NATURAL LANGUAGE PROCESSIONG (NLP)

Natural language processing (NLP) is a subfield of linguistics computer science, and artificial intelligence concerned with the interactions between computers and human language, in particular how to program computers to process and analyze large amounts of natural language data.

Natural Language ToolKit (NLTK)

Natural Language Toolkit (NLTK) is a Python library used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).

NLTK contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets for NLP.

TOKENIZATION

tokenization is the process of breaking up the original raw text into component pieces which are known as tokens. Tokens are the building blocks of Natural Language. Types of tokenization:

Character Tokenization is the process of breaking text into a list of characters.

Syntax:

```
text="Hello world"

list=[x for x in text]

print(list)
```

Word Tokenization is the process of breaking a string into a list of words.

Syntax:

```
from nltk.tokenize import word_tokenize

text="Hello there! Welcome to the programming world."

print(word_tokenize(text))
```

Sentence Tokenization is the process of breaking a paragraph or a string containing sentences into a list of sentences.

Syntax:

```
from nltk.tokenize import sent_tokenize

text="It’s easy to point out someone else’s mistake. Harder to recognize your own."

print(sent_tokenize(text))
```

Whitespace Tokenization module of NLTK tokenizes a string on whitespace (space, tab, newline). It is an alternate option for split().

Syntax:

```
from nltk.tokenize import WhitespaceTokenizer

s="Good muffins cost $3.88\nin New York.  Please buy me\ntwo of them.\n\nThanks."

Tokenizer=WhitespaceTokenizer()

print(Tokenizer.tokenize(s))
```

Word Punctuation Tokenizer module of NLTK tokenizes a string on punctuations.

Syntax:

```
from nltk.tokenize import WordPunctTokenizer

text="We're moving to L.A.!"

Tokenizer=WordPunctTokenizer()

print(Tokenizer.tokenize(text))
```

Regexp Tokenizer function of NLTK is used to remove punctuations from a sentence.

Syntax:

```
from nltk.tokenize import RegexpTokenizer

text="The children - Pierre, Laura, and Ashley - went to the store."

tokenizer = RegexpTokenizer(r"\w+")

lst=tokenizer.tokenize(text)

print(' '.join(lst))
```

DIFFERENCE BETWEEN NLTK TOKENIZE AND SPLIT

The split function is usually used to separate strings with a specified delimiter, e.g. in a tab-separated file, we can use str.split ('\t') or when we are trying to split a string by the newline '\n' when our text file has one sentence per line or when we are trying to split by any specific character. But the same can't be performed using the NLTK tokenize functions.

LEMMATAZATION:

Lemmatization is the process of grouping together different inflected forms of words having the same root or lemma for better NLP analysis and operations. The lemmatization algorithm removes affixes from the inflected words to convert them into the base words (lemma form).

Lemmatization understands the context of the word by analyzing the surrounding words and then convert them into lemma form. For example, “running” and “runs” are converted to its lemma form “run”.

Lemmatization helps to create better NLP models like Bag of Word, TF-IDF that depend on the frequency of the words. At the same time, it also increases computational efficiency.

NLTK LEMMATIZATION WITH WORDNETLEMMATIZER

Wordnet is a database of the English language that is used by NLTK internally. WordNetLemmatizer is an NLTK lemmatizer built using the Wordnet database and is very widely used.

There is one catch due to which NLTK lemmatization might not work. The NLTK lemmatizer requires POS tag information to be provided explicitly otherwise it assumes POS to be a noun by default and the lemmatization will not give the right results.

WordNetLemmatizer without pos_tag:

If we do not provide pos_tag, then by default it will be considered as noun ‘n’ and lemmatizer will not work properly.

Example:

```
from nltk.stem import WordNetLemmatizer

from nltk.tokenize import word_tokenize



text = "She jumped into the river and breathed heavily"

wordnet = WordNetLemmatizer()

tokenizer = word_tokenize(text)



for token in tokenizer:

    print(token,"--->",wordnet.lemmatize(token))
```

WordNetLemmatizer with pos_tag:

Syntax:

```
from nltk.stem import WordNetLemmatizer

from nltk import word_tokenize,pos_tag
```

```
text = "She jumped into the river and breathed heavily"

wordnet = WordNetLemmatizer()



for token,tag in pos_tag(word_tokenize(text)):

    pos=tag[0].lower()



    if pos not in ['a', 'r', 'n', 'v']:

        pos='n'



    print(token,"--->",wordnet.lemmatize(token,pos))
```

APPLICATION OF LEMMATIZATION

- Lemmatization is used to reduce text redundancy by converting words having the same meaning but different inflected forms to their base form.
- The reduced word density of redundant text helps to create better NLP models that are efficient and also computationally fast.

STEMMING VS LEMMATIZATION

- Stemming does not take the context of the word into account, for example, “meeting” can be a verb or noun based on the context. But lemmatization does consider the context of the word before generating its lemma.
- Stemming does not take the context of the word into account, for example, “meeting” can be a verb or noun based on the context. But lemmatization does consider the context of the word before generating its lemma.
- The stemming process just follows the step-by-step implementation of algorithms like SnowBall, Porter, etc. to derive the stem. Whereas lemmatization makes use of a lookup database like WordNet to derive lemma.