دانشگاه صنعتی امیرکبیر
( پلی تکنیک تهران )

دانشکده مهندسی کامپیوتر

# گزارش زیربخش اول فاز اول پروژه بازیابی اطلاعات

نگارش
مهدیه سادات بنیس

استاد درس
دکتر احمد نیک آبادی

نیم سال دوم ۱۴۰۱

- **مقدمه:**

در این پروژه میخواهیم بصورت عملی از مفاهیم تدریس‌شده در کلاس درس استفاده کنیم. پروژه در دو فاز تعریف میشود. هدف از انجام این پروژه این است که یک موتور جستجو برای بازیابی اسناد متنی ایجاد کنیم به گونه‌ای که کاربر پرسمان خود را وارد و سامانه اسناد مرتبط را بازنمایی کند.

## ۱. فاز اول:

در این فاز از پروژه به منظور ایجاد یک مدل بازیابی اطلاعات ساده نیاز است تا اسناد شاخص‌گذاری شوند تا در زمان دریافت پرسمان از شاخص مکانی برای بازیابی اسناد مرتبط استفاده شود.

### ۱.۱.مجموعه داده:

مجموعه داده استفاده شده در این پروژه مجموعه‌های از خبرهای واکشی‌شده از چند وبسایت خبری فارسی است در قالب یک فایل json در اختیار داریم. ما تنها محتوای "content" را بعنوان محتوای سند پردازش کنید. شماره ی هر خبر را به عنوان id آن سند (خبر) در نظر بگیرید و در زمان پاسخ به پرسمان، عنوان خبر(title) و URL مربوط به سند بازیابی شده را نمایش میدهیم تا امکان بررسی صحت عملکرد سیستم وجود داشته باشد.

- برای پیاده سازی این قسمت در تابع Load_Docs فایل مربوطه را باز کرده و در خروجی خبر(title) محتوای " content"و URL را برمیگردانیم.

```
def Load_Docs():

    all_docs = {}
    contents = []
    urls = []

    with open("IR_data_news_12k.json", 'r') as f:
        docs = json.load(f)
        for k in docs.keys():
            # index of files
            idx = k + ''
            all_docs[idx] = {'title': docs[idx]['title'],
                             'content': docs[idx]['content'],
                             'url': docs[idx]['url'],
                            }
            contents.append(docs[idx]['content'])
    return all_docs, contents, urls
```

### ۲.۱. پیش‌پردازش اسناد

قبل از ساخت شاخص مکانی لازم است متون را پیش‌پردازش کنیم. گام‌های لازم در این قسمت به صورت زیر می‌باشد.

- نرمال‌سازی متون

هدف این مرحله، تمیز و مرتب کردن متن و یکسان‌سازی کاراکترها با جایگزین کردن کاراکترهای استاندارد در متن ورودی است. در واقع قبل از پردازش متون جهت استانداردسازی حروف و فاصله‌ها بایستی پیش‌پردازش‌هایی روی آنها انجام شود. در واقع در این مرحله بایستی همه‌ی نویسه‌های (حروف) متن با جایگزینی با معادل استاندارد آنها، یکسان‌سازی گردند.

- استخراج توکن

هدف این ابزار شخیص جملات را در متن ورودی است. به عبارت دیگر جداسازی جملات ساده و مرکب (غیرتودرتو) و واژه‌ها و عبارات خاص (توکن‌ها) از یکدیگر هست. پس از پایان مرحله‌ی نرمالسازی متن، ابزار تشخیص‌دهنده‌ی جملات با استفاده از نمادهای (علامت‌های) پایانی جمله از قبیل: ".",";"، "!"، "؟"، ... و بکارگیری برخی علائم، قواعد دستوری زبان و در نظرگرفتن حروف ربط یا

برخی لغات آغاز کننده‌ی جملات مرکب (از قبیل حروف ربط مانند "که"، "تا"، "اما"، "ولی"، "زیرا"، "سپس"، "و"، "یا"، ... در زبان فارسی)، مرز جمله‌ها را تعیین می‌نماید.

- حذف کلمات پر تکرار

منظور از حذف کلمات توقف یا پرتکرار، حذف علائم، اعداد، کلمات عمومی و بدون ارزش معنایی (از قبیل: از، در، با، به، است، پس، ...) در جمله است. در بسیاری از کاربردهای بازیابی اطلاعات، حذف لغات کم‌اهمیت که شاخصه متن نیستند، می‌تواند بدون از بین بردن معنا باعث بهبود دقت و سرعت الگوریتم‌های متن‌کاوی شوند.

- ریشه‌یابی

ریشه‌یابی کلمات یکی از مهمترین عملیات پیش‌پردازش متون در بازیابی اطلاعات و پردازش زبان‌های طبیعی است. هدف الگوریتم‌های ریشه‌یابی، حذف وندهای کلمات (پیشوند و پسوندها) و تعیین ریشه اصلی کلمه، براساس قواعد ساخت واژه‌ای (ریخت‌شناسی)، هستند.

```
class DataPreprocessing:

    #Tokenization
    @staticmethod
    def Tokenization(text):
        return hazm.word_tokenize(text)

    #Normalization
    @staticmethod
    def Normalization(text):
        my_normalizer = hazm.Normalizer()
        return my_normalizer.normalize(text)

    #Stop_Word_Removal
    @staticmethod
    def Stop_Word_Removal(tokens):
        stop_words = hazm.stopwords_list()
        tokens_with_removed_stopwords = []
        for token in tokens:
            if not (token in stop_words):
                tokens_with_removed_stopwords.append(token)
        return tokens_with_removed_stopwords

    #Stemming
    @staticmethod
    def Stemming(tokens):
        stemmed = []
        my_stemmer = parsivar.FindStems()
        for token in tokens:
            stemmed.append(my_stemmer.convert_to_stem(token))
        return stemmed

    #Remove Punctuations
    @staticmethod
    def Remove_Punctuations(text):
        return re.sub(f'[{punctuation}؟،٪×÷»«]+', '', text)

    def preprocess(self, docs, contents):
        for idx, content in enumerate(contents):
            punctuated_content = self.Remove_Punctuations(content)
            normalized_content = self.Normalization(punctuated_content)
            tokens_of_a_sentence = self.Tokenization(normalized_content)
            tokens_of_a_sentence = self.Stop_Word_Removal(tokens_of_a_sentence)
            final_tokens_of_a_sentence = self.Stemming(tokens_of_a_sentence)
            docs[str(idx)]['content'] = final_tokens_of_a_sentence
            if idx % 1000 == 0:
                print(idx)
        return docs
```

استخراج توکن

نرمال سازی متن

حذف کلمات پرتکرار وتوقف

ریشه یابی

حذف علائم نقطه گذاری و غیره

پیاده سازی مراحل انجام متد ها

- برای پیاده سازی این قسمت در کلاس DataPreprocessing متد های مربوط به مراحل ذکر شده را با استفاده از کتاب خانه های parsivar , hazm پیاده سازی میکنیم.

- در پایان نیز فایل پیش پردازش شده را در "IR_data_news_12k_preprocessed.json" ذخیره میکنیم.

```
#Write Preprocessed Docs
with open("IR_data_news_12k_preprocessed.json", 'w') as f:
    json.dump(pre_processed_docs, f)
```