|  |  |
| --- | --- |
| **Department of Computer Engineering Class: B.E. (Computer) (Div- 8) (Sem-VIII)** | |
| **Subject: Computational Lab-II(NLP)(CSL804)** | |
| **Sr. No.** | **Title of Experiment** |
| **3.** | **To perform Morphological Analysis of any given text.** |

|  |
| --- |
| **Course Name: Computational Lab-II(NLP)** |
| **Course Code: CSL804** |
| **Experiment No.: 3** |
| **Lab Outcome: Have a broad understanding of the field of natural language processing** |
| **Name of Student: Moin Memon** |
| **Student Roll No.: 275** |
| **Year/Semester/Div: B.E./VIII/ B** |

**Experiment - 3**

**Title :** To perform Morphological Analysis of any given text.

**Theory :**

# SpaCy

spaCy is a free, open-source library for NLP in Python. It’s written in [Cython](https://cython.org/) and is designed to build information extraction or natural language understanding systems.

It’s built for production use and provides a concise and user-friendly API. **How to Install spaCy**

spaCy can be installed using **pip**, a Python package manager. You can use a **virtual environment** to avoid depending on system-wide packages.

Create a new virtual environment:

$ python3 -m venv env

Activate this virtual environment and install spaCy:

$ source ./env/bin/activate$ pip install spacy

# How to Download Models and Data

spaCy has [different types](https://spacy.io/models) of models. The default model for the English language is en\_core\_web\_sm.

Activate the virtual environment created in the previous step and download models and data for the English language:

$

python

-

m spacy download en\_core\_web\_sm

## Using spaCy

Load the language model instance in spaCy:

>>>

import

spacy

>>>

nlp

=

spacy

.

load(

'en\_core\_web\_sm'

)

Here, the nlp object is a language model instance. You can assume that, throughout this tutorial, nlp refers to the language model loaded by en\_core\_web\_sm. Now you can use spaCy to read a string or a text file.

# How to Read a String

You can use spaCy to create a processed [Doc](https://spacy.io/api/doc) object, which is a container for accessing linguistic annotations, for a given input string:

>>> introduction\_text = ('This tutorial is about Natural Language

Processing in Spacy.'

)

>>>

introduction\_doc

=

nlp(introduction\_text)

## Sentence Detection

**Sentence Detection** is the process of locating the start and end of sentences in a given text. This allows you to you divide a text into linguistically meaningful units. You’ll use these units when you’re processing your text to perform tasks such as **part of speech tagging** and **entity extraction**.

In spaCy, the sents property is used to extract sentences. Here’s how you would extract the total number of sentences and the sentences for a given input text:

>>> about\_text = ('Gus Proto is a Python developer currently working for a London-based Fintech company. He is interested in learning

Natural Language Processing.')

>>> about\_doc = nlp(about\_text) >>> sentences = list(about\_doc.sents)

>>> len(sentences)

2

>>> for sentence in sentences:

print (sentence)

Output :

'Gus Proto is a Python developer currently working for aLondon-based

Fintech company.'

'He is interested in learning Natural Language Processing.'

In the above example, spaCy is correctly able to identify sentences in the English language, using a full stop(.) as the sentence delimiter.

## Tokenization in spaCy

**Tokenization** is the next step after sentence detection. It allows you to identify the basic units in your text. These basic units are called **tokens**. Tokenization is useful because it breaks a text into meaningful units. These units are used for further analysis, like part of speech tagging.

In spaCy, you can print tokens by iterating on the Doc object:

about\_text

=

(

'Gus Proto is a Python developer

’

)

>>>

about\_doc

=

nlp(about\_text)

>>>

for

token

in

about\_doc:

print

(

token,

token

.

idx)

Output

:

Gus 0

Proto 4

is 10

a 13

Python 15

developer 22

## Lemmatization

**Lemmatization** is the process of reducing inflected forms of a word while still ensuring that the reduced form belongs to the language. This reduced form or root word is called a **lemma**.

For example, organizes, organized and organizing are all forms of organize. Here, organize is the lemma. The inflection of a word allows you to express different grammatical categories like tense (organized vs organize), number (trains vs train), and so on. Lemmatization is necessary because it helps you reduce the inflected forms of a word so that they can be analyzed as a single item. It can also help you **normalize** the text.

spaCy has the attribute lemma\_ on the Token class. This attribute has the lemmatized form of a token:

|  |
| --- |
| >>> conference\_help\_text = ('Gus is helping organize a developer conference on Applications of Natural Language Processing.')  >>> conference\_help\_doc = nlp(conference\_help\_text) >>> for token in conference\_help\_doc: print (token, token.lemma\_)    Output : Gus Gus is be helping help organize organize a a |
|

developer developer conference conference on on

Applications Applications of of

Natural Natural

Language Language Processing Processing

## Part of Speech Tagging

**Part of speech** or **POS** is a grammatical role that explains how a particular word is used in a sentence. There are eight parts of speech:

1. Noun
2. Pronoun
3. Adjective
4. Verb
5. Adverb
6. Preposition
7. Conjunction
8. Interjection

**Part of speech tagging** is the process of assigning a **POS tag** to each token depending on its usage in the sentence. POS tags are useful for assigning a syntactic category like **noun** or **verb** to each word.

In spaCy, POS tags are available as an attribute on the Token object:

about\_text

=

(

'Gus Proto is a Python developer

’

)

>>>

about\_doc

=

nlp(about\_text)

>>>

for

token

in

about\_doc:

print

token,

(

token

.

tag\_,

token

.

pos\_,

spacy

.

explain(token

.

tag\_))

Output

:

Gus NNP PROPN noun, proper singular

Proto NNP PROPN noun, proper singular

is VBZ VERB verb, 3rd person singular present

a DT DET

determiner

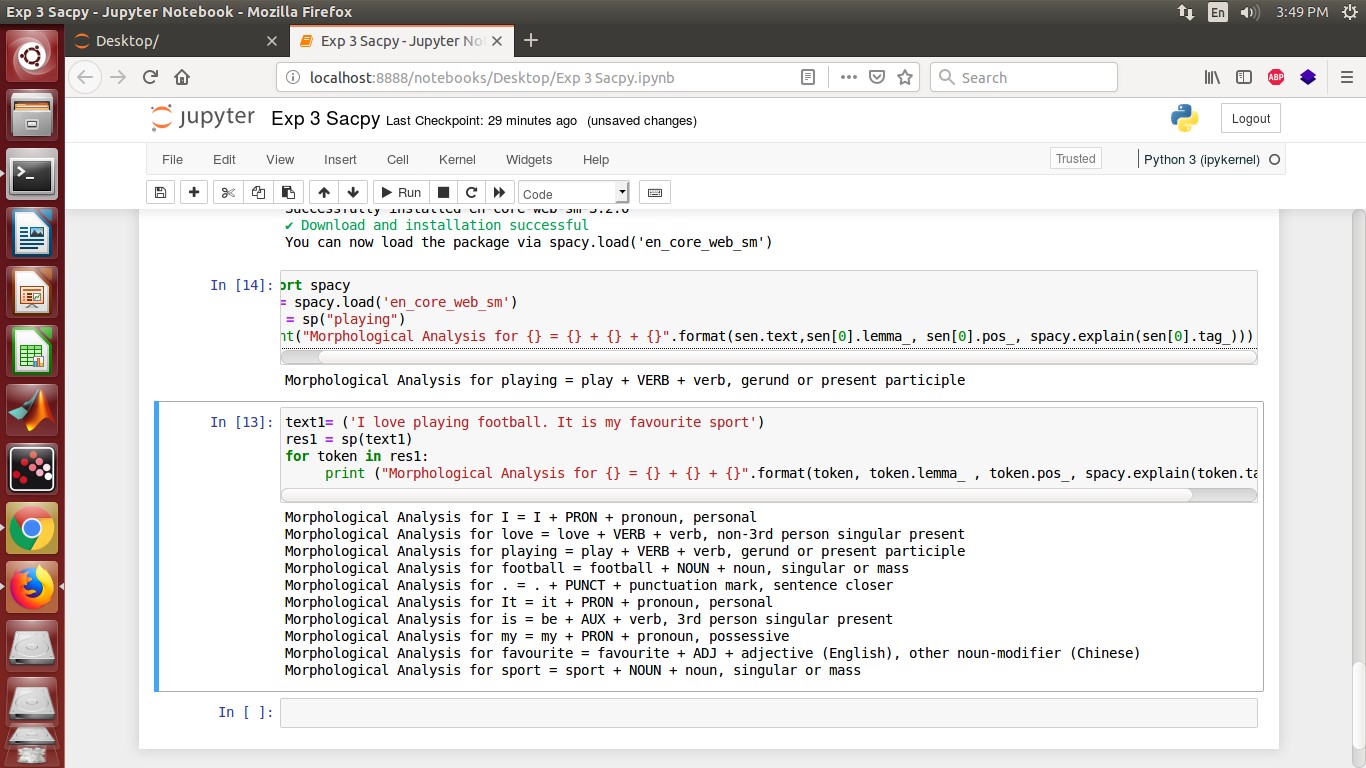
Python NNP PROPN noun, proper singular

developer NN NOUN noun, singular or mass

Here, two attributes of the Token class are accessed:

1. **tag\_** lists the fine-grained part of speech.
2. **pos\_** lists the coarse-grained part of speech.

**Output :**



**Conclusion:**

We have successfully performed Morphological Analysis of any given text using SpaCy.