this article. Be on the lookout for an in depth dive though!

To learn more feel free to reach out to me **@yujian_tang** on Twitter, connect with me on **LinkedIn**, and **join our Discord**. Remember to follow the blog to stay updated with cool Python projects and ways to level up your Python skills!

I run this site to help you and others like you find cool projects and practice software skills. If this is helpful for you and you enjoy your ad free site, please help fund this site by donating below! If you can't donate right now, please think of us next time.

Yujian Tang

I started my professional software career interning for IBM in high school after winning ACSL two years in a row. I got into

high school after winning ACSL two years in a row. I got into AI/ML in college where I published a first author paper to IEEE Big Data. After college I worked on the AutoML infrastructure at Amazon before leaving to work in startups. I believe I create the highest quality software content so that's what I'm doing now. Drop a comment to let me know!

Connect with me on LinkedIn!

Leave a Reply

Enter your comment here...