گزارش پروژه بازیابی اطلاعات

فاز۱ : بخش ۱ - پیش پردازش

- با ذکر مثال شرح دهید که در گام پیش پردازش چه عملیاتی انجام داده اید. همچنین دلیل انجام هر پردازش را ذکر کنید.

۱- برای پیاده سازی این بخش از کتابخانه هضم ، re (Regular Expression), JSON,String استفاده خواهیم کرد پس در قدم اول این کتابخانه هارا Import میکنیم و فایل اطلاعاتی که میخواهیم روی آن پردازش انجام دهیم را آدرس دهی میکنیم:

```
import json
from string import punctuation
import re
from hazm import *

file_path = 'C:/Users/Samin/Desktop/University/Term 7/Information
Retrieval/Project/Data/IR_data_news_12k.json'
```

۲- برای باز کردن فایل از try استفاده میکنیم که در صورت وقوع مشکل امکان مدیریت آن وجود داشته باشد. پس از باز شدن فایل اطلاعات درون آن را در دیکشنری ای به نام دیتا ذخیره میکنیم که اجزای آن خودشان دیکشنری هایی هستند که تیتر، متن و url داده هارا بصورت زیر ذخیره میکنند:

```
try:
    f = open(file_path, 'r', encoding='utf-8')
    data_raw = json.load(f)
    print("File opened successfully!")
    f.close()
except IOError:
    print("Error opening file.")

data = {}
for docID, body in data_raw.items():
    data[docID] = {}
    data[docID]['title'] = body['title']
    data[docID]['content'] = body['content']
    data[docID]['url'] = body['url']
```

پس اطلاعات با این ساختار ذخیره خواهند شد:

```
data = {
    'docID_1': {
        'title': 'Title of Document 1',
        'content': 'This is the content of Document 1.',
        'url': 'http://example.com/document1'
    },
    'docID_2': {
        'title': 'Title of Document 2',
        'content': 'This is the content of Document 2.',
        'url': 'http://example.com/document2'
    },
    # more documents...
}
```

برای مشاهده چگونگی کارکرد کد ما یک مثال میزنیم و یک داده با ساختار تعریف شده ایجاد میکنیم:

```
data_test = {
    'docID_۱': {
        'title': 'Test Dox',
        'content': 'رایج کلمات از بسیاری حاوی که است فارسی ینمونه متن یک این
کار به پردازشپیش توابع کارکرد بررسی برای و باشدمی ... و در با، و، از، به، مثل
رودمی.',
        'url': 'http://example.com/test_document'
    }
}
```

حالا روند پیش پردازش را با ارسال دیتا به تابع preprocess آغاز میکنیم:

```
preprocessed_data = preprocess(data_test)
```

تابع پیش پردازش بصورت زیر است:

```
# preprocessed data dictionary
def preprocess(data):
    # create a new dictionary to store the preprocessed data
    preprocessed_data = {}

    for docID, doc in data.items():
        preprocessed_doc = {}
        print(f'Doc Content: {doc["content"]}')
        pure_content = re.sub(f'[{punctuation}٪.؟×÷»«]+', '', doc['content'])
        print(f'Pure Content : {pure_content}')
        preprocessed_doc['content'] =
to_stem(to_remove_stop_words(to_tokenize(to_normalize(pure_content))))
        preprocessed_doc['url'] = doc['url']
        preprocessed_data[docID] = preprocessed_doc

    return preprocessed_data
```

در این تابع داده ای با ساختاری که تعریف شده بود (دیکشنری ای از دیکشنری هایی که داک آیدی، تیتر و url مشخصی دارند) دریافت میشود و دیکشنری دیگری با همان ساختار برای خروجی تعریف میشود، در این مثال دیکشنری ورودی ما تنها شامل یک دیکشنری شده است که به سراغ آن میرویم در حلقه بالا

دیکشنری ای معادل با دیکشنری ای که حلقه در این دور بر آن تمرکز دارد تعریف میشد و تیتر و url اختصاصی ورودی به آن داده میشود،

اما برای متن آن در قدم اول روند حذف علائم نگارشی انجام خواهد شد که در ادامه کار به آنها نیازی نداریم:

```
pure_content = re.sub(f'[{punctuation}٪.؟×÷»«]+', '', doc['content'])
```

این کار با استفاده از کتابخانه regular expression و punctuation از string انجام میشود که علائم نگارشی مشخصی را از متن حذف میکند، پس از اعمال این بخش خواهیم داشت:

Doc Content: این یک متن نمونه‌ی فارسی است که حاوی بسیاری از کلمات رایج مثل به، از، و، با، در و ... می‌باشد و برای بررسی کارکرد توابع پیش‌پردازش به کار می‌رود.

Pure Content : این یک متن نمونه‌ی فارسی است که حاوی بسیاری از کلمات رایج مثل به از و با در و می‌باشد و برای بررسی کارکرد توابع پیش‌پردازش به کار می‌رود

در قدم بعدی اعمال مراحل پیش پردازش را داریم که اولین آنها نرمالسازی است:

```
preprocessed_doc['content'] =
to_stem(to_remove_stop_words(to_tokenize(to_normalize(pure_content))))
```

برای انجام نرمال سازی تابع to_normalize فراخوانی میشود که بصورت زیر است:

```
# preprocessing functions
def to_normalize(input_text):
    print('Normalization..')
    output_text = Normalizer().normalize(input_text)
    print(f'input : {input_text} ')
    print(f'output : {output_text}')
    return output_text
```

در این تابع متن ورودی را گرفته و با استفاده از Normalizer از کتابخانه هضم نرمالسازی را بر متن ورودی انجام داده و آن را برمیگرداند.

Normalization..

input : این یک متن نمونه‌ی فارسی است که حاوی بسیاری از کلمات رایج مثل به از و با در و می‌باشد و برای بررسی کارکرد توابع پیش‌پردازش به کار می‌رود

output : این یک متن نمونه‌ی فارسی است که حاوی بسیاری از کلمات رایج مثل به از و با در و می‌باشد و برای بررسی کارکرد توابع پیش‌پردازش به کار می‌رود

گام بعدی توکن سازی متن است که با فراخوانی تابع to_tokenize انجام خواهد شد:

```
def to_tokenize(input_text):
    print('Tokenizing...')
    # remove all non-alphanumeric characters from the input_text
    input_text = re.sub(r'[^\w\s]','',input_text)
    # tokenize the cleaned text
    output_text = word_tokenize(input_text)
    print(f'input : {input_text} ')
    print(f'output : {output_text}')
    return output_text
```

در این تابع متن ورودی ابتدا از کاراکتر های non-alphanumeric پاکسازی میشود زیرا مشاهده میشد که پس توکن سازی /u۲۰۰ در توکن ها بوجود می آید که ناشی از کاراکتر های یونیکدیست که نماد whitespace هستند.

در قدم بعدی با استفاده از ماژول word_tokenize از کتابخانه هضم متن ورودی به توکن ها تبدیل میشود و بصورت خروجی فرستاده میشود:

Tokenizing...

input : این یک متن نمونهی فارسی است که حاوی بسیاری از کلمات رایج مثل به از و با در و میباشد و برای بررسی کارکرد توابع پیشپردازش به کار میرود

output : ['این', 'یک', 'متن', 'نمونهی', 'فارسی', 'است', 'که', 'حاوی', 'بسیاری', 'از', 'کلمات', 'رایج', 'مثل', 'به', 'از', 'و', 'با', 'در', 'و', 'میباشد', 'و', 'برای', 'بررسی', 'کارکرد', 'توابع', 'پیشپردازش', 'به', 'کار', 'میرود']

گام بعدی پاک سازی متن از کلمات پرتکرار است که با فراخوانی تابع to_remove_stop_words انجام میشود:

```
def to_remove_stop_words(input_text):
    print('Removing Stop Words..')
    stop_words = stopwords_list()
    output_text = [word for word in input_text if not word in stop_words]
    print(f'input : {input_text} ')
    print(f'output : {output_text}')
    return output_text
```

در این تابع با فراخوانی stopwords_list() که لیستی از کلمات پرتکرار در زبان فارسی است که در کتابخانه هضم تعریف شده و چک کردن اینکه کلمات درون متن در آن هستند یا نه کلمات پرتکرار را حذف و باقی را بصورت خروجی برمیگردانیم:

Removing Stop Words..

input : ['این', 'یک', 'متن', 'نمونهی', 'فارسی', 'است', 'که', 'حاوی', 'بسیاری', 'از', 'کلمات', 'رایج', 'مثل', 'به', 'از', 'و', 'با', 'در', 'و', 'میباشد', 'و', 'برای', 'بررسی', 'کارکرد', 'توابع', 'پیشپردازش', 'به', 'کار', 'میرود']

output : ['متن', 'نمونهی', 'فارسی', 'حاوی', 'کلمات', 'رایج', 'میباشد', 'بررسی', 'کارکرد', 'توابع', 'پیشپردازش', 'کار', 'میرود']

گام بعدی ریشه یابی است که با فراخوانی تابع to_stem انجام میشود:

```
def to_stem(input_text):
    print('Stemming...')
    stemmer = Stemmer()
    output_text = [stemmer.stem(word) for word in input_text]
    print(f'input : {input_text} ')
    print(f'output : {output_text}')
    return output_text
```

در این تابع متن ورودی با استفاده از ماژول Stemmer کتابخانه هضم کلمات موجود در متن را ریشه یابی میکند و متن اصلاح شده را بصورت خروجی برمیگرداند:

Stemming...

input : ['متن', 'نمونهی', 'فارسی', 'حاوی', 'کلمات', 'رایج', 'میباشد', 'بررسی', 'کارکرد', 'توابع', 'پیشپردازش', 'کار', 'میرود']

output : ['متن', 'نمونه', 'فارس', 'حاو', 'کل', 'رایج', 'میباشد', 'بررس', 'کارکرد', 'توابع', 'پیشپرداز', 'کار', 'میرود']

['متن', 'نمونه', 'فارس', 'حاو', 'کل', 'رایج', 'میباشد', 'بررس', 'کارکرد', 'توابع', 'پیشپرداز', 'کار', 'میرود']

حالا که متن دیتا بصورت کامل پیش پردازش شده است دیکشنری دیتایی که ساخته بودیم را به دیکشنری دیتایی که در حال بررسی بودیم نسبت میدهیم:

```
preprocessed_data[docID] = preprocessed_doc
```

در نهایت پس از بررسی دیکشنری های دیتا، دیکشنری نهایی برگردانده میشود.

نمونه داده های اصلی به صورت زیر فراخوانی و پیش پردازش شده و نهایتا در فایلی مجزا ذخیره خواهد شد:

```
# real data preprocessing
preprocessed_data = preprocess(data)
# print(preprocessed_data)

# save preprocessed data as a JSON file
output_file_path = 'C:/Users/Samin/Desktop/University/Term ۷/Information
Retrieval/Project/Data/IR_data_news_۱۲k_preprocessed.json'
with open(output_file_path, 'w', encoding='utf-۸') as f:
    json.dump(preprocessed_data, f, ensure_ascii=False, indent=٤)
```