Année Universitaire : 2024/2025 Master 2 : SII Module : TALN	Université des Sciences et de la Technologie Houari Boumediene Faculté d'Informatique Département d'Intelligence Artificielle et Sciences des Données	**TP N°1** Text Preprocessing

I. Text Manipulation in Python

Target	Function	Python shell	Outputs

Print a **Text**	`print()`	`Text = "Hello World!"`
		`print(Text)`
			Hello World!

Return the first character of a **Text**		`Text = "Hello World!"`
		`print(Text[0])`
			H

Return a substring from a **Text**		`Text = "Hello World!"`
		`print(Text[3:8])`
			lo wo

Return a substring from a **Text** starting at the beginning		`Text = "Hello World!"`
		`print(Text[:5])`
			Hello

Return a substring from a given position to the end of a **Text**		`Text = "Hello World!"`
		`print(Text[6:])`
			World!

Return the length of a **Text**	`len()`	`Text = "Hello World!"`
		`print(len(Text))`
			12

Return the number of occurrences of a specific character in a **Text**	`count()`	`Text = "Hello World!"`
		`print(Text.count("o"))`
			2

Split a **Text**	`split()`	`Text = "Hello World!"`
		`print(Text.split())`
			['Hello', 'World!']

Return the starting index of a substring in a **Text**	`find()`	`Text = "Hello World!"`
		`print(Text.find("W"))`
		`print(Text.find("o"))`
		`print(Text.find("SII"))`
			6
			4
			-1

Return the starting index of a substring in a **Text**	`index()`	`Text = "Hello World!"`
		`print(Text.index("H"))`
		`print(Text.index("d!"))`
			1
			10

Convert all characters in a **Text** to uppercase	`upper()`	`Text = "Hello World!"`
		`print(Text.upper())`
			HELLO WORLD!

Convert all characters in a **Text** to lowercase	`lower()`	`Text = "Hello World!"`
		`print(Text.lower())`
			hello world!

Capitalize the first letter of every word in a **Text**	`title()`	`Text = "hello world!"`
		`print(Text.title())`
			Hello World!

Capitalize only the first letter of a **Text**	`capitalize()`	`Text = "hello world!"`
		`print(Text.capitalize())`
			Hello world!

Change the case of each letter in a **Text**	`swapcase()`	`Text = "Hello World!"`
		`print(Text.swapcase())`
			hELLO wORLD

Target	Function	Python shell	Outputs

Add a substring to a **Text**		`Text = "Hello World!"`
		`print(Text+ " :)")`
			Hello World! :)

Multiply a **Text**		`Text = "Hello World! "`
		`print(Text*4)`
			Hello World! Hello World! Hello World! Hello World!

Reverse a **Text**	**`reversed()`**	`Text = "Hello World!"`
		`print(''.join(reversed(Text)))`
			!dlroW olleH

Add a whitespace between every character in a **Text**	**`join()`**	`Text = "Hello World!"`
		`print(" ".join(Text))`
			H e l l o W o r l d !

Check if all characters in a **Text** are alphanumeric	**`isalnum()`**	`Text = "Hello World!"`
		`print(Text.isalnum())`
			False

Check if all characters in a **Text** are alphabetic	**`isalpha()`**	`Text = "Hello World!"`
		`print(Text.isalpha())`
			False

Check if all characters in a **Text** are digits	**`isdigit()`**	`Text = "Hello World!"`
		`print(Text.isdigit())`
			False

Check if the first letter of every word in a **Text** is capitalized	**`istitle()`**	`Text = "Hello World!"`
		`print(Text.istitle())`
			True

Check if a **Text** is in uppercase	**`isupper()`**	`Text = "Hello World!"`
		`print(Text.isupper())`
			False

Check if a **Text** is in lowercase	**`islower()`**	`Text = "Hello World!"`
		`print(Text.lower())`
			False

Check if a **Text** starts with a specific character	**`startswith()`**	`Text = "Hello World!"`
		`print(Text.startswith('H'))`
			True

Check if a **Text** ends with a specific character	**`endswith()`**	`Text = "Hello World!"`
		`print(Text.endswith('!'))`
			True

Replace a substring in a **Text**	**`repalce()`**	`Text = "Hello World!"`
		`print(Text.replace("Wo","SII"))`
			Hello SIIrld

Remove leading and trailing whitespace or blank lines in a **Text**	**`strip()`**	`Text = "   Hello World!   "`
		`print(Text.replace())`
			Hello World!

II. Text Extraction

Extracting Text from a TXT file

```
txt = open("Name.txt", "r")
for line in txt:
    print(line)
             txt.close()
```

Extracting text from a DOC file

```
#Install library python-docx==1.2.0

import docx
from docx import Document

doc = open ("Name.docx", "rb")
document = Document(doc)

#Extracting text from a document word
doc_text = ""
for para in document.paragraphs:
    doc_text+= para.text

print(doc_text)
doc.close()
```

Extracting Text from a PDF file

```
#Install library PyPDF2==3.0.1
#PyPDF2: PDF manipulation library
from PyPDF2 import PdfReader

#Open PDF file
pdf = open("Name.pdf", "rb")

#PdfReader: read and interpret PDF files
#Creating PDF reader object
PDFreader = PdfReader(pdf)

# PDFreader.pages: list of the document's pages
# Fetching the number of pages in the PDF
num_pages = len(PDFreader.pages)
print(num_pages)

if num_pages > 0:
     #Selecting page 0
     page = PDFreader.pages[0]

     #extract_text(): extract text from a page
     text = page.extract_text()
     print(text)

 pdf.close()
```

Extracting text from a SCANNED document

```
#Install library pytesseract==0.3.10
#pytesseract: OCR manipulation library

#Install library opencv-python==4.10.0
#opencv-python(cv2): image manipulation library

import cv2
from pytesseract import image_to_string

filename = "Name.png"

#imeread(): read an image
img = cv2.imread(filename)

#image_to_text(): extract text from an image
text = image_to_string(img,lang ='eng')
print (text)
```

Extracting Text from a HTML page (web scraping)

```
#Install library BeautifulSoup4==4.13.5
#BeautifulSoup: parse HTML (analyze and extract data from web pages.

from bs4 import BeautifulSoup
from urllib.request import urlopen

myurl = "https://link.springer.com/article/10.1007/s12065-022-00794-z"

html = urlopen(myurl).read()

#Parse and convert the entire HTML page into a tree structure
soup = BeautifulSoup(html, "html.parser")

#Searches inside the parsed HTML:
#for a <div> tag that has the class c-article-body
text = soup.find("div",{"class":"c-article-body"})

#Extract all readable text inside the <div> (removes the HTML tags)
print (text.get_text())

abstract = soup.find("div",{"class":"c-article-section__content"})
print(abstract.get_text())
```

III. Text Preprocessing

NLTK, **spaCy** and **TextBlob** are commonly used **Natural Language Processing (NLP)** libraries in Python for text processing.

III. 1. Installation

1| NLTK library installation

NLTK	**NLTK** (Natural Language Toolkit) is a **NLP library** in Python It provides tools for **text preprocessing**, such as tokenization, stemming and lemmatization.

How to Use NLTK library for Text Processing

Steps	From	Commands

Step 1 — Install the NLTK Library	Terminal	`#Recommended: NLTK==3.9.2 (requires Python <= 3.11)`
		`pip install nltk`

Step 2 — Launch the NLTK Linguistic Data Downloader	Python Shell	`import nltk`
		`nltk.download()`

Step 3 — Download and Install `punkt` **model** Needed for word and sentence segmentation		
Step 4 — Download and Install `punkt_tab` **model** Needed for word and sentence segmentation		
Step 5 — Download and Install `stopwords` **corpus** Needed for stop word removal		

2| spaCy library installation

spaCy	**spaCy** is a **NLP library** in Python. It offers powerful functionalities, including: **Part-of-Speech (POS) tagging**: identifies the **grammatical category** of **each word in a text**, and **Named Entity Recognition (NER)**: detects and classifies proper nouns such as names of people, organizations, etc. **spaCy Language Model (LM)** allow performing **NLP tasks** such as **Tokenization**, **POS tagging** and **NER**.

How to Use spaCy library for Text Processing

Steps	From	Commands

Step 1 — Install the spaCy Library	Terminal	`#Recommended: spaCy==3.7.2 (requires Python <= 3.11)`
		`pip install spacy`

Step 2 — Download the English Language Model	Terminal	`spacy download en_core_web_sm`

Step 3 — Load the English Language Model	Python Shell	`import spacy`
		`nlp = spacy.load('en_core_web_sm')`

Step 4 — Process Text Using the spaCy NLP Pipeline[1]	Python Shell	`doc = nlp('U.S.T.H.B. is a University in Algeria.')`

3| TextBlob library installation

TextBlob	TextBlob is a Python library for processing textual data that provides simple APIs for common NLP tasks such as tokenization, part-of-speech tagging, sentiment analysis, and text classification.

How to Use spaCy library for Text Processing

Steps	From	Commands

Step 1 — Install the TextBlob Library	Terminal	`#Recommended: TextBlob==0.19.0 (Python <= 3.11)`
		`pip install textblob`

[1] The NLP pipeline in spaCy performs a sequence of text processing steps, starting with word segmentation (tokenization).

III.2. Text Preprocessing with NLTK

1| Word Segmentation (Tokenization)

Description	Python Shell	Outputs

Split the text into meaningful tokens using **word_tokenize**	`import nltk`
	`Text = "U.S.T.H.B. is a University in Algeria."`
	`Text = Text.lower() #Lowercasing before tokenizing`
	`Text = nltk.word_tokenize(Text)`
	`for token in Text:`
	`     print(token)`
		u.s.t.h.b
		.
		is
		a
		university
		in
		algeria
		.

2| Sentence segmentation

Description	Python Shell	Outputs

Split the text into multiple sentences using **sent_tokenize**	`import nltk`
	`Text = "Do you know? U.S.T.H.B. is a University."`
	`Text = Text.lower()`
	`Text = nltk.sent_tokenize(Text)`
	`for sent in Text:`
	`     print(sent)`
		do you know?
		u.s.t.h.b.
		is a university.

3| Word and Sentence segmentation using Regexp

<table>
<tr><th>Description</th><th>Python Shell</th><th>Outputs</th></tr>
<tr><td>Split text into tokens or sentences, depending on the desired criteria, using <b>regexp</b></td><td>import nltk

Text = "U.S.T.H.B. is a University in Algeria."
ExpReg = nltk.RegexpTokenizer('\w+') #w > [a-zA-Z0-9_]
Text = nltk.ExpReg.tokenize(Text)
for sent in Text:
     print(sent)

Text = " U.S.T.H.B. is a University in Algeria."
ExpReg = nltk.RegexpTokenizer('(?:[A-Z]\.)+|\w+')
Text = nltk.ExpReg.tokenize(Text)
for sent in Text:
     print(sent)</td><td>U
S
T
H
B
is
a
University
in
Algeria

U.S.T.H.B.
is
a
University
in
Algeria</td></tr>
</table>

4| Word Normalization

Two popular techniques for achieving word **normalization** are **stemming** and **lemmatization**. Stemming refers to the process of removing suffixes and reducing a word to its base or root form so that all its variants can be represented by the same form. Lemmatization, on the other hand, maps all the different forms of a word to its dictionary base form, or lemma. While this may seem similar to stemming, lemmatization requires deeper linguistic knowledge and modeling.

Description	Python Shell	Outputs

Word **Stemming**
(Porter stemmer)
	`import nltk`
	`stemmer = nltk.PorterStemmer()`
	`Text = "U.S.T.H.B. is a University in Algeria."`
	`Text = Text.lower()`
	`Text = nltk.word_tokenize(Text)`
	`for token in Text:`
	`    print(token, stemmer.stem(token))`
		u.s.t.h.b **u.s.t.h.b**
		. **.**
		is **is**
		a **a**
		university **univers**
		in **in**
		algeria **algeria**
		. **.**

Description	Python Shell	Outputs

Word **Stemming**
(Lancaster stemmer)
	`import nltk`
	`stemmer = nltk.LancasterStemmer()`
	`Text = "U.S.T.H.B. is a University in Algeria."`
	`Text = Text.lower()`
	`Text = nltk.word_tokenize(Text)`
	`for token in Text:`
	`    print(token, stemmer.stem(token))`
		u.s.t.h.b **u.s.t.h.b**
		. **.**
		is **is**
		a **a**
		university **univers**
		in **in**
		algeria **alger**
		. **.**

Description	Python Shell	Outputs

Word **Stemming**
(Snowball stemmer)
	`import nltk`
	`stemmer = nltk.SnowballStemmer("english")`
	`Text = "U.S.T.H.B. is a University in Algeria."`
	`Text = Text.lower()`
	`Text = nltk.word_tokenize(Text)`
	`for token in Text:`
	`    print(token, stemmer.stem(token))`
		u.s.t.h.b **u.s.t.h.b**
		. **.**
		is **is**
		a **a**
		university **univers**
		in **in**
		algeria **algeria**
		. **.**

Description	Python Shell	Outputs

Word **Lemmatization**	import nltk
	lemma = **nltk.WordNetLemmatizer()**
	Text = "U.S.T.H.B. is a University in Algeria."
	Text = Text.lower()
	Text = nltk.word_tokenize(Text)
	
	for token **in** Text:
	    # Assume the POS tag for all tokens is "v" (verb).
	    # Use nltk.pos_tag() to get real POS tags.
	    # POS tagging assigns each token a grammatical
	    # category such as noun, verb, or adjective).
	    print(token, lemma.**lemmatize(token, pos="v")**)
		u.s.t.h.b **u.s.t.h.b**
		. .
		is **be**
		a **a**
		university **university**
		in **in**
		algeria **algeria**
		. .

5| Stop Word Removal

In NLP, the words do not always provide useful insights are called stop words. There is no universal stop words list for each language. Usually, it is a list of the most common words in the language, such as of, the, want, to, and have.

Description	Python Shell	Outputs

Check if the token is a stop word or not.	import nltk
	stop = **nltk.corpus.stopwords.words('english')**
	Text = "U.S.T.H.B. is a University in Algeria."
	Text = Text.lower()
	Text = nltk.word_tokenize(Text)
	
	for token in Text:
	    if token in stop:
	        print(token, True)
	    else:
	        print(token, False)
		u.s.t.h.b **False**
		. **False**
		is **True**
		a **True**
		university **False**
		in **True**
		algeria **False**
		. **False**

III.2. Text Preprocessing and with spaCy

1| Word Segmentation (Tokenization)

Description	Python Shell	Outputs

The **first step** in the **spaCy NLP pipeline** is **Tokenization**. spaCy uses **rule-based tokenization**, relying on **language-specific rules** to extract tokens.	`import spacy`
	`nlp = spacy.load('en_core_web_sm')`
	`Text = nlp('U.S.T.H.B. is a University in Algeria.')`
	`for token in Text:`
	`     print(token.text)`
		U.S.T.H.B.
		is
		a
		University
		in
		Algeria
		.

2| Part-of-Speech (POS) tagging

Description	Python Shell	Outputs

Another step in the **spaCy NLP** pipeline is **POS tagging**. This process is **model-based**, meaning it relies on statistical, machine learning (ML), or deep learning (DL) models to predict the correct tags.	`import spacy`
	`nlp = spacy.load('en_core_web_sm')`
	`Text = nlp('U.S.T.H.B. is a University in Algeria.')`
	`for token in Text:`
	`     print(token.text, token.pos_)`
		U.S.T.H.B. **PROPN**
		is **AUX**
		a **DET**
		University **PROPN**
		in **ADP**
		Algeria **PROPN**
		. **PUNCT**

3| Named Entity Recognition (NER)

Description	Python Shell	Outputs

Another key step in the **spaCy NLP pipeline** is **NER tagging**, which identifies and classifies named entities in text (such as people, organizations, locations). This process is **model-based**, meaning it relies on statistical, ML, or DL models to predict the correct tags.	`import spacy`
	`nlp = spacy.load('en_core_web_sm')`
	`Text = nlp('U.S.T.H.B. is a University in Algeria.')`
	`for ent in Text.ents:`
	`     print(ent.text, ent.label_)`
		U.S.T.H.B. **GPE**
		Algeria **GPE**

4| Sentence segmentation

Description	Python Shell	Outputs

Split the text into multiple sentences.	import spacy
	nlp = spacy.load('en_core_web_sm')
	Text = nlp('Do you know? U.S.T.H.B. is a University.')
	for sent **in** **Text.sents**:
	     print(sent)
		Do you know?
		U.S.T.H.B. is a University.

5| Word Normalization

Description	Python Shell	Outputs

Word **lemmatization**	import spacy
	nlp = spacy.load('en_core_web_sm')
	doc = nlp('U.S.T.H.B. is a University in Algeria.')
	for token **in** doc:
	     print(token.text, **token.lemma_**)
		U.S.T.H.B. **U.S.T.H.B.**
		is **be**
		a **a**
		University **University**
		in **in**
		Algeria **Algeria**

2| Stop Word Removal

Description	Python Shell	Outputs

Add **custom stop words** to the standard list.	import spacy
	nlp = spacy.load('en_core_web_sm')
	
	my_stop_words = ['say', 'be', 'said', 'says']
	for stopword in my_stop_words:
	     token = nlp.vocab[stopword]
	     token.is_stop = True
	
	doc = nlp("say be said says")
	for token in doc:
	    print(token.text, **token.is_stop**)
		say **True**
		be **True**
		said **True**
		says **True**

Description	Python Shell	Outputs

Add **a stop words** to the list of stop word.	`import spacy`
	`nlp = spacy.load('en_core_web_sm')`
	
	`stop_words_list = spacy.lang.en.stop_words.STOP_WORDS`
	`stop_words_list.add('saying')`	

Description	Python Shell	Outputs

Check if the token is a stop word or not using attribute **is_stop**	`import spacy`
	`nlp = spacy.load('en_core_web_sm')`
	`doc = nlp('U.S.T.H.B. is a University in Algeria.')`
	`for token in doc:`
	`    print(token.text, token.is_stop)`
		U.S.T.H.B. **False**
		is **True**
		a **True**
		University **False**
		in **True**
		Algeria **False**
		. **False**

Overview of spaCy's Built-in Token Attributes and Linguistic Features

Attribute	Description	Attribute	Description
lower_	Lowercase form of the **Token**	is_title	If the first letter of the **Token** is capitalized
upper_	Uppercase form of the **Token**	is_punct	**Token** is punctuation
is_alpha	**Token** consists of alphanumeric chars	is_space	**Token** is whitespaces
is_digit	**Token** consists of digits	like_num	**Token** resembles a number
is_lower	**Token** text is in lowercase	like_url	**Token** resembles an URL
is_upper	**Token** text is in uppercase	like_email	**Token** resembles an email

POS Tags			NER Tags	
Tag	**Full name**	**Example**	**Label**	**Description**
ADJ	Adjective	happy, new	ORG	Organizations (companies, agencies, institutions, etc.)
ADP	Adposition	in, to, during	GPE	Geopolitical entities (countries, cities, states)
ADV	Adverb	quickly, never	LOC	Non-GPE locations (mountain ranges, bodies of water, etc.)
AUX	Auxiliary verb	is, have, do	PRODUCT	Objects, vehicles, foods, etc. (not services)
CCONJ	Coordinating conjunction	and, but, or	EVENT	Named events (wars, sports events, natural disasters)
DET	Determiner	the, a, this	WORK_OF_ART	Titles of books, songs, artworks, etc.
INTJ	Interjection	oh, wow, hey	LAW	Named laws or legal documents
NOUN	Noun	dog, house, computer	LANGUAGE	Named languages
NUM	Numeral	one, 100, third	DATE	Dates or periods (e.g., January, 2020, last year)
PART	Particle	to, not	TIME	Times smaller than a day (e.g., 2 p.m., morning)
PRON	Pronoun	he, she, they, it	PERCENT	Percentage values (e.g., 50%)
PROPN	Proper noun	John, London, Microsoft	MONEY	Monetary values (e.g., $5, €100)
PUNCT	Punctuation	., ?, !	QUANTITY	Measurements (e.g., 10 km, 5 liters)
SCONJ	Subordinating conjunction	because, although	ORDINAL	Ordinal numbers (e.g., first, second)
SYM	Symbol	$, %, ©	CARDINAL	Cardinal numbers (e.g., one, 1000)
VERB	Verb	run, eat, think		
X	Other	foreign words, errors		

III.3. Text Preprocessing with TextBlob

1| Word Segmentation (Tokenization)

Description	Python Shell	Outputs

Split the text into meaningful tokens using **TextBlob().words**	`import textblob`
	`Text = "U.S.T.H.B. is a University in Algeria."`
	`Text = Text.lower() #Lowercasing before tokenizing`
	
	`Text = textblob.TextBlob(Text).words`
	`for token in Text:`
	`     print(token)`
		u.s.t.h.b
		is
		a
		university
		in
		algeria

2| Sentence segmentation

Description	Python Shell	Outputs

Split the text into multiple sentences using **TextBlob().sentences**	`import textblob`
	`Text = "Do you know? U.S.T.H.B. is a University."`
	`Text = Text.lower()`
	`Text = textblob.TextBlob(Text).sentences`
	`for sent in Text:`
	`     print(sent)`
		do you know?
		u.s.t.h.b.
		is a university.

3| Word Normalization

Description	Python Shell	Outputs

Word **Lemmatization**	`import textblob`
	`Text = "U.S.T.H.B. is a University in Algeria."`
	`Text = Text.lower()`
	`Text = textblob.TextBlob(Text).words`
	
	`for token in Text:`
	`    # Assume the POS tag for all tokens is "v" (verb).`
	`    # Use attribute .pos_tags to get real POS tags.`
	`    Lemma = textblob.Word(token).lemmatize(pos="v")`
	`    print(token, Lemma)`
		u.s.t.h.b **u.s.t.h.b**
		is **be**
		a **a**
		university **university**
		in **in**
		algeria **algerias**

4| Spelling corrections

Description	Python Shell	Outputs

Correct spelling errors using **correct()**	`import textblob`
	`Text = "Doo youu kwnow? U.S.T.H.B. ies a University."`
	`Text = Text.lower()`
	`print(Text)`
	
	`Text = textblob.TextBlob(Text).`**`correct()`**
	`print(Text)`
		`Doo youu kwnow? U.S.T.H.B. ies a Univercity."`
		
		`do you know? u.s.t.h.b. is a university.`

IV. Distance Metrics

Many NLP applications involve tasks such as computing the similarity between two pieces of text. This can be done at different levels: word, phrase, sentence, or document. The goal may be to identify either syntactic or semantic similarity. Similarity measures can also be applied in various contexts; one such context is character-level similarity, which examines how different two strings are based on their characters. Popular approaches for measuring this difference include the Levenshtein edit distance and Jaro similarity methods. **Textdistance** is a popular Python library for computing such measures.

Description	Python Shell	Outputs

Levenshtein Edit Distance
It measures the minimum number of operations required to transform one string into another. The allowed operations include insertion, deletion, or substitution of a character.
	`import textdistance`
	`print(textdistance.levenshtein('Algeria', 'Algiers'))`
		3

Jaro similarity
It measures string similarity based on matching characters and their order, yielding a score from 0 (no match) to 1 (exact match).
	`import textdistance`
	`print(textdistance.jaro('Algeria', 'Algiers'))`
		0.8492063492063492

V. Text Visualization

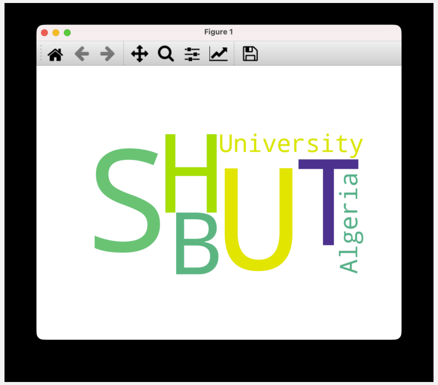
Description

WordCloud is the most popular Python library for representing text. The WordCloud library allows the generation of visualizations from a body of text, where the frequency of words or phrases is reflected by their size and opacity in the plot.

Python Shell

```
# Import the matplotlib library for creating plots
import matplotlib

# Use the Qt5 graphical interface
matplotlib.use('Qt5Agg')

# pyplot provides functions for plotting
import matplotlib.pyplot as plt

# To generate a word cloud visualization from text
from wordcloud import WordCloud

Text = "U.S.T.H.B. is a University in Algeria."

wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(Text)

# Display the word cloud image
plt.imshow(wordcloud.to_array(),
interpolation='bilinear')

# Remove the axis labels for a cleaner visualization
plt.axis('off')

# Display the plot
plt.show()
```

Outputs



Année Universitaire : 2024/2025 Master 2 : SII Module : TALN	Université des Sciences et de la Technologie Houari Boumediene Faculté d'Informatique Département d'Intelligence Artificielle et Sciences des Données	**TP N°1** Text Preprocessing

Exercise:

Text Extraction

From the following link, extract the text (title and abstract) of each scientific article, then save it in a text file named $\mathbf{D} < \boldsymbol{i} >\mathbf{.txt}$, where $1 \leq i \leq \mathbf{N}$, with $\mathbf{N}$ being the number of scientific articles.

https://link.springer.com/journal/12065/volumes-and-issues/18-5

Text Preprocessing

- Create a file **T.txt** containing all the **tokens** of the extracted scientific articles, using the **Regex** function from **NLTK**.
- Create a file **T_N.txt** containing all the **normalized tokens** of the extracted scientific articles, using the **Snowball stemming** function from **NLTK**. Tokens must be converted to **lowercase** first.
- Create a file **V.txt** containing the **vocabulary** (set of unique words) of all the extracted scientific articles, using the **Regex** function from **NLTK**.
- Create a file **V_N.txt** containing the **vocabulary** (set of unique normalized words) of all the extracted scientific articles, using the **Regex** function from **NLTK**.
- Create a file **S.txt** containing all the **sentences** of the extracted scientific articles, using the **Regex** function from **NLTK**.

Data Visualization

Using the **WordCloud** library, visualize the content of the files **T.txt** and **T_N.txt**.

Edit Distance

Implement the algorithm that computes the **minimum edit distance** (or **Levenshtein distance**) between two strings, as covered in class.

References

[1] Bhargav Srinivasa-Desikan. Natural Language Processing and Computational Linguistics (2018)

[2] Delip Rao and Brian McMahan. Natural Language Processing with PyTorch - Build Intelligent Language Applications Using Deep Learning (2019)

[3] Jyotika Singh. Natural Language Processing in the Real World. Text Processing, Analytics, and Classification (2023)

[4] Rosario Moscato, et al. Natural Language Processing Cookbook - Step-by-step practical solution for unlocking the power of natural language processing potential (2025)

[5] Sowmya Vajjala, et al. Practical Natural Language Processing - A Comprehensive Guide to Building Real-World NLP Systems (2020)