Aim : Apply Various text preprocessing techniques Lemmatization / Stemming.

Objective: To study and implement lemmatization and stemming in python.

Theory:

Stemming:

Stemming is a technique used to reduce an inflected word down to its word stem. For example, the words "programming," "programmer," and "programs" can all be reduced down to the common word stem "program." In other words, "program" can be used as a synonym for the prior three inflection words.

Performing this text-processing technique is often useful for dealing with sparsity and/or standardizing vocabulary. Not only does it help with reducing redundancy, as most of the time the word stem and their inflected words have the same meaning, it also allows NLP models to learn links between inflected words and their word stem, which helps the model understand their usage in similar contexts.

Stemming algorithms function by taking a list of frequent prefixes and suffixes found in inflected words and chopping off the end or beginning of the word. This can occasionally result in word stems that are not real words; thus, we can affirm this approach certainly has its pros, but it's not without its limitations.

Lemmatization:

Lemmatization is another technique used to reduce inflected words to their root word. It describes the algorithmic process of identifying an inflected word's "lemma" (dictionary form) based on its intended meaning.

As opposed to stemming, lemmatization relies on accurately determining the intended part-of-speech and the meaning of a word based on its context. This means it takes into consideration where the inflected word falls within a sentence, as well as within the larger context surrounding that sentence, such as neighboring sentences or even an entire document.

Program: from nltk import

PorterStemmer import spacy nlp =

spacy.load('en_core_web_sm') doc

= nlp(u"hello") for token in doc:

```
    # print(token , "-" , token.lemma_)
print(token , "-" , token.pos_)  stemmer
= PorterStemmer()
 words = ['run', 'runner', 'ran', 'runs', 'easily',
'fairly']  for word in words:  print(word + '-' +
stemmer.stem(word))
```

Output: hello - INTJ

run-run

runner-runner

ran-ran

runs-run

easily-easili

fairly-fairli

Conclusion:

The application of text preprocessing techniques, including lemmatization and stemming, plays a crucial role in standardizing and normalizing textual data for improved NLP analysis and interpretation. By reducing lexical variations and consolidating words to their base or root forms, these techniques enhance the efficiency and accuracy of various NLP tasks, contributing to more effective information retrieval, search relevance, and overall linguistic analysis in diverse domains. Incorporating these preprocessing steps into the NLP pipeline is essential for extracting meaningful insights and patterns from text data, thereby facilitating more accurate and impactful analyses and applications.