Natural Language analysis

Is a part of Natural Language Processing.

refers to the process of examining and understanding human language using computational techniques

All NLP software typically works at the sentence level and expects a separation of words at the minimum.
So, we need some way to split a text into **words** and **sentences** before proceeding further in a processing pipeline.

Sometimes, we need to remove **special characters** and **digits**, and sometimes, we don't care whether a word is in **upper** or **lowercase** and want everything in lowercase. Many more decisions like these are made in this step.

Natural Language analysis steps

Morphological analysis — Work on the Word

Syntactic analysis — Work on the gramatical relation between the words

Semantic analysis — Work on semantical relation between the words

Morphological analysis

to identify the **morphemes** in a word

the smallest units of meaning

word

Stem/root

Affix
Prefix
suffix

Two types of morphemes:

- **<u>Free morphemes</u>**: stand alone as a word (book; run; go..).
- **<u>Bound morphemes</u>**: Cannot stand alone and must be attached to other morphemes (prefixes like "un-" in "undo”, or suffixes like "-ed" in "played").

Cats = cat
Adjustable= adjust
Does= do
Better =?
Went =?

Better = good/ well
Went= go

Morphological analysis

Finite State Transducers

A finite state transducer (FST) is a finite state machine with two tapes: an input tape and an output tape,
with finite number of states.

M= <Q, ∑, q0, δ, F>

Q: {q0, q1, …, qn-1}
∑ : finite alphabet of complex symbols, $\sum \epsilon$ I×O: I set of input of alphabet
O set of output alphabet

q0 : start state
δ : Q × ∑ δ is a transition function
F: final state

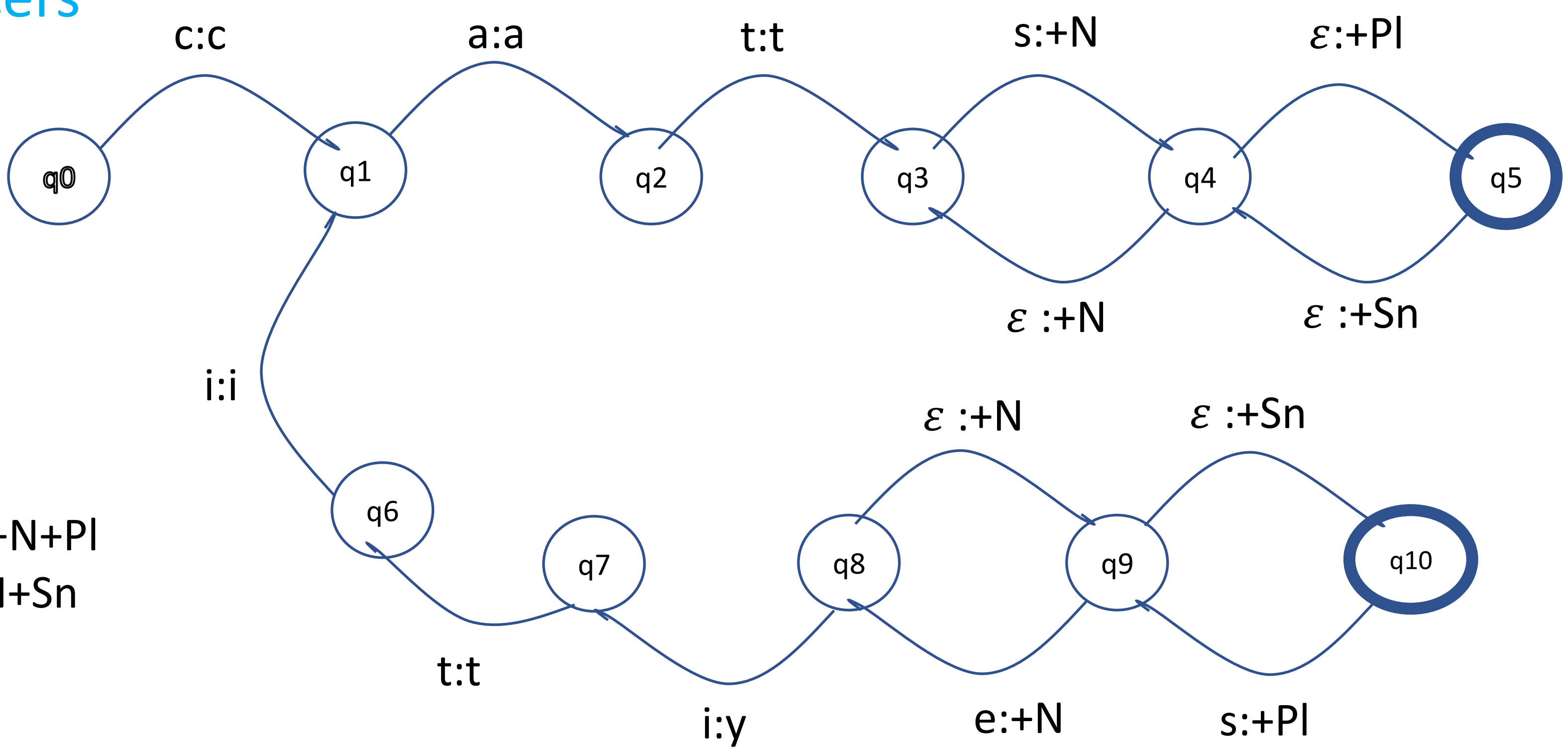
The input and output symbols both include empty string (ε)

Morphological analysis

examples

Finite State Transducers

Cats= cat +N+Pl
Cat= cat +N+Sn



Cities= city +N+Pl
City= city +N+Sn

What about the word feet?

Syntactic analysis

To analyze the grammatical structure of a sentence to determine **how** the words **relate** to **each other**.

The target of Syntactic analysis is to:

✓ Find the roles played by words in a sentence.

✓ Interpret the relationship between words.

✓ Interpret the grammatical structure of sentences.

Syntactic analysis

Context Free Grammar

It is a type of formal grammar used to define the syntax of languages,

Component of CFG

1. A set of nonterminal symbols **N** are placeholders for patterns of terminal symbols created by nonterminal symbols. These symbols usually located at the LHS (left-hand-side) of production rules (P). The strings generated by CFG usually consist of symbols only from nonterminal symbols.

2. A set of terminal symbols **Σ** (disjoint from N) are characters appear in strings generated by grammar. Terminal symbols usually located only at RHS (right-hand-side) of production rules (P).

3. A set of production rules **P: A → α**,where A is a nonterminal symbol and α is a string of symbols from the infinite set of strings (Σ ∪ N).

4. A designated start symbol **S** is a start symbol of the sentence

Syntactic analysis

Context Free Grammar

typical non-terminal components

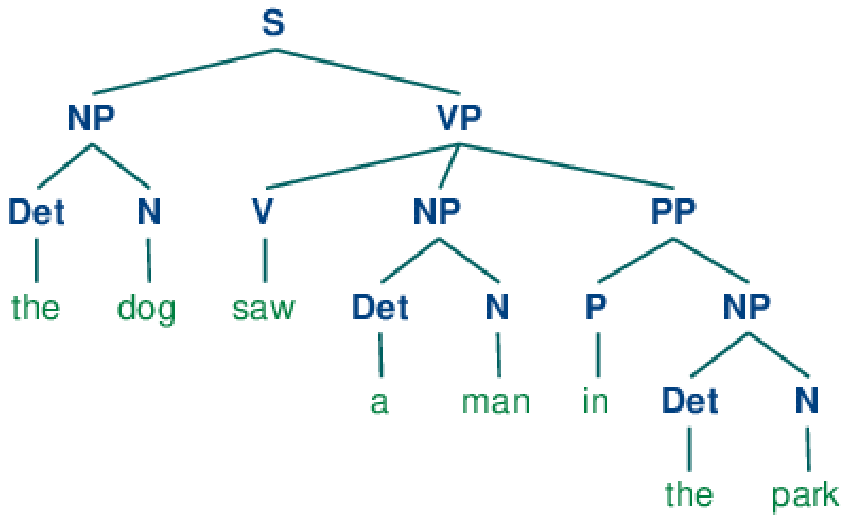
Non terminal component	symbol
Noun phrase	NP
Verb phrase	VP
Prepositional phrase	PP
Determiner	Det (the, a ..)
complementizer	C (that, who…)
Auxiliary	Aux
Conjunction	Conj (and, but..)
Noun	N
Verb	V
adjective	Adj
Adverb	Adv
preposition	P (on, under…)

Syntactic analysis

Example: The dog saw a man in the park

The cat chased the mouse

The boy who was hungry ate the pizza



Semantical analysis

understanding the **meaning** of words and sentences.

interpreting the **correct context** of words or phrases with **multiple meanings**

Example1: Mary was excited about the party, but she forgot to bring her gift

The system identifies that "she" and "her" refer to "Mary" based on the context of the sentence.

Example2: John gave Salma a book

John is the **giver**

Gave is the **action**

Salma the **receiver**

Book is the **object**

Semantical analysis

Conceptual dependency theory

to help computers understand the meanings of sentences in human languages. The theory focuses not on how a sentence is written (its grammar), but on what the sentence **means.**

It focuses on the **actions** and **people involved**. It tries to capture the meaning of these sentences by describing what is happening.

CD theory reduces sentences to a series of **primitive actions** and relationships between **concepts**.

Example:

-John gave a book to Mary.

-Mary received a book from John.

-The book was given to Mary by John.

Semantical analysis

Conceptual dependency theory

Conceptual categories

category	meaning
PP	Real world object
ACT	Action
PA	modifier of object
AA	modifier of action
T	time
LOC	location

The used actions

Action	meaning	example
ATRANS	Abstract transfer (ownership)	Give
PTRANS	Physical transfer of an object	Go
PROPEL	Applying physical force to an object	Push
MTRANS	Mental transfer (information)	Think
MBUILD	Construct new information	Decide
SPEAK	Utter a sound	say
INGEST	Eating or drinking	eat
EXPEL	Expulsion of something from body	cry

Semantical analysis

Conceptual dependency theory

CD theory breaks down sentences by clearly defining:

•**Who** is doing the action.

•**What** the action is.

•**Who or what** is receiving the action.

Using rules

Semantical analysis

CD theory

Rule 1: PP⟺ACT : relationship between an actor and the action he did

ex: John go

John ⟺ PTRANS

Rule 2: ACT⟵ PP : relationship between an action the object of action

Ex: John shoved mary

John ⟺ PROPEL⟵ mary

Semantical analysis

CD theory

Rule 3: PP ⟺ PP : relationship between two PP, one of which belongs to the set difined by the other

ex: John is an engineer

John ⟺ engineer

Rule 4: PP ← PP : relationship between two PP, one of which provides an information about the other

Ex: John’s car

Car ← john

Semantical analysis

CD theory

Rule 5: PP⟺ PA : relationship between PP and PA that is used to describe it (PA is a state of PP)

ex: John is tall

John ⟺ hight (>80)

Rule 6: PP ⟵ PA : relationship between PP and the attribute that has been predicated of it

Ex: genius artist

Artist ⟵ genius

Semantical analysis

CD theory

PP

Rule 7: ACT ← R

PP

: relationship between the action and the source and the recipient of the action

ex: John gave the cat to Bill

Bill

John ⟺ ATRANS ←

John

cat

PA

Rule 8: PP ←

PA

: relationship that describes the change in a state

Ex: The child grows

size > 0.50

Child ←

size = 0.50

Semantical analysis

CD theory

$$\begin{array}{l} \Leftrightarrow \{x\} \\ \uparrow \\ \Leftrightarrow \{y\} \end{array}$$

Rule 9:

: relationship between one conceptualisation and another that cases it. {x} causes {y}

ex: I studied hard, I got good marks

I studied hard

$\uparrow$

I got good marks

Rule 10:

$$\begin{array}{l} \Leftrightarrow \{x\} \\ \downarrow \\ \Leftrightarrow \{y\} \end{array}$$

: relationship between two conceptualisations happening in the same time

Ex: while I am going home, I lost my keys

I am going home

$\downarrow$

I lost my keys

Semantical analysis

CD theory

Example:

While John eats his big burger, he watches TV

Preprocessing

raw text is often noisy and unstructured, containing irrelevant information that could hinder accurate analysis.

To extract meaningful patterns from text, various steps and techniques are applied

- Sentence segmentation and word tokenization
- Stop word removal, stemming and lemmatization, removing digits/punctuation, lowercasing
- POS tagging, parsing, coreference resolution

Preprocessing

Stop word removal

The goal is to remove common words (called **stop words**) that do not carry significant meaning and are not useful for tasks

Removing stop-words can reduce noise in the data
May not always be appropriate

Like: the, is, in, and, on, at, for…

We use it for: text classification, sentiment analysis, or information retrieval.

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("This is a sample sentence with some stop words")
filtered_tokens = [token.text for token in doc if not token.is_stop]
print(filtered_tokens)
```

```
['sample', 'sentence', 'stop', 'words']
```

Preprocessing

Stop word
removal

Preprocessing

Digit/punctuation removing and lowercasing

Removing Punctuation marks (like commas, periods, question marks, etc.) simplifies the text by focusing on words, because they are generally used for sentence structure but often do not add meaning for text analysis tasks.

Lowercasing the text makes the data more uniform. This ensures that words are treated as the same word,

May lose some information (e.g., proper nouns, acronyms)

example

The cost of the phone is $799. It's an amazing deal!!!"

The cost of the phone is it’s an amayzing deal

cost phone amayzing deal

Preprocessing

Sentence segmentation

We can do sentence segmentation by breaking up text into sentences at the appearance of full stops and question marks. It handles punctuation ambiguities like abbreviations (e.g., "Dr.", "Mr.") and numbers (e.g., "5.6").

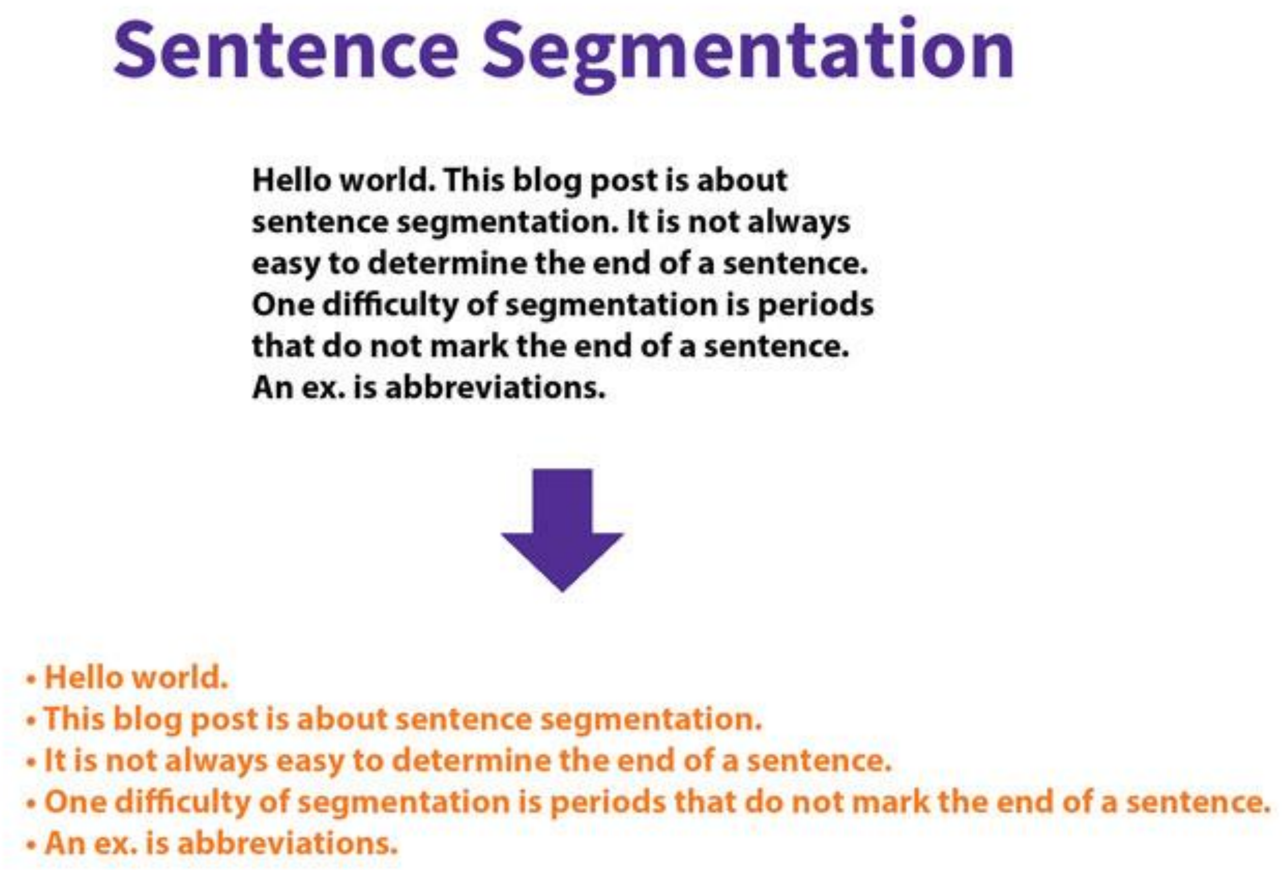
Sentence segmentation is an important preprocessing step because most NLP tasks, like question answering, summarization, and document clustering, operate at the sentence level.

```
import spacy
nlp=spacy.load('en_core_web_sm')
doc1=nlp(u'''Sentence segmentation in Natural Language Processing (NLP) is the process of dividing a body of text into individu
for sentence in doc1.sents:
  print(sentence)
```

```
Sentence segmentation in Natural Language Processing (NLP) is the process of dividing a body of text into individual sentences.
It identifies the boundaries between sentences in a given text.
This identification is often by detecting punctuation marks (e.g., periods, question marks, and exclamation points) or specific
```

Preprocessing

Sentence segmentation



Preprocessing

Word tokenization

is the process of breaking a string of text into individual words or tokens.

These tokens are the smallest units of meaning in the text.

Types of tokenization

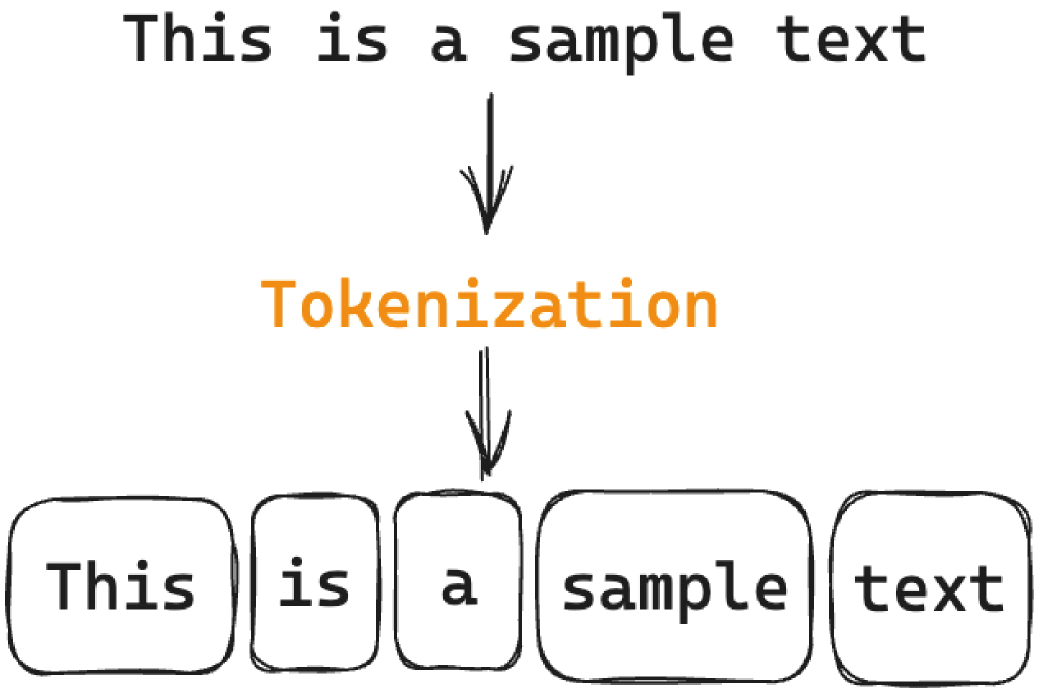
- Word Tokenization : Splits text into individual words
- Sentence Tokenization: Splits text into sentences.
- Subword Tokenization: Splits words into **morphemes** or subwords.
- Character-level Tokenization : Splits text into individual characters.

```
import spacy
nlp=spacy.load('en_core_web_sm')
doc1=nlp(u'''Sentence segmentation in Natural Language Processing (NLP) is the process of dividing a body of text into individual sentences. It identifies
print([token.text for token in doc1])
```

```
['Sentence', 'segmentation', 'in', 'Natural', 'Language', 'Processing', '(', 'NLP', ')', 'is', 'the', 'process', 'of', 'dividing', 'a', 'body', 'of', 'text
```

Preprocessing

Word tokenization



Example:

How to tokenize Isn’t

WT: isn ‘t

SW: is ##n’t

CT: i s n ’ t

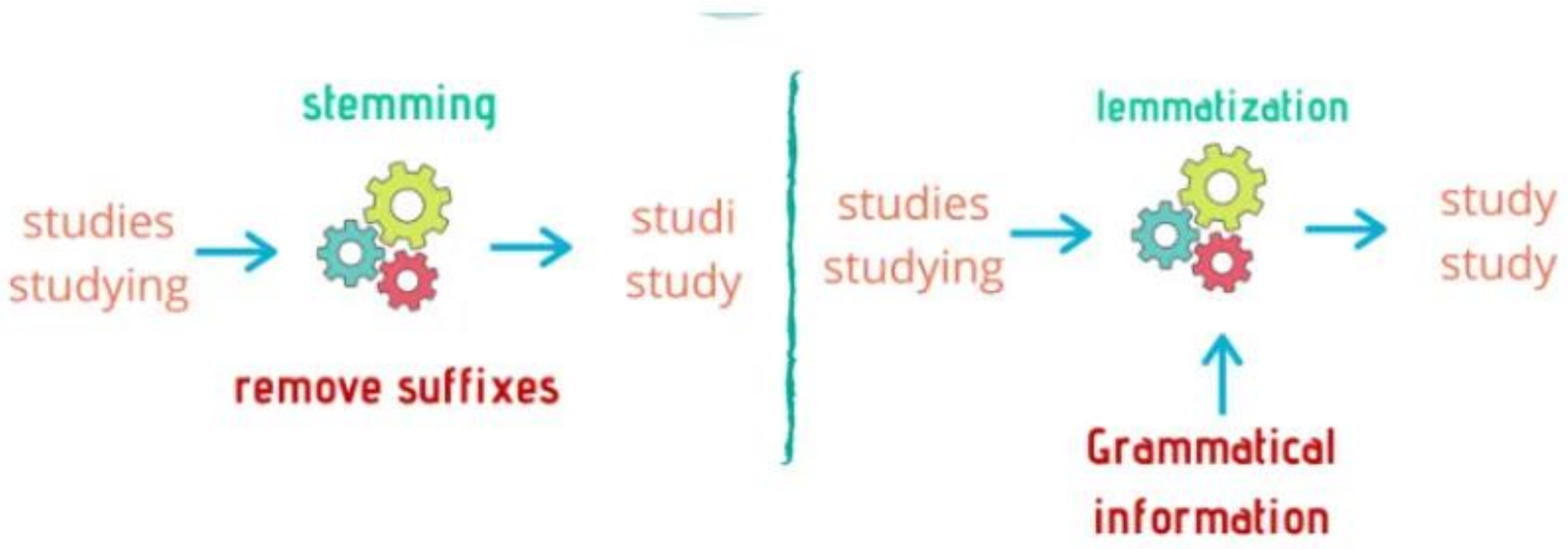
Preprocessing

Stemming and lemmatization

Stemming involves removing suffixes from words to obtain their base form. Fast but can produce non-words

Lemmatization involves converting words to their morphological base form. it produces valid dictionary words

More accurate but slower than stemming



Preprocessing

Stemming and lemmatization

- Accuracy: Lemmatization is generally more accurate
- Speed: Stemming is typically faster
- Output: Lemmatization produces real words, stemming may not
- Context: Lemmatization can use context (POS), stemming doesn’t
- Resource usage: Lemmatization requires more computational ressources

Preprocessing

Stemming and lemmatization

```
import nltk

from nltk.stem.porter import *
stemmer = PorterStemmer()
tokens = ['compute', 'computer', 'computed', 'computing']
for token in tokens:
    print(token + ' --> ' + stemmer.stem(token))
```

```
compute --> comput
computer --> comput
computed --> comput
computing --> comput
```

```
import spacy
nlp = spacy.load("en_core_web_sm")
sentence6 = nlp(u'compute computer computed computing')
for word in sentence6:
    print(word.text,  word.lemma_)
```

```
compute compute
computer computer
computed compute
computing computing
```

Preprocessing

Advanced Preprocessing Techniques

- Spelling Correction
- Named Entity Recognition (NER)
- Part-of-Speech (POS) Tagging
- Text Normalization (e.g., date formats, numbers)
- Handling Emojis and Emoticons
- Language Detection

Preprocessing

How might preprocessing techniques vary for different languages?