Лабораторна робота №2

Попередня обробка тексту за допомогою NLTK

Мета роботи: Ознайомитись з основними операціями з попередньої обробки тексту та їх реалізацією у бібліотеці NLTK.

Короткі теоретичні відомості

Попередня обробка або нормалізація тексту — це процес, який складається з серії кроків для перетворення, очищення та стандартизації текстових даних у форму, яка може бути використана іншими задачами ОПМ та інтелектуальними системами, що працюють на основі машинного та глибокого навчання. Ключова ідея полягає в тому, щоб видалити непотрібний вміст з одного чи кількох текстових документів у корпусі (або корпусах) і отримати чисті текстові документи.

Попередня обробка тексту передбачає широкий спектр методів, які перетворюють необроблений текст у чітко визначені послідовності мовних компонентів, які мають стандартну структуру та позначення. До попередньої обробки можна віднести наступні дії:

- Токенізація
- Видалення непотрібних токенів і стоп-слів
- Усунення скорочень
- Виправлення орфографічних помилок
- Пошук кореня
- Лематизація
- Тегування
- Синтаксичний розбір
- Видалення тегів HTML

NLTK (Natural Language Toolkit) є популярною платформою для створення програм Python для роботи з даними природної мови https://www.nltk.org/ Потрібно встановити дану бібліотеку та завантажити додаткові дані: корпуси, натреновані моделі і т.д.

```
import nltk

nltk.download('all')
```

Токени (лексеми) — це незалежні та мінімальні текстові компоненти, які мають певний синтаксис і семантику. Абзац тексту або текстовий документ складається з кількох компонентів, які можна розбити на речення, фрази та слова. Найпопулярніші методи токенізації включають токенізацію речень і слів, які

використовуються для розбиття текстового документа (або корпусу) на речення та кожного речення на слова. Таким чином, токенізацію можна визначити як процес розбиття або поділу текстових даних на менші та більш значущі компоненти, які називаються токенами.

```
text = ' Some sentences. Maybe two! Or three? No.'

from nltk.tokenize import sent_tokenize

sents = sent_tokenize(text)

punkt_st = nltk.tokenize.PunktSentenceTokenizer()

sents  = punkt_st.tokenize(s)
```

Токенізація слів — це процес поділу або сегментації речень на складові слова. Речення — це набір слів, і за допомогою токенізації ми по суті розбиваємо речення на список слів, які можна використовувати для реконструкції речення. Токенізація слів дійсно важлива в багатьох процесах, особливо в очищенні та нормалізації тексту, де такі операції, як визначення кореня та лематизація, працюють над кожним окремим.

```
s.split()

from nltk.tokenize import word_tokenize

word_tokenize(s)

treebank_wt = nltk.TreebankWordTokenizer()

words = treebank_wt.tokenize(s)

from nltk.tokenize.toktok import ToktokTokenizer

tokenizer = ToktokTokenizer()

words = tokenizer.tokenize("It's a cat!")

from nltk.tokenize import regexp_tokenize

regexp_tokenize(s, pattern='\w+’)

regexp_tokenize(s, pattern='\s+',gaps=True)

from nltk.tokenize import wordpunct_tokenize

wordpunct_tokenize(s)

whitespace_wt = nltk.WhitespaceTokenizer()

words = whitespace_wt.tokenize(s)
```

Корені слів також часто називають основною формою слова, і можливо створювати нові слова, додаючи до них афікси (суфікси та префікси).

Протилежністю до цього процесу є отримання основної форми слова з його відмінюваної форми, тобто пошук кореня.

```
from nltk.stem import PorterStemmer

pst = PorterStemmer()

tokens=regexp_tokenize(data, pattern='\w+')

stemmed=[]

for item in tokens:

    stemmed.append(pst.stem(item))

result = " ".join(stemmed)

from nltk.stem.lancaster import LancasterStemmer

lst = LancasterStemmer()

from nltk.stem.snowball import EnglishStemmer

sst=EnglishStemmer()
```

Процес лематизації дуже схожий на пошук кореня, коли видаляються афікси слова, щоб отримати базову форму слова. Різниця між ними полягає в тому, що коренева основа не завжди може бути лексикографічно правильним словом, тобто її може не бути в словнику, а лема завжди буде присутня у словнику.

```
from nltk.stem import WordNetLemmatizer

wlem = WordNetLemmatizer()

wlem.lemmatize("went",pos='v’)
```

Видалення стоп-слів є одним із найбільш часто використовуваних етапів попередньої обробки в різних програмах ОПМ. Ідея полягає в тому, щоб просто видалити слова, які часто зустрічаються в усіх документах у корпусі, але не несуть інформації. Як правило, артиклі та займенники класифікуються як стоп-слова.

```
from nltk.corpus import stopwords

stoplist = stopwords.words('english')

text = "This is a cat"

cleanwordlist = [word for word in text.split() if word not in stoplist]
```

Частини мови — це специфічні лексичні категорії, до яких віднесено слова на основі їхнього синтаксичного контексту та ролі. Основними частинами мови є іменники, дієслова, прикметники та прислівники. Процес класифікації та

маркування тегів ЧМ для слів називаєтсья тегування частин мови (ЧМ-тегування).

В NLTK можна отримати частини мови за допомогою функції pos_tag()

pos_tagged = nltk.pos_tag(nltk.word_tokenize("This is a cat"))

NLTK має свої корпуси текстів.

from nltk.corpus import brown

len(brown.categories())

print(brown.categories())

from nltk.corpus import reuters

len(reuters.categories())

В цілому, корпуси текстів у NLTK можуть мати наступні методи:

- words(): список слів
- sents(): речення, розбиті на слова: список списків слів
- paras(): абзаци: список списків (абзац) списків (речення) слів
- tagged_words(): анотовані слова
- tagged_sents(): анотовані слова, об’єднані в речення
- tagged_paras(): анотовані слова, об’єднані в абзаци
- chunked_sents(): анотовані слова у вигляді дерева
- parsed_sents(): токени у вигляді дерева
- parsed_paras(): абзаци у вигляді дерева
- xml(): дерево xml
- raw(): «сирий» текст

Завдання до лабораторної роботи

Створити програму, яка виконує завдання відповідно до варіанту, використовуючи бібліотеку NLTK.

Оформити звіт. Звіт повинен містити:

— титульний лист;
— код програми;
— результати виконання коду;

Продемонструвати роботу програми та відповісти на питання стосовно теоретичних відомостей та роботи програми.