

In [20]: *# 1. Load the basic libraries and packages*

```
import spacy
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from spacy import displacy
from PIL import Image
import io
import cairosvg

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data] Package punkt_tab is already up-to-date!
```

Out[20]: True

In [21]: *# Sample Text*

```
text = "Natural Language Processing (NLP) is a field of AI that focuses on the inte
```

In [22]: *# 1. Tokenization*

```
sent_tokens = sent_tokenize(text)
word_tokens = word_tokenize(text)

print("\nSentence Tokenization:")
print(sent_tokens)

print("\nWord Tokenization:")
print(word_tokens)
```

Sentence Tokenization:

```
['Natural Language Processing (NLP) is a field of AI that focuses on the interaction between computers and human language.', 'It enables machines to read, understand, and interpret human language.']
```

Word Tokenization:

```
['Natural', 'Language', 'Processing', '(', 'NLP', ')', 'is', 'a', 'field', 'of', 'AI', 'that', 'focuses', 'on', 'the', 'interaction', 'between', 'computers', 'and', 'human', 'language', '.', 'It', 'enables', 'machines', 'to', 'read', ',', 'understand', ',', 'and', 'interpret', 'human', 'language', '.']
```

In [23]: *# 2. Filtration*

```
filtered_tokens = [word for word in word_tokens if word.isalpha()]
print("After Filtration (Only Words):")
print(filtered_tokens)
```

After Filtration (Only Words):

```
['Natural', 'Language', 'Processing', 'NLP', 'is', 'a', 'field', 'of', 'AI', 'that',
'focuses', 'on', 'the', 'interaction', 'between', 'computers', 'and', 'human', 'language', 'It', 'enables', 'machines', 'to', 'read', 'understand', 'and', 'interpret', 'human', 'language']
```

In [24]: *# 3. Stopwords Removal*

```
stop_words = set(stopwords.words('english'))
tokens_without_stopwords = [word for word in filtered_tokens if word.lower() not in stop_words]
print("After Stopwords Removal:")
print(tokens_without_stopwords)
```

After Stopwords Removal:

```
['Natural', 'Language', 'Processing', 'NLP', 'field', 'AI', 'focuses', 'interaction', 'computers', 'human', 'language', 'enables', 'machines', 'read', 'understand', 'interpret', 'human', 'language']
```

In [25]: *# 4. PoS Tagging*

```
nlp = spacy.load("en_core_web_sm")
doc = nlp(text)

print("Part-of-Speech (PoS) Tagging:")
for token in doc:
    print(f"{token.text:<15} {token.pos_:<10} {token.dep_:<10}")
```

Part-of-Speech (PoS) Tagging:

Natural	PROPN	compound
Language	PROPN	compound
Processing	PROPN	nsubj
(PUNCT	punct
NLP	PROPN	appos
)	PUNCT	punct
is	AUX	ROOT
a	DET	det
field	NOUN	attr
of	ADP	prep
AI	PROPN	pobj
that	PRON	nsubj
focuses	VERB	relcl
on	ADP	prep
the	DET	det
interaction	NOUN	pobj
between	ADP	prep
computers	NOUN	pobj
and	CCONJ	cc
human	ADJ	amod
language	NOUN	conj
.	PUNCT	punct
It	PRON	nsubj
enables	VERB	ROOT
machines	NOUN	nsubj
to	PART	aux
read	VERB	ccomp
,	PUNCT	punct
understand	VERB	conj
,	PUNCT	punct
and	CCONJ	cc
interpret	VERB	conj
human	ADJ	amod
language	NOUN	dobj
.	PUNCT	punct

In [26]: *# 5. Noun Phrase Chunking*

```
print("\nNoun Phrase Chunking:")
for chunk in doc.noun_chunks:
    print(f"Chunk: {chunk.text} | Root: {chunk.root.text} | Dep: {chunk.root.dep_}")
```

Noun Phrase Chunking:

Chunk: Natural Language Processing | Root: Processing | Dep: nsubj | Head: is
Chunk: NLP | Root: NLP | Dep: appos | Head: Processing
Chunk: a field | Root: field | Dep: attr | Head: is
Chunk: AI | Root: AI | Dep: pobj | Head: of
Chunk: that | Root: that | Dep: nsubj | Head: focuses
Chunk: the interaction | Root: interaction | Dep: pobj | Head: on
Chunk: computers | Root: computers | Dep: pobj | Head: between
Chunk: human language | Root: language | Dep: conj | Head: computers
Chunk: It | Root: It | Dep: nsubj | Head: enables
Chunk: machines | Root: machines | Dep: nsubj | Head: read
Chunk: human language | Root: language | Dep: dobj | Head: interpret

In [32]: # 6. Dependency Parsing

```
print("Dependency Parsing Visualization:")  
  
displacy.render(nlp("I am Learning Artificial Intelligence at 11:40AM in MA112."),
```

Dependency Parsing Visualization:

