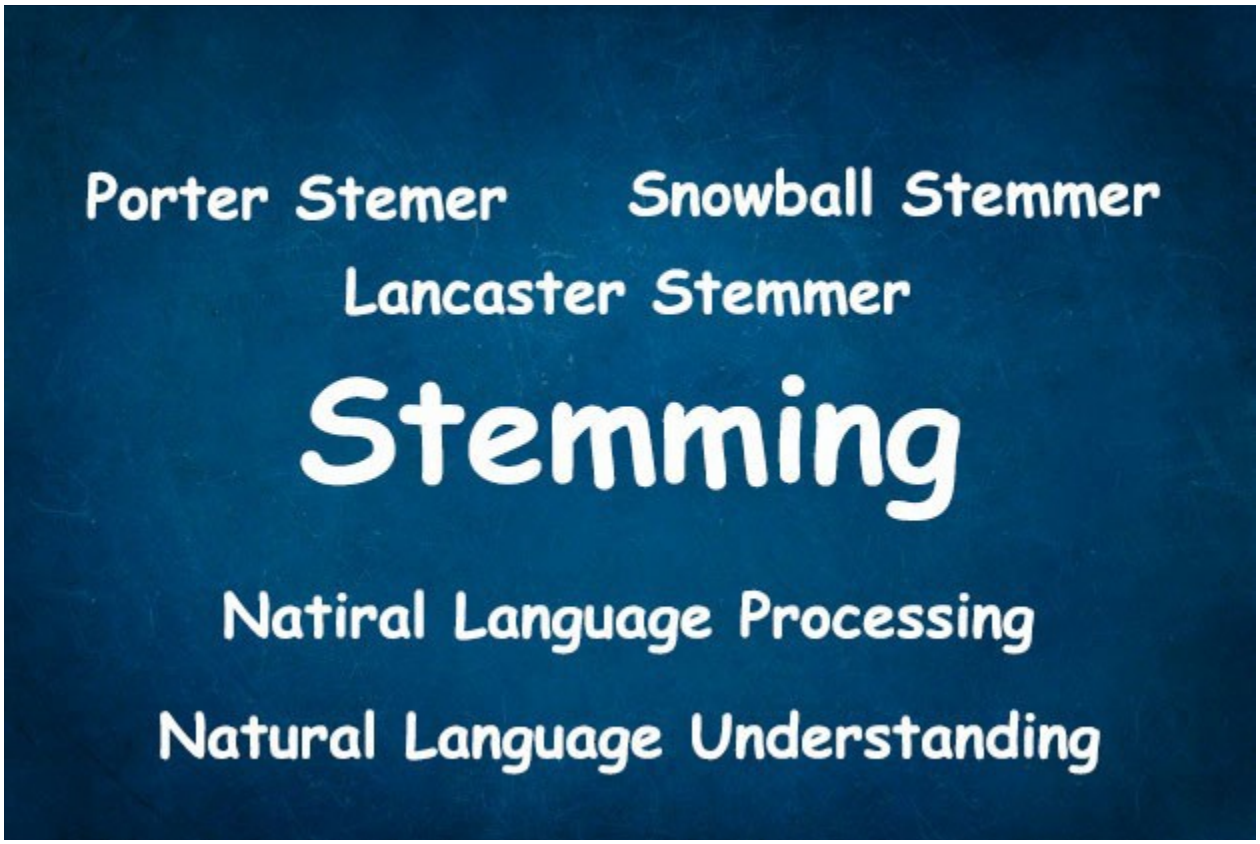
Published in Towards Data Science

Aditya Beri Follow

May 14, 2020 · 5 min read ★ · Listen

Stemming vs Lemmatization

Truncate a word to its root or base unit



Stemming and Lemmatization are text normalization techniques within the field of

lemmatization, and the way to stem and lemmatize words, sentences and documents using the Python nltk package which is the natural language package provided by Python

In natural language processing, you may want your program to acknowledge that the words “kick” and “kicked” are just different tenses of the same verb. this can be the concept of reducing different kinds of a word to a core root.

Stemming

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers.

Often when searching text for a certain keyword, it helps if the search returns variations of the word. For instance, searching for “boat” might also return “boats” and “boating”. Here, “boat” would be the stem for [boat, boater, boating, boats].

Stemming is a somewhat crude method for cataloging related words; it essentially chops off letters from the end until the stem is reached. This works fairly well in most cases, but unfortunately English has many exceptions where a more sophisticated process is required. In fact, spaCy doesn’t include a stemmer, opting instead to rely entirely on lemmatization.

S1		S2	word		stem
SSES	→	SS	caresses	→	caress
IES	→	I	ponies	→	poni
			ties	→	ti
SS	→	SS	caress	→	caress
S	→		cats	→	cat

Porter Stemmer

One of the most common, and effective, stemming tools is Porter’s Algorithm

reduction, each with its own set of mapping rules. In the first phase, simple suffix mapping rules are defined, such as:

```
# Import the toolkit and the full Porter Stemmer library
import nltk

from nltk.stem.porter import *

p_stemmer = PorterStemmer()

words = ['run','runner','running','ran','runs','easily','fairly']

for word in words:
    print(word+' --> '+p_stemmer.stem(word))
```

Output


```
running --> run
ran --> ran
runs --> run
easily --> easili
fairly --> fairli
```

Note how the stemmer recognizes “runner” as a noun, not a verb form or participle. Also, the adverbs “easily” and “fairly” are stemmed to the unusual root “easili” and “fairli”

Snowball Stemmer

This is somewhat of a misnomer, as Snowball is the name of a stemming language developed by Martin Porter. The algorithm used here is more accurately called the “English Stemmer” or “Porter2 Stemmer”. It offers a slight improvement over the original Porter stemmer, both in logic and speed. Since nltk uses the name SnowballStemmer, we’ll use it here.

```
from nltk.stem.snowball import SnowballStemmer

# The Snowball Stemmer requires that you pass a language parameter
s_stemmer = SnowballStemmer(language='english')

words = ['run','runner','running','ran','runs','easily','fairly'

for word in words:
    print(word+' --> '+s_stemmer.stem(word))
```


```
running --> run
ran --> ran
runs --> run
easily --> easili
fairly --> fair
```

In this case, the stemmer performed the same as the Porter Stemmer, with the exception that it handled the stem of “fairly” more appropriately with “fair”

Stemming has its drawbacks. If given the token saw, stemming might always return saw, whereas lemmatization would likely return either see or saw depending on whether the use of the token was as a verb or a noun

Lemmatization

In contrast to stemming, lemmatization looks beyond word reduction and considers a language’s full vocabulary to apply a morphological analysis to words. The lemma of ‘was’ is ‘be’ and the lemma of ‘mice’ is ‘mouse’.

Lemmatization is typically seen as much more informative than simple stemming, which is why Spacy has opted to only have Lemmatization available instead of Stemming

Lemmatization looks at surrounding text to determine a given word’s part of speech, it does not categorize phrases.

```
# Perform standard imports:
import spacy
nlp = spacy.load('en_core_web_sm')

def show_lemmas(text):
    for token in text:
        print(f'{token.text:{12}} {token.pos_:{6}} {token.lemma:
<{22}} {token.lemma_}')
```


```
doc = nlp(u"I saw eighteen mice today!")

show_lemmas(doc)
```

Output

```
I           PRON   561228191312463089     -PRON-
saw         VERB   11925638236994514241   see
eighteen    NUM    9609336664675087640    eighteen
mice        NOUN   1384165645700560590    mouse
today       NOUN   11042482332948150395   today
!           PUNCT  17494803046312582752   !
```

Notice that the lemma of `saw` is `see`, `mice` is the plural form of `mouse`, and yet `eighteen` is its own number, *not* an expanded form of `eight`.

CONCLUSION

> *One thing to note about lemmatization is that it is harder to create a lemmatizer in a new language than it is a stemming algorithm because we require a lot more knowledge about structure of a language in lemmatizers.*
>
> *Stemming and Lemmatization both generate the foundation sort of the inflected words and therefore the only difference is that stem may not be an actual word whereas, lemma is an actual language word.*
> *Stemming follows an algorithm with steps to perform on the words which makes it faster. Whereas, in lemmatization, you used a corpus also to supply lemma which makes it slower than stemming. you furthermore might had to define a parts-of-speech to get the proper lemma.*
>
> *The above points show that if speed is concentrated then stemming should be used since*

Note- All the code explained has been given in the GitHub repo with some more examples to enhance your knowledge and get a better grip over this topic. Also, extra concepts on stop words and vocabulary have been included there. Click on the link below:-

jn aditya-beri/Stemming-vs-Lemmatization

Contribute to aditya-beri/Stemming-vs-Lemmatization development by creating an account on GitHub.

github.com

This was just a small sneak peek into what stemming and lemmatization is and how they work.
Feel free to respond to this blog below for any doubts and clarifications!

Thanks to Yenson Lau and Elliot Gunn