STOPWORDS :-

Stopwords are common words in a language that usually do not add important meaning to text, so they are often removed during the NLP processing.

Example of Stopwords :-

is, am, are, the, a, an, in, on

Example :-

Sentence :- This is a machine learning project

After Removing Stopwords :-

machine learning project

Syntax :-

```
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')   # downloading stopwords from nltk.
```

```
# we use stopwords.words('english')
```

to get stopwords fro english language

Main Code :-

paragraph = "God of war - - - - -
- - - - - - - September 2024."

from nltk.tokenize import sent_tokenize
first we convert paragraph into sentences using sent_tokenize

Sentences = sent_tokenize(paragraph)
Storing all sentences in object → 'sentences'

from nltk.tokenize import word_tokenize
will use word_tokenize to convert all sentences into sepearte words

from nltk.stem import PorterStemmer
Stemmer_object = PorterStemmer()
So here we are using Porter stemmer also, to reduce words.

from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
} importing stopwords

```
for i in range (len(sentences)):
    words = word_tokenize(sentcs[i])
    # sentences[i] -> take one sentence
    # word_tokenize -> split sentence in words
    words = [stemmer-object.stem(word) for word
             in words if word not in
             set(stopwords.words('english'))]
# using List comprehension here
    for word in words --> check each word
    if word not in stpwords -> skip common words
    stemmer_object.stem(word) -> shorten the word
    [...] -> making a new list

    Sentences[i] = " ".join(words)
Joining all words together in one sentence

    print(sentences)
```

Note :- here we are using porter stemmer, we can also use Regexp stemme and Snowball stemmer but to get the best out of all we should use lemmatizer.

Main Code Using Lemmatization :-

paragraph = { God of War ——— 2024. }

converting into sentence

```
from nltk.tokenize import sent_tokenize
sentences = sent_tokenize(paragraph)
```

import word_tokenize and lemmatizer

```
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

import stopwords

```
from nltk.corpus import stopwords
import nltk
nltk.download('english stopwords')
```

```
for i in range(len(sentences)):
    words = word_tokenize(sentences[i])
words = [lemmatizer.lemmatize(word,Pos='v') for word in words
        if word not in set(stopwors.word('english'))]
```

Sentences[i] = " ".join(words)

in this we are using lemmatizer.lemmatize(word, pos='v')

POS_TAG

POS (Part of Speech) tagging means assigning grammatical labels to words. like

- Noun
- Verb
- adjective
- adverb, etc

Pos tagging in NltK :-

nltk.pos_tag()

Example :-

```
import nltk
from nltk.tokenize import word_tokenize

Sentence = "Taj Mahal is a beautiful Monument"
words = word_tokenize(sentece)

pos_tag = nltk.pos_tag(words)

print (pos_tag)
```

[('Taj', 'NNP'), ('Mahal', 'NNP'), ('is', 'VBZ'), ('a', 'DT') ...

Common Pos tags

Tag	Meanings
NN	Noun
NNP	Proper Noun
VB	Verb
VBZ	Verb (3rd peron singular)
DT	Determiner

NAMED-ENTITY RECOGNITION

Named-Entity Recognition is an NLP technique used to identify and classify real world entities in text.

What Named-Entity Recoguition detects -

- PERSON → name of people
- ORG → organisation
- GPE / LOCATION → country, city, place
- DATE / TIME
- MONEY / PERCENT

Example :-

```
import nltk
from nltk.tokenize import word_tokenize
from nltk import pos_tag, ne_chunk

# ne_chunk is used for Named Entity Recognition.

Sentence = " Ritik Sharma works at Google in India"
words = word_tokenize (sentence)
postag = pos-tag (words)
ner = ne_chunk (pos_tags)
print (ner)
```

Output :-

```
(S
    (PERSON RITIK / NNP)
    (PERON SHarma / NNP)
    works / VBZ
    at / IN
    (ORGANIZATION Google / NNP)
    in / IN
    (GPE India / NNP))
```

But, we can also draw this using (.draw) or (ne_chunk (pos_tags).draw)

continue from there

```
ner = ne_chunk(pos_tags).draw()
print(ner)
```

output :-