Lab1. Data preprocessing and Extract information

[redacted]

Activity 1. Regular Expression

- Khai báo thư viện re (Regular Expression).

```
import re
```

- Tìm kiếm string với từ bắt đầu và kết thúc được khai báo.

```
txt = "The rain in Spain"
x = re.search("^The.*Spain$", txt)
print(x)
```

- Tìm kiếm tất cả các kết quả phù hợp.

```
txt = "The rain in Spain"
x = re.findall("in", txt)
print(x)
```

- Tìm kiếm ký tự khoảng trắng đầu tiên trong chuỗi.

```
txt = "The rain in Spain"
x = re.split("\s", txt)
print(x)
```

- Tách chuỗi với số lượng tách tối đa là 2 lần - tương ứng với 2 dấu cách đầu tiên.

```
txt = "The rain in Spain"
x = re.split("\s", txt, 2)
print(x)
```

- Thay thế ký tự trong string.

```
txt = "The rain in Spain"
x = re.sub("\s", '_', txt, 2)
print(x)
```

\

- **Chạy và giải thích 3 đoạn code sau!**

```
txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.string)
```

```
txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.group())
```

```
txt = "The rain in Spain"
x = re.search(r"\bS\w+", txt)
print(x.span())
```

- Một số biểu thức chính quy thường được sử dụng:

. (Dấu chấm):
  - Tương đương với bất kỳ ký tự nào, ngoại trừ ký tự xuống dòng (\n).
  - Ví dụ: a.b sẽ tìm kiếm chuỗi như "aab", "acb", "axb",...

* (Asterisk):
  - Ký tự trước nó có thể xuất hiện 0 hoặc nhiều lần.
  - Ví dụ: ab*c sẽ tìm kiếm chuỗi như "ac", "abc", "abbc", "abbbc",...

+ (Dấu cộng):
  - Ký tự trước nó phải xuất hiện ít nhất một lần.
  - Ví dụ: ab+c sẽ tìm kiếm chuỗi như "abc", "abbc", "abbbc",...

? (Dấu hỏi):
  - Ký tự trước nó có thể xuất hiện hoặc không.
  - Ví dụ: ab?c sẽ tìm kiếm chuỗi như "ac", "abc".

[] (Dấu ngoặc vuông):
  - Được sử dụng để chỉ định một tập hợp các ký tự có thể khớp.
  - Ví dụ: [aeiou] sẽ khớp bất kỳ nguyên âm nào.

() (Dấu ngoặc tròn):
  - Tạo một nhóm kết hợp các toán tử khác để tạo một phần biểu thức chính quy.
  - Ví dụ: (ab)+ sẽ tìm kiếm chuỗi như "ab", "abab", "ababab",...

| (Dấu gạch đứng):
  - Hoạt động như một phép OR logic, khớp một trong các biểu thức ở hai bên dấu |.
  - Ví dụ: a|b sẽ tìm kiếm chuỗi chứa "a" hoặc "b".

\ (Dấu gạch chéo ngược):
  - Được sử dụng để xác định các ký tự đặc biệt hoặc áp dụng các quy tắc đặc biệt.
  - Ví dụ: \d khớp với bất kỳ chữ số nào, \s khớp với khoảng trắng, \\ khớp với dấu gạch chéo ngược thực sự.

{} (Dấu ngoặc nhọn):
  - Xác định số lần xuất hiện của một biểu thức cụ thể.

- Ví dụ: a{2} sẽ tìm kiếm chuỗi như "aa", a{2,4} sẽ tìm kiếm chuỗi có 2 đến 4 ký tự "a".

^ (Dấu mũ):

- Khớp với đầu chuỗi.
- Ví dụ: ^abc sẽ tìm kiếm chuỗi bắt đầu bằng "abc".

$ (Dấu đô la):

- Khớp với cuối chuỗi.
- Ví dụ: xyz$ sẽ tìm kiếm chuỗi kết thúc bằng "xyz".

\b (Dấu word boundary):

- Khớp với ranh giới từ (word boundary), giúp xác định các từ riêng lẻ trong văn bản.
- Ví dụ: \bword\b sẽ tìm kiếm từ "word" đứng một mình.

(tìm hiểu thêm các biểu thức khác!)

Activity 2. Json file.

- Khai báo thư viện json.

```
import json
```

- Khởi tạo một json đơn giản bằng python.

```
json_data = '{"python": 1, "php": 2, "c": 3, "java": 4, "ruby": 5}'

json_load = (json.loads(json_data))

print(json.dumps(json_load, indent=4))
```

- Tạo một file *json_data.json* có nội dung như sau.

```
{
    "employees": [
        {
            "name": "John Doe",
            "department": "Marketing",
            "place": "Remote"
        },
        {
            "name": "Jane Doe",
            "department": "Software Engineering",
            "place": "Remote"
        },
        {
            "name": "Don Joe",
            "department": "Software Engineering",
            "place": "Office"
        }
    ]
}
```

- Đọc file *json_data.json* .

```
with open('json_data.json', 'r') as file:
    data = json.load(file)

print(json.dumps(data, indent=4))
```

- Trích xuất dữ liệu từ file *json_data.json* .
(Giải thích lại 2 đoạn code sau)

```
for x in data['employees'][0]:
    print(x)
```

```
print(len(data['employees']))
```

- Khởi tạo file *order.json* có nội dung như sau:

```
{
    "orders": [
        {
            "size": "medium",
            "price": 15.67,
            "toppings": [
                "mushrooms",
                "pepperoni",
                "basil"
            ],
            "extra_cheese": false,
            "delivery": true,
            "client": {
                "name": "Jane Doe",
                "phone": null,
                "email": "janedoe@email.com"
            }
        },
        {
            "size": "small",
            "price": 6.54,
            "toppings": null,
            "extra_cheese": true,
            "delivery": false,
            "client": {
                "name": "Foo Jones",
                "phone": "556-342-452",
                "email": null
            }
        }
    ]
}
```

- Đọc file order.json và in ra nội dung.
- In ra thông tin order thứ 2.
- Trích xuất thông tin client có name là “Foo Jones”.

Activity 3. HTML file.

- Khai báo thư viện BeautifulSoup và request.

```
from bs4 import BeautifulSoup as bs

import requests as req
```

- Lấy đường dẫn đến trang web và hiển thị.
(link: https://vnexpress.net/apple-ra-mat-iphone-14-4508267.html)

```
Web = req.get('https://vnexpress.net/apple-ra-mat-iphone-14-4508267

S = bs(Web.text, 'lxml')

print(S.prettify())
```

- Hiển thị tất cả các thẻ của file .html

```
print([tag.name for tag in S.find_all()])
```

- Tìm kiếm thẻ.

```
find_header = S.find('title')
find_header
```

- Tìm thẻ có nội dung cho trước.

```
tag = S.find(id="to_top")
print(tag)
```

- Tìm kiếm tất cả thẻ có cùng tên.

```
findall = S.find_all('a')
for i in findall:
    print(i)
```

- Giải thích đoạn code sau.

```
findall = S.find_all('a')
for i in findall:
    i = i.get_text()
    # i = i.text
    print(i.replace('\n', ''))
```

- Giải thích đoạn code sau.

```
for link in S.find_all('a'):
    print(link.get('title'))
```

- Trích xuất ra tất cả đường dẫn trong thẻ **a ('href').**
- Tìm kiếm tất cả thẻ **span.**
- Lấy thông tin tất cả các thẻ **span**.

Activity 4. NLTK Tokenizer Package

- word_tokenize, TreebankWordTokenizer:

```
import nltk
from nltk.tokenize import word_tokenize, TreebankWordTokenizer

nltk.download('punkt')

text = "I can't believe it's already September. How time flies!"

# Sử dụng word_tokenize
tokens_word_tokenize = word_tokenize(text)
print("word_tokenize:", tokens_word_tokenize)

# Sử dụng TreebankWordTokenizer
tokenizer = TreebankWordTokenizer()
tokens_treebank = tokenizer.tokenize(text)
print("TreebankWordTokenizer:", tokens_treebank)
```

- **RegexpTokenizer** so sánh với **re**:

```
import nltk
from nltk.tokenize import RegexpTokenizer

text = "Hello, world! How are you doing?"

# Tạo một RegexpTokenizer với biểu thức chính quy tách các từ
#dựa trên các ký tự không phải là chữ cái hoặc số
tokenizer = RegexpTokenizer(r'\w+')

tokens = tokenizer.tokenize(text)
print(tokens)
```

```
import re

text = "Hello, world! How are you doing?"

# Sử dụng re.findall để tách các từ dựa trên các ký tự chữ cái hoặc số
tokens = re.findall(r'\w+', text)
print(tokens)
```

- Xử lý hậu tố:

```
#xử lý hậu tố

import nltk
from nltk.stem import PorterStemmer

nltk.download('punkt')

words = ["happiness", "organized", "organization", "happened", "happening"]

stemmer = PorterStemmer()
prefixed_words = [stemmer.stem(word) for word in words]

print(prefixed_words)
```

- Chia động từ về dạng nguyên thể:

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import TreebankWordTokenizer

nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')

text = "He is swimming and running in the park."

tokenizer = TreebankWordTokenizer()
tokens = tokenizer.tokenize(text)

lemmatizer = WordNetLemmatizer()
lemmatized_tokens = [lemmatizer.lemmatize(token, pos='v') if token.endswith('ing') else token for token in tokens]

print(lemmatized_tokens)
```

Activity 5. Natural Language Processing With spaCy

- Word tokenize

```
import spacy

nlp = spacy.load("en_core_web_sm")

text = "This is an example sentence."
doc = nlp(text)

tokens = [token.text for token in doc]
print(tokens)
```

- Index tokens

```
about_doc = nlp(about_text)

for token in about_doc:
    print (token, token.idx)
```

- Sentence detection

```
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)
print(len(sentences))
for sentence in sentences:
    print(f"{sentence[:6]}...")
```

- Some attributes

```
print(f"{'Text with Whitespace':22}"
      f"{'Is Alphanumeric?':15}"
      f"{'Is Punctuation?':18}"
      f"{'Is Stop Word?'}")

for token in about_doc:
    print(
        f"{str(token.text_with_ws):22}"
        f"{str(token.is_alpha):15}"
        f"{str(token.is_punct):18}"
        f"{str(token.is_stop)}"
    )
```

.text_with_ws : in văn bản mã thông báo cùng với bất kỳ dấu cách nào ở cuối, nếu có.
.is_alpha : cho biết mã thông báo có bao gồm các ký tự chữ cái hay không.

.is_punct : cho biết mã thông báo có phải là ký hiệu dấu chấm câu hay không.
.is_stop : cho biết mã thông báo có phải là "**STOP WORD**" hay không ("**STOP WORD**" là một khái niệm trong xử lý ngôn ngữ tự nhiên (NLP) và thường được sử dụng để chỉ những từ phổ biến và thường xuất hiện trong ngôn ngữ, nhưng thường không mang nhiều ý nghĩa hoặc thông tin quan trọng khi xử lý văn bản. Các từ ngừng thường được loại bỏ trong quá trình tiền xử lý dữ liệu văn bản để cải thiện hiệu suất và độ chính xác của các tác vụ NLP.)

Ví dụ về một số từ ngừng trong tiếng Anh:
"a", "an", "the": Các từ quyền lợi chung.
"and", "but", "or": Các từ nối.
"in", "on", "at": Các từ chỉ vị trí.
"is", "am", "are": Các từ động từ "to be".
"this", "that", "these", "those": Các từ chỉ định.

- Stop_word

```
spacy_stopwords = spacy.lang.en.stop_words.STOP_WORDS
print(len(spacy_stopwords))

for stop_word in list(spacy_stopwords)[:10]:
    print(stop_word)
```

- Delete Stop_word

```
custom_about_text = (
    "Gus Proto is a Python developer currently"
    " working for a London-based Fintech"
    " company. He is interested in learning"
    " Natural Language Processing."
)
nlp = spacy.load("en_core_web_sm")
about_doc = nlp(custom_about_text)
print([token for token in about_doc if not token.is_stop])
```

- **Lemmatization**

```
nlp = spacy.load("en_core_web_sm")
conference_help_text = (
    "Gus is helping organize a developer"
    " conference on Applications of Natural Language"
    " Processing. He keeps organizing local Python meetups"
    " and several internal talks at his workplace."
)
conference_help_doc = nlp(conference_help_text)
for token in conference_help_doc:
    if str(token) != str(token.lemma_):
        print(f"{str(token):>20} : {str(token.lemma_)}")
```

- Word frequency

Complete_text:

"Gus Proto is a Python developer currently"
" working for a London-based Fintech company. He is"
" interested in learning Natural Language Processing."
" There is a developer conference happening on 21 July"
' 2019 in London. It is titled "Applications of Natural'
' Language Processing". There is a helpline number'
" available at +44-1234567891. Gus is helping organize it."
" He keeps organizing local Python meetups and several"
" internal talks at his workplace. Gus is also presenting"
' a talk. The talk will introduce the reader about "Use'
' cases of Natural Language Processing in Fintech".'
" Apart from his work, he is very passionate about music."
" Gus is learning to play the Piano. He has enrolled"
" himself in the weekend batch of Great Piano Academy."
" Great Piano Academy is situated in Mayfair or the City"
" of London and has world-class piano instructors."

```
from collections import Counter
nlp = spacy.load("en_core_web_sm")
complete_text = (
    "Gus Proto is a Python developer currently"
    " working for a London-based Fintech company. He is"
    " interested in learning Natural Language Processing."
    " There is a developer conference happening on 21 July"
    ' 2019 in London. It is titled "Applications of Natural'
    ' Language Processing". There is a helpline number'
    " available at +44-1234567891. Gus is helping organize it."
    " He keeps organizing local Python meetups and several"
    " internal talks at his workplace. Gus is also presenting"
    ' a talk. The talk will introduce the reader about "Use'
    ' cases of Natural Language Processing in Fintech".'
    " Apart from his work, he is very passionate about music."
    " Gus is learning to play the Piano. He has enrolled"
    " himself in the weekend batch of Great Piano Academy."
    " Great Piano Academy is situated in Mayfair or the City"
    " of London and has world-class piano instructors."
)
complete_doc = nlp(complete_text)

words = [
    token.text
    for token in complete_doc
    if not token.is_stop and not token.is_punct
]

print(Counter(words).most_common(3))
print('\n')
print(Counter(words))
```

- POS of Tagging (Part-of-speech tagging)

```
import spacy
nlp = spacy.load("en_core_web_sm")
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)
for token in about_doc:
    print(
        f"""
TOKEN: {str(token)}
=====
TAG: {str(token.tag_):10} POS: {token.pos_}
EXPLANATION: {spacy.explain(token.tag_)}"""
    )
```

- Using POS tags, you can extract a particular category of words:

```
nouns = []
adjectives = []
for token in about_doc:
    if token.pos_ == "NOUN":
        nouns.append(token)
    if token.pos_ == "ADJ":
        adjectives.append(token)


print(nouns)
print(adjectives)
```

- Visualization POS:

```
import spacy
from spacy import displacy
nlp = spacy.load("en_core_web_sm")

about_interest_text = (
    "He is interested in learning Natural Language Processing."
)
about_interest_doc = nlp(about_interest_text)
displacy.serve(about_interest_doc, style="dep")
```

- Dependency Parsing (Phân tích cú pháp):

```
import spacy
nlp = spacy.load("en_core_web_sm")
piano_text = "Gus is learning piano"
piano_doc = nlp(piano_text)
for token in piano_doc:
    print(
        f"""
TOKEN: {token.text}
=====
{token.tag_ = }
{token.head.text = }
{token.dep_ = }"""
    )
```

nsubj is the subject of the word, and its headword is a verb.
aux is an auxiliary word, and its headword is a verb.
dobj is the direct object of the verb, and its headword is also a verb.

- Verb Phrase Detection:

```
import textacy

about_talk_text = (
    "The talk will introduce reader about use"
    " cases of Natural Language Processing in"
    " Fintech, making use of"
    " interesting examples along the way."
)

patterns = [{"POS": "AUX"}, {"POS": "VERB"}]
about_talk_doc = textacy.make_spacy_doc(
    about_talk_text, lang="en_core_web_sm"
)
verb_phrases = textacy.extract.token_matches(
    about_talk_doc, patterns=patterns
)

# Print all verb phrases
for chunk in verb_phrases:
    print(chunk.text)



# Extract noun phrase to explain what nouns are involved
for chunk in about_talk_doc.noun_chunks:
    print (chunk)
```

- Named-Entity recognition (NER):

```
import spacy
nlp = spacy.load("en_core_web_sm")

piano_class_text = (
    "Great Piano Academy is situated"
    " in Mayfair or the City of London and has"
    " world-class piano instructors."
)
piano_class_doc = nlp(piano_class_text)

for ent in piano_class_doc.ents:
    print(
        f"""
{ent.text = }
{ent.start_char = }
{ent.end_char = }
{ent.label_ = }
spacy.explain('{ent.label_}') = {spacy.explain(ent.label_)}"""
)
```

- Visualize NER:

```
1 from spacy import displacy
2 displacy.serve(piano_class_doc, style="ent")
```

```
C:\Users\ADMIN\anaconda3\lib\site-packages\spacy\displacy\__init__.py:106: UserWar
placy.serve from within a Jupyter notebook or a similar environment. This likely m
r, so there's no need to make displaCy start another one. Instead, you should be a
ender to show the visualization.
  warnings.warn(Warnings.W011)
```

Bài tập:

1. **Hoàn thành tất cả các phần thực hành từ Act1 - Act5.**
2. **Implement lại thuật toán HMM cho 2 ví dụ trong slide Week-3:**

Give the tagged corpus:

```
(D,the) (N,wine) (V,ages) (A,alone)
(D,the) (N,wine) (N,waits) (V,last) (N,ages)
(D,some) (N,flies) (V,dove) (P,into) (D,the) (N,wine)
(D,the) (N,dove) (V,flies) (P,for) (D,some) (N,flies)
(D,the) (A,last) (N,dove) (V,waits) (A,alone)
```

a: Build the HMM model for POS tagging (unigram)

b: Draw the word lattice of sentence: "the flies waits alone", and find the best POS tag for this sentence.

Corpus:

(The, D) (boy, N) (likes, V) (books, N).

(The, D) (books, N) (are, V) (bank, N) (books.N).

(Some, D) (books, N) (are, V) (on, P) (the, D) (table, N).

(Some, D) (boys, N) (books, V) (the, D) (table, N).

(The, D) (book, N) (has, V) (many,A) (likes, N).

(Many, A) (banks, N) (like, V) (books, N).

- Build the vocabulary of corpus (convert to infinitive form)
- Build the HMM model for POS tagging (unigram and infinitive form of each word)
- Draw the word lattice of sentence: "The likes are the books" and use the Viterbi algorithm to obtain the best POS tagging for this sentence.