Lab 4: Text Normalization: Stemming and Lemmatization

**This session covers:**

* Different types of Stemmers & stemming 🡪 Porter | Lancaster | SnowBall
* Different types of Lemmatizer & lemmatizing 🡪 WordNet lemmatizer | textblob lemmatizer
* Lemmatization with & without POS tags

**Learning Outcome:**

Apply text mining and natural language processing methodologies to textual data.

Quick Review

Mapping from foxes to fox is called stemming. Morphological STEMMING parsing or stemming applies to many affixes other than plurals; for example we might need to take any English verb form ending in -ing (going, talking, congratulating) and parse it into its verbal stem plus the -ing morpheme.

The Porter algorithm is a simple and efficient way to do stemming, stripping off affixes. It is not as accurate as a transducer model that includes a lexicon, but may be preferable for applications like information retrieval in which exact morphological structure is not needed.

Practices

* 1. Stemmers

Use the simple following code to implement ***porter stemmer***.

from nltk.stem import PorterStemmer

from nltk.tokenize import word\_tokenize

ps = PorterStemmer()

example\_words1 = ["python", "pythoner", "pythoning", "pythoned", "pythonly"]

example\_words2 = ["List", "listed", "lists", "listing", "listings"]

for w in example\_words1:

print(ps.stem(w))

for w in example\_words2:

print(ps.stem(w))

You can use this algorithm combining with a tokenization code in order to stem the words in a sentence, as below:

from nltk.stem import PorterStemmer

from nltk.tokenize import word\_tokenize

ps = PorterStemmer()

new\_text = """It is very important to be pythonly while you are pythoning

with python. All pythoners have pythoned poorly at least once."""

words = word\_tokenize(new\_text)

print([ps.stem(w) for w in words])

NLTK includes several off-the-shelf stemmers, and if you ever need a stemmer you should use one of these in preference to crafting your own using regular expressions, since these handle a wide range of irregular cases. The Porter and ***Lancaster stemmers*** follow their own rules for stripping affixes. Observe that the Porter stemmer correctly handles the word lying (mapping it to lie), while the Lancaster stemmer does not.

from nltk.stem import PorterStemmer, LancasterStemmer

from nltk.tokenize import word\_tokenize

raw = """DENNIS: Listen, strange women lying in ponds distributing swords

is no basis for a system of government. Supreme executive power derives from

a mandate from the masses, not from some farcical aquatic ceremony."""

tokens = word\_tokenize(raw)

porter = PorterStemmer()

lancaster = LancasterStemmer()

print([porter.stem(t) for t in tokens])

print("\n").

print([lancaster.stem(t) for t in tokens])

Compare and discuss the results of two stemmers (Porter and Lancaster), if you observe any difference.

* 1. Lemmatization using *WordNet lemmatizer*

The ***WordNet lemmatizer*** only removes affixes if the resulting word is in its dictionary. This additional checking process makes the lemmatizer slower than the above stemmers.

from nltk.stem import WordNetLemmatizer, PorterStemmer

lemmatizer = WordNetLemmatizer()

print("rocks :", lemmatizer.lemmatize("rocks"))

print("\nproduced :", lemmatizer.lemmatize("produced", pos ="v"))

ps = PorterStemmer()

print("\nStem of the word produced :", ps.stem("produced"))

print("\nbetter :", lemmatizer.lemmatize("better", pos ="a"))

print("\nwomen :", lemmatizer.lemmatize("women", pos ="n"))

Notice that it doesn't handle *lying*, but it converts *women* to *woman*.

from nltk.stem import PorterStemmer

from nltk.tokenize import word\_tokenize

raw = """DENNIS: Listen, strange women lying in ponds distributing swords

is no basis for a system of government. Supreme executive power derives from

a mandate from the masses, not from some farcical aquatic ceremony."""

tokens = word\_tokenize(raw)

wnl = nltk.WordNetLemmatizer()

print([wnl.lemmatize(t) for t in tokens])

print()

for t in tokens:

print ("{0:20}{1:20}".format(t, wnl.lemmatize(t, pos="v")))

print()

example\_words = ["List", "listed", "lists", "listing", "listings"]

print([wnl.lemmatize(w) for w in example\_words])

print()

for words in example\_words:

print ("{0:20}{1:20}".format(words, wnl.lemmatize(words, pos="v")))

The WordNet lemmatizer is a good choice if you want to compile the vocabulary of some texts and want a list of valid lemmas (or lexicon headwords). The results would result lemma not as 100% accurate according to the lemmas found in the dictionary.

However, to have the exact lemma as per the dictionary, POS tagging better be included in the code.

for t in tokens:

print ("{0:20}{1:20}".format(t, wnl.lemmatize(t, pos="v")))

* 1. Lemmatization using *TextBlob*

Words can be lemmatized by calling the [*lemmatize*](https://textblob.readthedocs.io/en/dev/api_reference.html#textblob.blob.Word.lemmatize) method via the [*TextBlob*](https://textblob.readthedocs.io/en/dev/api_reference.html#textblob.blob.TextBlob) objects

from textblob import TextBlob

sentence = TextBlob('DENNIS: Listen, strange women lying in ponds distributing swords is no basis for a system of government. Supreme executive power derives from a mandate from the masses, not from some farcical aquatic ceremony.')

tokens = sentence.words

print(tokens)

print

tokens.lemmatize()

for t in tokens:

print ("{0:20}{1:20}".format(t, wnl.lemmatize(t, pos="v")))

* 1. Stemming & Lemmatization

### ***Stemming*** and ***Lemmatization*** both generate the root form of the inflected words. The difference is that stem might not be an actual word whereas, lemma is an actual language word.

import nltk

from nltk.stem import PorterStemmer

from nltk.stem import LancasterStemmer

from nltk.tokenize import word\_tokenize

from nltk.stem.snowball import SnowballStemmer

#file = open ("D:/APU/TXSA-CT107-3-3/TUTORIAL/sample01.txt")

#raw = file.read()

raw = """DENNIS: Listen, strange women lying in ponds distributing swords

is no basis for a system of government. Supreme executive power derives from

a mandate from the masses, not from some farcical aquatic ceremony."""

words = raw.lower()

print(words)

print()

tokens = word\_tokenize(words)

print("Tokens")

print(tokens)

print()

print("Lemmas")

wnl = nltk.WordNetLemmatizer()

print([wnl.lemmatize(t, pos = "v") for t in tokens])

print()

print("Porter Stemming")

ps = PorterStemmer()

print ([ps.stem(t) for t in tokens])

print()

print("Lancaster Stemming")

ls = LancasterStemmer()

print ([ls.stem(t) for t in tokens])

print()

print("Snowball Stemming")

sn = nltk.SnowballStemmer("english")

print([sn.stem(t) for t in tokens])

**NOTE:**

Stemming and Lemmatization both generate the root form of the inflected words. The difference is that stem might not be an actual word whereas, lemma is an actual language word. Whereas, in lemmatization, you used WordNet corpus and a corpus for stop words as well to produce lemma which makes it slower than stemming.

* 1. Stemmers --> *Snowball Stemmer*

import nltk

print(nltk.SnowballStemmer.languages)

print(len(nltk.SnowballStemmer.languages))

print()

text = "This is achieved in practice during stemming, a text preprocessing operation."

tokens = nltk.tokenize.word\_tokenize(text)

print()

stemmer = nltk.SnowballStemmer('english')

print([stemmer.stem(t) for t in tokens])

print()

text2 = "Ceci est réalisé en pratique lors du stemming, une opération de prétraitement de texte."

tokens2 = nltk.tokenize.word\_tokenize(text2)

print()

stemmer = nltk.SnowballStemmer('french')

print([stemmer.stem(t) for t in tokens2])

* 1. Snowball Stemmer --> for other space delimited languages

from textblob import TextBlob

import nltk

en\_blob = TextBlob(u'This is achieved in practice during stemming, a text preprocessing operation.')

print(en\_blob.detect\_language())

fr\_blob = en\_blob.translate(from\_lang="en", to='fr')

print(fr\_blob)

tokens = fr\_blob.words

print(tokens)

print()

stemmer = nltk.SnowballStemmer('french')

print([stemmer.stem(t) for t in tokens])

**References:**

1. Regular Expressions: The Complete Tutorial, by Jan Goyvaerts, 2007.
2. Speech and Language Processing, by Dan Jurafsky and James H. Martin. Prentice Hall Series in Artificial Intelligence, 2008.
3. Natural Language Processing with Python, by Steven Bird, Ewan Klein and Edward Loper, 2014.
4. Lemmatization approaches with examples in Python (<https://www.machinelearningplus.com/nlp/lemmatization-examples-python/>)

**Revision Quiz:**

[*https://quizlet.com/512734559/test?answerTermSides=2&promptTermSides=6&questionCount=7&questionTypes=14&showImages=true*](https://quizlet.com/512734559/test?answerTermSides=2&promptTermSides=6&questionCount=7&questionTypes=14&showImages=true)

**Take-home task:**

Perform the lemmatization including the POS tags such as ADJECTIVE, NOUN, VERB, and ADVERB. Write the suitable python code to get the proper output.