

# Project Report

در این گزارش به جزئیات پیاده سازی پروژه بازیابی اطلاعات می پردازیم.

## جزئیات پیاده سازی :

### ۱. پیش پردازش اسناد:

پیش پردازش هر متن در تابع `tokenize_text` انجام میشود: (شکل ۱)

```
def tokenize_text(self, text , is_indexing= False):  
    tokens = self.tokenizer.tokenize(text)  
    tokens = self.normalizer.normalize_tokens(tokens)  
  
    tokens = [self.stemmer.convert_to_stem(t) for t in tokens]  
  
    if not is_indexing:  
        tokens = [ t for t in tokens if not (t in self.eliminated_words or t == "")]  
  
    return tokens
```

شکل ۱

### ۱.۱. استخراج توکن ها:

استخراج توکن ها در کلاس `WordTokenizer` انجام میشود. (شکل ۲). در کلاس، علاوه بر استخراج توکن ها با استفاده از فاصله اشان، علامت های نگارشی نیز حذف میشوند. در مرحله حذف علائم نگارشی و نشانه گذاری، حذف میشوند. اگر نقطه ای بین دو عدد یا بین دو حرف انگلیسی است، آن را حذف نمیکند. چرا که در این دو حالت نقطه جزوی از متن است. (در حالتی که بین دو حرف انگلیسی باشد، یعنی به ایمیل، یک آی دی و .. است.)

در این مرحله کلمه هایی که با نیم فاصله از هم جدا شده اند را دو توکن در نظر نگرفته ام. چون احتمالاً زیاد این دو کلمه یک کلمه را تشکیل میداده اند و نباید جدا شوند.

```

word_tokenizer.py > WordTokenizer > tokenize
class WordTokenizer:
    def __init__(self) -> None:
        self.punctuations = [",", ".", "!", "?", "\"", "\'", "(", ")", "{", "}", "[", "]", "/", "\\", ";", ":", "»", "«", ">", "<", ":"]

    def delete_punctuations_symbols(self, text: str):
        for p in self.punctuations:
            text = text.replace(p, " ")
        return text

    def isAlphaOrNum(self, c):
        if (c >= "a" and c <= "z"):
            return True

        if (c >= "A" and c <= "Z"):
            return True

        if (c >= "0" and c <= "9"):
            return True

        return False

    def tokenize(self, text: str):
        text = text.replace("\t", " ")
        text = text.replace("\n", " ")
        # text = text.replace("\u200c", " ")
        text = text.replace(u'\xa0', " ")

        dot_number = text.count(".")
        last_dot_index = -1
        text_list = list(text)
        for i in range(dot_number):
            dot_index = text.find(".", last_dot_index + 1)
            last_dot_index = dot_index

            if (self.isAlphaOrNum(text_list[dot_index - 1]) and self.isAlphaOrNum(text_list[dot_index + 1])):
                continue
            else:
                text_list[dot_index] = " "

        text = "".join(text_list)
        text = self.delete_punctuations_symbols(text)
        tokens = text.split(" ")

        return tokens

```

شکل ۲

در شکل ۳ تستی از این مرحله را مشاهده میکنیم:

```
tokens = WordTokenizer().tokenize("aida.mobli@aut.ac.ir!! این یک تست است. برای این که بفهمیم درست کار میکنه یا نه")
```

```

test_tokenizer.txt
1  این
2  یک
3  تست
4  است
5  برای
6  این
7  که
8  بفهمیم
9  درست
10 کار
11 میکنه
12 یا
13 نه
14
15 aida.mobli@aut.ac.ir
16

```

شکل ۳

## ۱.۲. نرمال سازی متن

این مرحله را در کلاس Normalizer پیاده سازی شده است. نرمال سازی به ترتیبی که در تابع normalize\_tokens است، انجام میشود. (شکل ۴)

```

def normalize_tokens(self, tokens: list[str]):
    for i, t in enumerate(tokens):
        tokens[i] = self.normalize_alphabets(tokens[i])
        tokens[i] = self.normalize_numbers(tokens[i])
        tokens[i] = self.normalize_multiform_words(tokens[i])
    tokens = self.process_verbs(tokens)
    tokens = self.process_nouns(tokens)
    return tokens

```

شکل ۴

در تابع normalize\_alphabets، حذف اعراب، تعویض حروف عربی با حروف فارسی، و تبدیل کلمات فارسی با حروف کوچک انگلیسی انجام میشود (شکل ۵)

```
def normalize_alphabets(self, token: str):
    for d in self.diacritics:
        token = token.replace(d, "")

    token = token.replace("ؤ", "و")
    token = token.replace("ئ", "ی")
    token = token.replace("ی", "ی")
    token = token.replace("ا", "ا")
    token = token.replace("آ", "ا")
    token = token.replace("إ", "ا")
    token = token.replace("ة", "ه")
    token = token.replace("ك", "ک")

    token = token.lower()

    return token
```

شکل ۵

در تابع `normalize_numbers`، اعداد انگلیسی را به اعداد فارسی تبدیل میکنیم. (شکل ۶)

```
def normalize_numbers(self, token: str):
    token = token.replace("0", "۰")
    token = token.replace("1", "۱")
    token = token.replace("2", "۲")
    token = token.replace("3", "۳")
    token = token.replace("4", "۴")
    token = token.replace("5", "۵")
    token = token.replace("6", "۶")
    token = token.replace("7", "۷")
    token = token.replace("8", "۸")
    token = token.replace("9", "۹")

    return token
```

شکل ۶

در تابع `normalize_multiform_words`، بعضی از کلماتی که دو املا دارند را یکی میکنیم. (شکل ۷)

```
def normalize_multiform_words(self,token):
    token = token.replace("اينه","اينه")
    token = token.replace("اومبيل","اومبيل")
    token = token.replace("مليون","مليون")
    token = token.replace("هيات","هيت")
    token = token.replace("درم","درهم")
    token = token.replace("تهران","تهران")
    token = token.replace("باطري","باتري")
    token = token.replace("ذغال","زغال")
    token = token.replace("اطاق","اتاق")
    token = token.replace("امپراطور","امپراتور")
    return token
```

شکل ۷

در دو تابع process\_verbs و process\_nouns، فاصله گذاری ها اصلاح میشود. (شکل ۸) در این دو تابع، ها، تر، ترین، گر، گری، اش، ات، ام، می و نمی را با نیم فاصله به کلمه قبلی اش در اسم ها و به کلمه بعدش در فعل ها اضافه میکنیم.

```
def process_verbs(self,tokens: list[str]):
    return self.correct_spacing(tokens,["می","نمی"],before_word=True)

def process_nouns(self,tokens: list[str]):
    self.delete_useless_tokens(tokens)
    return self.correct_spacing(tokens,["ها","تر","ترین","گر","گری","ام","ات","اش"],before_word=False)

def correct_spacing(self,tokens:list[str],list_of_corrections,before_word=True):
    del_index = []
    for i,t in enumerate(tokens):
        if t in list_of_corrections:
            if(before_word):
                t = t+ "\u200c"
                tokens[i+1] = t+ tokens[i+1]
            else:
                a = tokens[i-1]+ "\u200c"
                tokens[i-1] = a+t
        del_index.append(i)

    for i in range(len(del_index)-1,-1,-1):
        tokens.pop(del_index[i])

    return tokens
```

شکل ۸

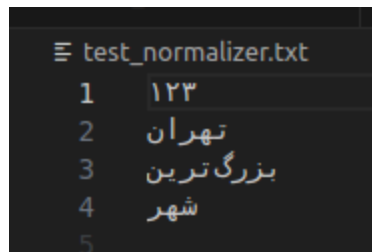
همچنین در تابع `delete_useless_tokens`، "ی" های میان توکن ها حذف میشوند. (شکل ۹)

```
def delete_useless_tokens(self, tokens: list[str]):  
    del_index = []  
    useless_tokens = ["ی", "ای"]  
    for i, t in enumerate(tokens):  
        if (t in useless_tokens):  
            del_index.append(i)  
  
        if t == "های" or t=="هایِ":  
            tokens[i] = "ها"  
        if t == "تری":  
            tokens[i] = "تر"  
        if t == "ترینِ":  
            tokens[i] = "ترین"  
        if t == "های" or t=="هایِ":  
            tokens[i] = "ها"  
  
    for i in range(len(del_index)-1, -1, -1):  
        tokens.pop(del_index[i])
```

شکل ۹

در شکل ۱۰، تستی از کارکرد این کلاس را مشاهده میکنید:

```
# print(json_file[ser(6565)]["content"])  
tokens = WordTokenizer().tokenize("123 شهر ترین بزرگ تهران")  
tokens = Normalizer().normalize_tokens(tokens)
```



|   |           |
|---|-----------|
| 1 | ۱۲۳       |
| 2 | تهران     |
| 3 | بزرگ‌ترین |
| 4 | شهر       |
| 5 |           |

شکل ۱۰

### ۱.۳. ریشه یابی

برای ریشه یابی از کتابخانه parsivar استفاده شد. با بررسی دو کتاب خانه parsivar , hazm به این نتیجه رسیدیم که parsivar ریشه یابی دقیق تری دارد. در کتاب خانه hazm کلمات زیادی که نباید ریشه یابی میشدند، به طور غلط ریشه یابی شدند. به طور مثال کلمه گزارش را به گزار تبدیل میکرد.

### ۱.۴. حذف ۵۰ کلمه پر تکرار:

پس از انجام مراحل بالا، ۵۰ کلمه پر تکرار را حذف میکنیم. برای پیدا کردن تعداد تکرار هر کلمه در سند ها و حذف ۵۰ کلمه پر تکرار، از کلاس MostFrequencies استفاده میشود. (شکل ۱۱)

```
class MostFrequencies :
    def __init__(self) -> None:
        self.term_dictionary = {}

    def count_terms(self,tokens):
        tokens = set(tokens)
        for t in tokens:
            if t == " ":
                continue
            if (t in self.term_dictionary.keys()):
                self.term_dictionary[t]+=1
            else:
                self.term_dictionary[t]=1

    def find_most_freq_terms(self):
        the_most_freq_terms= sorted(self.term_dictionary.items(), key=lambda x:x[1], reverse=True)
        converted_dict = dict(the_most_freq_terms)

        with open('sorted_term_freq.txt', 'w') as f:
            for line in converted_dict.keys():
                f.write(f"{line}\t {self.term_dictionary[line]}\n")

        terms = list(converted_dict.keys())
        terms = terms[0:50]
        with open('50_most_freq_terms.txt', 'w') as f:
            for line in terms:
                f.write(f"{line}\t {self.term_dictionary[line]}\n")
```

شکل ۱۱

---

در طر فرایند توکنایز کردن داکيومنت ها، به ازای هر داکيومنت تابع `count_terms` صدا زده میشود تا تعداد تکرار توکن در سند ها را بشمارد. این تابع ابتدا لیست توکن های یک سند را به یک مجموعه تبدیل میکند تا تکرار یک توکن ها را برای یک سند از بین ببرد.

پس از توکنایز کردن تمام داکيومنت ها، تابع `find_most_freq_terms` صدا زده میشود تا ۵۰ کلمه پر تکرار را پیدا کند. در شکل ۱۲ لیست ۵۰ کلمه پر تکرار را مشاهده میکنید.



| 50_most_freq_terms.txt |       |          |
|------------------------|-------|----------|
| 1                      | 12180 | به       |
| 2                      | 12120 | پیام     |
| 3                      | 12119 | فارس     |
| 4                      | 12071 | انتهای   |
| 5                      | 12062 | در       |
| 6                      | 12006 | و        |
| 7                      | 11999 | خبرگزاری |
| 8                      | 11445 | از       |
| 9                      | 11356 | این      |
| 10                     | 11151 | با       |
| 11                     | 10481 | گزارش    |
| 12                     | 10219 | را       |
| 13                     | 9682  | که       |
| 14                     | 9385  | اس       |
| 15                     | 8389  | کرد      |
| 16                     | 8385  | برای     |
| 17                     | 7913  | داشتند   |
| 18                     | 7889  | شد       |
| 19                     | 7878  | کردند    |
| 20                     | 7630  | شدند     |
| 21                     | 6775  | بودند    |
| 22                     | 6460  | ان       |
| 23                     | 6365  | تا       |
| 24                     | 6280  | خبرنگار  |
| 25                     | 6220  | وی       |
| 26                     | 6167  | یک       |
| 27                     | 6145  | بر       |
| 28                     | 5929  | خود      |
| 29                     | 5814  | تیم      |
| 30                     | 5748  | کشور     |
| 31                     | 5741  | شده      |
| 32                     | 5733  | گفت      |
| 33                     | 5600  | هم       |
| 34                     | 5501  | خواستند  |
| 35                     | 5385  | گرفتند   |
| 36                     | 5194  | دادند    |
| 37                     | 5098  | قرار     |
| 38                     | 4866  | امروز    |
| 39                     | 4823  | بايد     |
| 40                     | 4810  | داشتند   |
| 41                     | 4676  | داد      |
| 42                     | 4514  | بازی     |
| 43                     | 4489  | اما      |
| 44                     | 4479  | توانستند |
| 45                     | 4438  | اطهار    |
| 46                     | 4360  | اسلامی   |
| 47                     | 4307  | کند      |
| 48                     | 4286  | ادامه    |

شکل ۱۲

#### ۱.۴. ساخت شاخص مکانی

به طور کلی، تمامی فرایندها پیش پردازش در کلاس Indexer انجام میشود. فرایندهای قبلی در تابع tokenize\_text که یکی از تابع های این کلاس است انجام شد. این کلاس به دو صورت استفاده میشود: ۱. توکنایز کردن داکيومنت ها + پیدا کردن ۵۰ کلمه پر تکرار + حذف کلمات پرتکرار از لیست term های هر داکيومنت+ساخت شاخص مکانی ۲. توکنایز کردن داکيومنت ها +حذف کلمات پرتکراری که قبلاً مشخص شده است و در فایل most\_freq\_terms.txt\_50 ذخیره شده از لیست term های هر داکيومنت+ساخت شاخص مکانی

این دو حالت با استفاده از مقدار save\_most\_frequent\_words مشخص میشود که در ابتدای ساخت کلاس مشخص میشود. (شکل ۱۳) این مقدار اگر True باشد، حالت اول رخ میدهد.

```
class Indexer:
    def __init__(self, save_most_frequent_words = False) -> None:
        self.save_most_frequent_words= save_most_frequent_words
        self.tokenizer = WordTokenizer()
        self.normalizer = Normalizer()
        self.stemmer = FindStems()
        self.freq_term = MostFrequencies()
        if (not self.save_most_frequent_words) :
            self.get_eliminate_words()

        else:
            self.doc_token_list = dict()

        self.IR_dictionary = dict()
```

شکل ۱۳

برای ساخت شاخص از تابع tokenize\_docs استفاده شده است. در این تابع، ابتدا کل فایل json داکيومنت ها خوانده میشود. و بعد تمام داکيومنت ها به لیستی از term هایشان تبدیل شده و شاخص مکانی اشان ذخیره میشود. در شکل ۱۴ پیاده سازی تابع tokenize\_docs را مشاهده میکنید.

```

index_tokens.py > Indexer > tokenize_docs
65 def tokenize_docs(self):
66     data_address = "IR_data_news_12k.json"
67     f = open(data_address)
68     json_file = json.load(f)
69     self.doc_numbers = len(json_file)
70
71     # self.doc_numbers = 10
72     docs_collections = list(json_file.keys())
73     docs_collections.sort()
74     cnt = 0
75     for i in docs_collections:
76         if cnt%1000 == 1:
77             print (f"process {cnt} docs\n")
78             cnt+=1
79
80         text = json_file[i]["content"]
81
82
83
84         if (self.save_most_frequent_words):
85             tokens = self.tokenize_text(text,True)
86             self.freq_term.count_terms(tokens)
87             self.doc_token_list[i]= tokens
88
89         else:
90             tokens = self.tokenize_text(text)
91             self.create_posting_list(tokens,i)
92
93
94
95     if(self.save_most_frequent_words):
96         self.freq_term.find_most_freq_terms()
97         self.get_eliminate_words()
98         for i in docs_collections:
99             tokens = [ t for t in self.doc_token_list[str(i)] if not (t in self.eliminated_words or t == "")]
100             self.create_posting_list(tokens,str(i))
101             del self.doc_token_list[str(i)]
102     for t in self.IR_dictionary.keys():
103         self.IR_dictionary[t]["doc_frequency"] = len(self.IR_dictionary[t]["docs"].keys())
104

```

شکل ۱۴

همان طور که در شکل ۱۴ مشخص است، زمانی که به طور همزمان می‌خواهیم ۵۰ کلمه پر تکرار را استخراج کنیم، شاخص مکانی در زمان توکنایز کردن دایومنت ها انجام نمی‌شود. صرفاً تعداد تکرار term ها شمارش می‌شود تا در ادامه از آن ها برای مشخص کردن ۵۰ کلمه پر تکرار استفاده شود. و در زمانی که همه داکيومنت ها پردازش شد، ابتدا ۵۰ کلمه پر تکرار مشخص شده، آن کلمات از لیست کلمات هر داکيومنت حذف می‌شود و بعد به ساخت شاخص مکانی پرداخته می‌شود.

ساخت شاخص مکانی در تابع create\_posting\_list انجام می‌شود. (شکل ۱۵) برای ساخت شاخص مکانی از ساختمان داده dictionary پایتون استفاده شده است. این ساختمان داده برای حجم داده ای که داشتیم مناسب بود و در عین حال که پیاده سازی را ساده تر می‌کند، از سرعت مطلوبی برخوردار بود.

```

def create_posting_list(self, tokens: list[str], docID):

    count = 0
    for t in tokens :

        if t in self.IR_dictionary.keys(): ...
        else:
            self.IR_dictionary[t] = dict()
            self.IR_dictionary[t]["docs"] = dict()
            self.IR_dictionary[t]["docs"][docID]= dict()
            self.IR_dictionary[t]["docs"][docID]["positions"] = list()
            self.IR_dictionary[t]["docs"][docID]["term_frequency"] = 0

            self.IR_dictionary[t]["docs"][docID]["positions"].append(count)
            self.IR_dictionary[t]["docs"][docID]["term_frequency"] += 1

            count +=1

```

شکل ۱۵

## ۲. پاسخ به پرسمان در فضای برداری

### ۲.۱. مدل سازی اسناد در فضای برداری

در ابتدا باید امتیاز tf-idf را به ازای هر term و داکيومت بسازيم. اين کار را در تابع tokenize\_docs زمانی که شاخص های مکانی را ساختيم، انجام ميدهيم. (شکل ۱۶) و همچنين نرمال سازی اين امتیاز ها را هم در اين مرحله انجام ميدهيم تا در زمان پاسخگویی به پرسمان ها سريع تر عمل کنيم.

```

99         tokens = [ t for t in self.doc_token_list[str(i)] if not (t in self.eliminated_words or t == "")]
100         self.create_posting_list(tokens, str(i))
101         del self.doc_token_list[str(i)]
102     for t in self.IR_dictionary.keys():
103         self.IR_dictionary[t]["doc_frequency"] = len(self.IR_dictionary[t]["docs"].keys())
104
105     print("calculate tf-idfs\n")
106     self.calculate_tf_idf()
107     self.calculate_doc_vector_normalization()
108
109     # normalize tf_idfs
110
111     for t in self.IR_dictionary.keys():
112         for d in self.IR_dictionary[t]["docs"].keys():
113             self.IR_dictionary[t]["docs"][d]["tf-idf"] = self.IR_dictionary[t]["docs"][d]["tf-idf"] / self.normalization_vector_docs[d]
114
115
116
117

```

شکل ۱۶

```

def calculate_tf_idf(self):
    for t in self.IR_dictionary.keys():
        n_t = self.IR_dictionary[t]["doc_frequency"]
        for d in self.IR_dictionary[t]["docs"].keys():
            f_td = self.IR_dictionary[t]["docs"][d]["term_frequency"]
            self.IR_dictionary[t]["docs"][d]["tf-idf"] = (1+math.log(f_td,10))*math.log(self.doc_numbers/n_t)

def calculate_doc_vector_normalization (self):
    self.normalization_vector_docs = dict()

    for t in self.IR_dictionary.keys():
        for d in self.IR_dictionary[t]["docs"].keys():
            if (not d in self.normalization_vector_docs.keys()):
                self.normalization_vector_docs[d] = 0
            term_frequency = self.IR_dictionary[t]["docs"][d]["term_frequency"]
            tf_idf = self.IR_dictionary[t]["docs"][d]["tf-idf"]
            self.normalization_vector_docs[d] += term_frequency * (tf_idf **2)

    self.normalization_vector_docs = {d:self.normalization_vector_docs[d]**0.5 for d in self.normalization_vector_docs.keys() }

```

شکل ۱۷

امتیاز  $tf\_idf$  با استفاده از فرمول زیر محاسبه میشود که این محاسبه در تابع `calculate_tf_idf` پیاده سازی شده است. (شکل ۱۷)

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) = (1 + \log(f_{t,d})) \times \log\left(\frac{N}{n_t}\right)$$

پس از محاسبه تمام امتیازهای  $tf\_idf$  به ازای هر داکيومنت، اندازه بردارها را محاسبه میکنیم تا بتوانیم برمال سازی را انجام دهیم.

یک مثال از مقادیر  $tf\_idf$  در شاخص یکی از کلمات مشاهده میکنید:

گلف = {'275': {'positions': [7, 43], 'term\_frequency': 2, 'tf-idf': 0.1593529236267452}, '4650': {'positions': [7, 46, 86, 144], 'term\_frequency': 4, 'tf-idf': 0.17567142152994583}, '472': {'positions': [641], 'term\_frequency': 1, 'tf-idf': 0.060480493117780566}, '5470': {'positions': [3, 9, 69, 96, 153, 185, 213, 233, 262, 294], 'term\_frequency': 10, 'tf-idf': 0.1347209691174863}, '5715': {'positions': [5, 37, 44], 'term\_frequency': 3, 'tf-idf': 0.2720360610933851}}, 'doc\_frequency': 5

۲.۲. پاسخ دهی به پرسشان

شباهت دو داکيومنت به هم، از شباهت کسینوسی بین آنها بدست می آید. برای افزایش سرعت پاسخ دهی، از تکنیک index elimination استفاده میشود که در آن شباهت کسینوسی فقط بین کلمات موجود در پرسمان حساب میشود و این شباهت را بین کل کلمات داکيومنت حساب نمیکنیم. چرا که در پرسمان وجود ندارند پس در انتها مقدارشان صفر میشود .

پاسخ دهی به پرسمان ها در تابع answer\_query که در کلاس IR است، انجام میشود. (شکل ۱۸)

```
def answer_query(self, query):
    dict_query = dict()
    tokens = self.indexer.tokenize_text(query)

    for t in tokens :
        if not t in dict_query.keys():
            dict_query[t]=0
        dict_query[t]+=1

    tf_idf_query = {t:1+math.log(dict_query[t],10) for t in dict_query.keys()}
    print(tokens)

    docs = self.find_suitable_docs(set(tokens))
    docs_score = dict()
    for d in docs :
        for t in docs[d].keys():
            if not d in docs_score.keys():
                docs_score[d] = 0
            docs_score[d] += docs[d][t] * tf_idf_query[t]

    sorted_docs = sorted(docs_score.items(), key=lambda x:x[1], reverse=True)[:10]
    best_docs = dict(sorted_docs)

    with open(f'{query}.txt', 'w') as f:
        for d in best_docs.keys():
            title = self.json_file[d]["title"]
            link = self.json_file [d]["url"]
            f.write(f"score: {best_docs[d]} - doc : {title}\t{link}")
            f.write("-----\n")
```

شکل ۱۸

ابتدا پرسمان را همانند پیش پردازشی که روی داکيومنت ها انجام داده ایم، توکنایز میکنیم.

سپس tf\_idf را برای پرسمان محاسبه میکنیم. چون که idf به داکيومنت مرتبط نیست و یک بار در محاسبه tf\_idf اسناد محاسبه اش کردیم، دوباره محاسبه اش نمیکنیم. و فقط tf را محاسبه میکنیم. و همچنین نیازی نیست که نرمال سازی برای پرسمان انجام شود زیرا پرسمان قرار است با همه داکيومنت ها مقایسه شود در نتیجه طول پرسمان مهم نیست.

بعد از استخراج term های پرسمان، داکيومنت های مرتبط با آن term ها را پیدا میکنیم. (شکل ۱۹) این کار به این صورت انجام میشود که به ازای هر term داکيومنت های مرتبط با آن را استخراج میکنیم و docs\_dict را میسازیم که یک دیگشنری است که به ازای هر داکيومنت، مجموعه کلمات مشترک بین پرسمان و داکيومنت را نگه میدارد.

```
def find_suitable_docs(self,tokens):
    docs_dict = dict()
    if self.enable_champion_list:
        for t in tokens:
            for d in self.champion_list[t]:
                if not d in docs_dict.keys():
                    docs_dict[d] = dict()
                    docs_dict[d][t] = float(self.champion_list[t][d])
    else:
        for t in tokens:
            for d in self.IR_dictionary[t]["docs"].keys():
                if not d in docs_dict.keys():
                    docs_dict[d] = dict()
                    docs_dict[d][t] = float(self.IR_dictionary[t]["docs"][d]["tf-idf"])

    return docs_dict
```

شکل ۱۹

پس از مشخص کردن تمام داکيومنت هایی که حداقل یکی از کلمات پرسمان را دارند، شباهت کوسینوسی آنها با پرسمان محاسبه شده. سپس آن ها را به صورت کاهش مرتب میکنیم و ۱۰ داکيومنت اول، نزدیک ترین داکيومنت ها به پرسمان ما خواهد بود. (شکل ۱۸)

### ۲.۳. افزایش سرعت پرسمان با لیست قهرمانان

با استفاده از تابع generate\_champion\_list که در کلاس Indexer قرار دارد، این لیست ساخته میشود (شکل ۲۰) برای این کار، ابتدا داکيومنت ها بر اساس tf\_idf شان مرتب میشوند و k سند با بالاترین tf\_idf ها انتخاب میشوند.

```
def generate_champion_list(self,k):
    self.champion_list = dict()
    for t in self.IR_dictionary.keys():
        docs = {str(d):self.IR_dictionary[t]["docs"][d]["tf-idf"] for d in self.IR_dictionary[t]["docs"].keys()}
        sorted_docs = sorted(docs.items(), key=lambda x:x[1], reverse=True)
        if k <= len(docs):
            sorted_docs = sorted_docs[0:k]
        else:
            sorted_docs = sorted_docs[0:len(docs)]

        self.champion_list[t] = dict(sorted_docs)
```

شکل ۲۰

در زمان پاسخ دهی، اگر استفاده از لیست قهرمانان فعال باشد، بر اساس لیست قهرمانان مناسب ترین داکيومنت ها را انتخاب میکند. در غیر این صورت از خود دیکشنری استفاده میکند. (شکل ۱۹)

## بررسی پرسمان:

### الف) یک پرسمان از کلمات ساده و متداول تک کلمه ای

به طور مثال کلمه ایران را جست و جو میکنیم. در شکل ۲۱ نتایج رو مشاهده میکنید.

|    |              |                            |         |  |  |
|----|--------------|----------------------------|---------|--|--|
| 1  | docid= 9893  | score:0.10029934442804164  | - link= | https://www.farsnews.ir/news/14000921000552/ | با حکم رئیس‌جمهور، مختارپور رئیس سازمان اسناد و کتابخانه ملی شد:               |
| 2  | docid= 3119  | score:0.08929291554933852  | - link= | https://www.farsnews.ir/news/1400112000930/  | بهدین بازیکن تیم مقابل امارات معرفی شد:  |
| 3  | docid= 11876 | score:0.0887229715643296   | - link= | https://www.farsnews.ir/news/14000803000598/ | سفر جدید جمهوری آذربایجان در ایران استوارنامه خود را تقدیم رئیس‌جمهور کرد:     |
| 4  | docid= 3529  | score:0.08726354943649418  | - link= | https://www.farsnews.ir/news/14001107000561/ | حضور وزیر ورزش در ورزشگاه آزادی برای نمایش بازی ایران و عراق:                  |
| 5  | docid= 4405  | score:0.08384401390516989  | - link= | https://www.farsnews.ir/news/14001027000611/ | اعلام ساعت دیدارهای تیم ملی مقابل عراق و امارات:                               |
| 6  | docid= 2665  | score:0.083667320088087817 | - link= | https://www.farsnews.ir/news/14001118000913/ | کامپانیای 200 بازیکن تیم ملی ایران:  |
| 7  | docid= 3636  | score:0.08255140874900196  | - link= | https://www.farsnews.ir/news/14001106000296/ | برخلاف سابق مطرح شده: بازی ایران و عراق با حضور 10 هزار تماشاگر برگزار می‌شود: |
| 8  | docid= 9572  | score:0.077032820508003048 | - link= | https://www.farsnews.ir/news/14000930000841/ | بنام تسلیت رئیس‌جمهور به مناسبت غرق شدن ایلو:                                  |
| 9  | docid= 9525  | score:0.07467874033803282  | - link= | https://www.farsnews.ir/news/14001002000537/ | برگزاری جلسه شورای عالی اطفال به ریاست رئیس‌جمهور:                             |
| 10 | docid= 7145  | score:0.07458995800494141  | - link= | https://www.farsnews.ir/news/14001216000384/ | راکنش ارکان جبهه اصلاحات ایران به همه آمده موکول شد:                           |

شکل ۲۱

#### ۱. با حکم رئیس‌جمهور، مختارپور رئیس سازمان اسناد و کتابخانه ملی شد

به گزارش خبرنگار حوزه دولت خبرگزاری فارس، با حکم آیت الله سید ابراهیم رئیسی، رئیس‌جمهور اسلامی ایران، علیرضا مختارپور به عنوان رئیس جدید سازمان اسناد و کتابخانه ملی جمهوری اسلامی ایران منصوب شد.

#### ۲. بهترین بازیکن دیدار تیم ملی مقابل امارات معرفی شد

به گزارش خبرنگار ورزشی خبرگزاری فارس، تیم ملی فوتبال کشورمان در ادامه بازی‌های مرحله نهایی انتخابی جام جهانی 2022، در ورزشگاه آزادی تیم ملی امارات را شکست داد.

در پایان این بازی مهدی طارمی به عنوان بهترین بازیکن زمین انتخاب و معرفی شد.

این بازیکن تک گل ایران را در مسابقه امشب به ثمر رساند.

#### ۳. سفیر جدید جمهوری آذربایجان در ایران استوارنامه خود را تقدیم رئیس‌جمهور کرد



به گزارش خبرگزاری فارس، آیت‌الله سید ابراهیم رئیسی رئیس‌جمهور صبح امروز (دوشنبه) استوارنامه «علی‌علیزاده» سفیر جدید **جمهوری آذربایجان** در **ایران** را دریافت کرد.

در این حالت جواب هایی که مشخص شده است مرتبط با ایران هستند ولی احتمالا وقتی تک کلمه ایران را جست و جو میکنیم منظورمان این است که راجع به خود ایران میخواهیم بدانیم اما در اینجا موضوعات در مورد موضوعاتی مانند ورزشی و .. است. همانطور هم که مشخص است، اسنادی بالا ترین امتیاز را گرفته اند که طول محتوایشان کم است.

(ب) یک یرسمان از کلمات ساده و متداول چند کلمه ای

در این مثال یرسمان "بازیکن فوتبال ایران" جست و جو شده است. (شکل ۲۲)

[illegible]

شکا، ۲۲

### ۱. بهترین بازیکن دیدار تیم ملی مقابل امارات معرفی شد

به گزارش خبرنگار ورزشی خبرگزاری فارس، تیم ملی فوتبال کشورمان در ادامه بازی‌های مرحله نهایی انتخابی جام جهانی 2022، در ورزشگاه آزادی تیم ملی امارات را شکست داد.

در پایان این بازی مهدی طارمی به عنوان بهترین بازیکن زمین انتخاب و معرفی شد.

این بازیکن تک گل **ایران** را در مسابقه امشب به ثمر رساند.

## ۲. استیلی سرپرست تیم ملی فوتبال شد

به گزارش خبرنگار ورزشی خبرگزاری فارس، حمید استیلی **بازیکن** سابق تیم ملی **فوتبال ایران** به عنوان سرپرست تیم ملی فوتبال انتخاب شد. قرار است وی به زودی امور سرپرستی تیم ملی را بر عهده دارد.

بیش از این مجتبی خورشیدی سرپرستی تیم ملی فوتبال را بر عهده داشت.

### ۳. کامیابی‌نیا در باشگاه 200 تایی‌های پرسپولیس

به گزارش خبرگزاری فارس، کمال کامیابی‌نیا با قرار گرفتن در فهرست بازی با فولاد در چارچوب دیدار سوپر جام فوتبال ایران، تعداد بازی‌های رسمی خود برای سرخپوشان ایران را به عدد 200 می‌رساند.

او چهاردهمین بازیکن تاریخ باشگاه است که این تعداد بازی را ثبت خواهد کرد.

وی پیش از این با رسیدن به رکورد 198 بازی حمید درخشان، پیشکسوت نامی فوتبال ایران و پرسپولیس، گذشته بود.

همانطور که در نتایج مشخص است، سند های بازگردانده شده مرتبط هستند و همه ی خبر ها مرتبط به یک بازیکن فوتبال ایران هستند!

### پ) یک پرسمان از کلمات دشوار و کم تکرار تک کلمه ای

به طور مثال کلمه انسان‌شناسی را جست و جو میکنیم. (شکل ۲۳)

```
#انسان‌شناسی.txt
1 docid= 10236 score:0.06624627982385702 - link: رحیم پور ازغدی: انسان‌شناس سکولار حتماً باید با عقل و تجربه بررسی شود: https://www.farsnews.ir/news/14000913000937
2 docid= 9875 score:0.04893556789665433 - link: نادر/https://www.farsnews.ir/news/14000922000364 میرگرد «اسلام، مبدأ تحول در علوم انسانی» / متولیان پاسخگو و معیت تحول در علوم انسانی باشد:
3 docid= 9206 score:0.036140357234515365 - link: اسخ به سوال و ضیاء جوانان/https://www.farsnews.ir/news/14001011000519 عبدالله مصباح‌زادی: از طرح ولایت تا پاسخ به سوال و ضیاء جوانان:
4 docid= 7341 score:0.03060822487026932 - link: https://www.farsnews.ir/news/14001209000061 میرگرد: عملکرد جمهوری اسلامی در تحول علوم انسانی | غربیها علوم خود را بر اساس ریسنبوم خود نوشند:
5 docid= 9180 score:0.0241598145034622 - link: ریش انقلاب/https://www.farsnews.ir/news/14001012000305 ریش انقلاب‌ها» داریم اما «ریش انقلاب» نه/ باید با همگرای، اختلاف را کم کنیم:
6
```

شکل ۲۳

### ۱. رحیم پور ازغدی: انسان‌شناسی سکولار حتماً باید با عقل و تجربه بررسی شود

(متن خیلی بزرگ است تیکه های مرتبط را میگذارم)

وی در ادامه گفت: علوم انسانی ادعای انسان‌شناسی است، یعنی داریم به انسان علم پیدا می‌کنیم. ما انسان را شناختیم، انسان را دانستیم، در حالی که این ادعا است! نه اینجا، بلکه ادعای کل جهان، ادعای کاذب و دروغ و تهمت به انسان است. ما و شما به بعضی از لایه‌های انسان و آثار افعال و رفتارها یک وقوف اجمالی پیدا می‌کنیم اما یک جمع‌بندی علمی و صددرصد مورد پسند در جهان نداریم.

وی با تاکید بر این موضوع که انسان‌شناسی سکولار ناقص و تک‌بعدی است، افزود: اما انسان‌شناسی الهی چندبعدی است و باید همه ابعاد آن را به وسیله عقل و تجربه در نظر گرفت و بررسی کرد. مکتب‌های مارکسیست و لیبرال و فاشیست فقط مدعی علم‌اند و ۹۹ درصد را فدای ۱ درصد می‌کنند و اسلام الهی را زیر سوال می‌برند. تجربه به تنهایی برای انسان کافی نیست بلکه عقل هم به همان میزان اهمیت دارد اما یک منبعی وجود دارد که نه تجربی است و نه مستقیماً عقلی اما انسان‌ها

را به آگاهی می‌رساند که از طریق عقل آن را می‌پذیریم به نام انبیا. انبیا کمک می‌کنند که عقل و تجربه در حوزه انسان‌شناسی به درستی عمل کنند.

رحیم پور از غدی تاکید کرد: **انسان‌شناسی** واقعی در مرگ رخ می‌دهد؛ زیرا فقط خود انسان و اعمالش هستند، این خود انسان است که مسئول تک تک کنش‌هایش است و فراخود مربوط به وجدان آدمی است. مثل یک دستگاه در وجود انسان. عضو شورای عالی انقلاب فرهنگی ادامه داد: **انسان‌شناسی** سکولار حتماً باید با عقل و تجربه بررسی شود. در این زمینه چندین مکتب مورد بررسی قرار می‌گیرد و بعد برخی از آنها پذیرفته می‌شود و برخی دیگر پذیرفته نمی‌شود.

همانطور که مشخص است، نتیجه مرتبط است.

## ۲. میزگرد «اسلام، مبدا تحول در علوم انسانی» / متولیان پاسخگوی وضعیت تحول در علوم انسانی باشند

اسلامی با بیان اینکه وقتی صحبت از تحول و ضرورت تحول می‌شود باید ببینیم نگاه ما معنای تحول از دیدگاه ما چیست؟ و در ادامه افزود: در کشور ما وقتی صحبت از تحول می‌شود چهار شاخص بیان می‌گردد. ۱- روزآمد بودن تحول‌ها، ۲- میزان کارآمدی و کاربردی بودن علوم انسانی در جامعه (وقتی این علم به نیازها و مشکلات جامعه پاسخ نداده ناکارآمدی آن مشخص می‌شود). ۳- بومی‌سازی علوم انسانی (یعنی مسائلی که درباره آنها صحبت می‌شود تا چه میزان مقتضیات فرهنگی، سیاسی، اجتماعی و زیستی یک انسان ایرانی را در بافت فرهنگ اسلام حل می‌کند). ۴- اسلامی‌سازی (یعنی ما بتوانیم با توجه به مبانی هستی‌شناسی، معرفت‌شناسی، ارزش‌شناسی، **انسان‌شناسی** تولید علم انجام بدهیم. باید توجه داشت دو شاخص اول حتماً شروط لازمی هستند اما کافی نیستند و زمانی به کفایت می‌رسد که به بومی‌سازی و اسلامی‌سازی نیز توجه شود).

وی گفت: معلومات تجربی به شدت محدود هستند. معلومات عقلی از رسیدن به آن ابدیت و دیدن آن بی‌نهایت بسیار قاصرند. براساس آموزه‌های صحیح انسان آفریده شده است برای اینکه باشد. اگر بنا است که ما باشیم علم ماندنمان و علمی که بتوانیم برای رسیدن به آن هدف سالم بمانیم چیست؟ اصلی‌ترین آنها چیست؟ البته عقل، تجربه، آموزش، تکرار و آزمون موثر است ولی وحی و گواهی نقش بسیار سازنده‌ای در این زمینه دارند. بنابراین اگر وحی، گواهی و تعلیمات انبیاء در این زمینه تاثیر اصلی را برای انسان‌شناسی صحیح دارد در مقابل انسان‌شناسی تجربی و عقلی، یک **انسان‌شناسی** اسلامی داریم که در آن ضمن استفاده از منبع عقل، تجربه، شهود بیشترین استفاده را از منبع وحی و آگاهی‌های مصون از خطا دارد اگر انسان حقیقی از این مسیر شناخته شود.

محمود نمازی عضو هیئت علمی موسسه امام خمینی (ره) در ادامه این راهکارها اظهار داشت: ما با یک چرخه موجود بسیار معیوب مواجه هستیم و وضع مطلوب این است که مستنداً به فرمایشات حضرت آقا، مبانی علوم انسانی موجود در معرفت‌شناسی، در هستی‌شناسی، **انسان‌شناسی**، ارزش‌شناسی و... دچار تحول شود و با ایجاد مبانی صحیح در این موارد به

جای مبنای غربی می‌توانیم راه اسلامی‌سازی را هموار کنیم. راهبرد ما برای عبور از وضع موجود و رسیدن به وضع مطلوب ساختن انسان‌هایی عمیق و دقیق هم در علوم غربی و هم در علوم اسلامی است.

### ۳. آیات‌الله مصباح‌یزدی؛ از طرح ولایت تا پاسخ به سوالات و شبهات جوانان

پس روشن شد که فرمول اصول دین، خیلی کارساز است؛ در ابتدا ما فکر می‌کردیم یک چیزی است که فقط باید آن را حفظ کنیم و نمره‌ای بگیریم؛ درحالی که این اساس زندگی انسان است و سعادت و شقاوت ابدی ما به آن بستگی دارد. ابتدا باید روشن شود که آیا خدا، آخرت و پیامبر را قبول دارید یا خیر؟ اگر بخواهید این را در قالب یک نظام علمی درآورید، همان خداشناسی، هستی‌شناسی و **انسان‌شناسی** است. آیا انسان همین موجودی است که هفتاد سال زندگی می‌کند و بعد در زیر خاک متلاشی می‌شود، یا نه، این یک دوره محدود است؛ یک دوره جنینی است که انسان باید خودش را برای ورود به یک عالم ابدی آماده سازد؟ اگر خدا و معاد را پذیرفتیم، چه کنیم که از این‌جا به آن‌جا برسیم؟ پیغمبر باید بگوید که این مسیر چگونه طی شود. این‌که از چه راهی این شناخت‌ها را به دست آوریم، در شاخه‌ای از علم به‌نام معرفت‌شناسی تبیین می‌شود. بنابراین، تا این مرحله باید این سه اصل را بپذیریم.

پس مجموع این آموزه‌ها را می‌توانیم در شش بخش طراحی کنیم که عبارت‌اند از: هستی‌شناسی، **انسان‌شناسی**، معرفت‌شناسی، فلسفه اخلاق، فلسفه حقوق و فلسفه سیاست.

### ت) یک پرسمان از کلمات دشوار و کم تکرار چند کلمه ای

به طور مثال هنرستان عکاسی را جست و جو میکنیم (شکل ۲۴)

|    |  |
|----|--|
| 1  | docid= 4386 score:0.2813894442443689 - link: فیلم: فرهمند سوره عکاسی شد + فیلم: <a href="https://www.farsnews.ir/news/14001028000389">https://www.farsnews.ir/news/14001028000389</a>  |
| 2  | docid= 8161 score:0.20171668515066957 - link: اجازه تلفیق به دولت برای استفاده از فاینانس به منظور تجهیز آژما بنگاه‌ها: <a href="https://www.farsnews.ir/news/1400111000674">https://www.farsnews.ir/news/1400111000674</a>                  |
| 3  | docid= 10864 score:0.11016718659429983 - link: آقامه می‌نمود: <a href="https://www.farsnews.ir/news/14000027000241">https://www.farsnews.ir/news/14000027000241</a>  |
| 4  | docid= 7282 score:0.08568478025477229 - link: می‌تکن سن آتم جمع: ارسال به 42 سال گاهش باف: <a href="https://www.farsnews.ir/news/14001212000148">https://www.farsnews.ir/news/14001212000148</a>   |
| 5  | docid= 7179 score:0.0817572828505626 - link: نحوه هزینه کرد فاینانس برای طرح های دولتی و غیردولتی تعیین شد: <a href="https://www.farsnews.ir/news/14001215000728">https://www.farsnews.ir/news/14001215000728</a>                            |
| 6  | docid= 11723 score:0.07122692231057566 - link: مطهری: بنیضرت وافعی جز از طریق نفیوت آموزش و پرورش ممکن نیست: <a href="https://www.farsnews.ir/news/14000809000410">https://www.farsnews.ir/news/14000809000410</a>                           |
| 7  | docid= 10547 score:0.06215010073734235 - link: برگزاری جشنواره ملی عکس "شکو جمعه" به همت شورای سیاستگذاری انجمنجمعه: <a href="https://www.farsnews.ir/news/14000907000242">https://www.farsnews.ir/news/14000907000242</a>                   |
| 8  | docid= 7856 score:0.05970264458243529 - link: معاون اول رئیس‌جمهور: دولت مصمم به اجرای طرح ریمه‌بندی معلمان است: <a href="https://www.farsnews.ir/news/1400119000741">https://www.farsnews.ir/news/1400119000741</a>                         |
| 9  | docid= 9970 score:0.03870071423660548 - link: هزار کلاس بدون معلم/ نثار 200 هزار میلیاردی برای سنددار کردن اموال آموزش و پرورش 70: <a href="https://www.farsnews.ir/news/14000915000506/0">https://www.farsnews.ir/news/14000915000506/0</a> |
| 10 |  |

### شکل ۲۴

#### ۱. جام ملت‌های فوتبال بانوان آسیا | کودک فرهمند سوژه عکاسی شد + فیلم

الهام فرهمند بازیکن تیم ملی فوتبال بانوان کشورمان که همراه با کودک 6 ماهه‌اش راهی هند شده در هند محل برگزاری مسابقات هم سوژه **عکاسی** و در ژست‌های مختلف از او عکس گرفته شد.

#### ۲. اجازه تلفیق به دولت برای استفاده از فاینانس به منظور تجهیز آزمایشگاه‌ها

---

وی افزود: همچنین مصوب شد تا یک میلیارد یورو برای **هنرستان** های وزارت آموزش و پرورش، دانشگاه فرهنگیان، دانشگاه تربیت دبیر شهید رجایی، مراکز آموزش فنی و حرفه ای دولتی و سازمان تحقیقات، آموزش و ترویج کشاورزی با تضمین دولت و باز پرداخت آن از محل اعتبارات بودجه عمومی تأمین کند.

### ۳. نماز جمعه این هفته تهران به امامت حجت الاسلام حاج علی اکبری اقامه می شود

حجت الاسلام علی نوری گفت: بر همین اساس برگزاری نگارگزار کتاب های تفریضی مقام معظم رهبری، عضویت رایگان کتابخانه، مسابقه دلقوشته و شعر از فهمیده تا لندی، کتاب در گردش، هم عهدی با شهدا، فال شهدا، ایستگاه خلاقیت و نقاشی، یک جمله یک کتاب، جشنواره رادیویی یک گام به جلو، قرآن بخوانید هدیه بگیرید، نمایشگاه کتاب (اتوبوس هدهد)، ایستگاه **عکاسی** با شهدا، ایستگاه خوشنویسی، توزیع ماسک رایگان، ایستگاه جورچین کودکان، ایستگاه بازی های **کودک** و نوجوان، ایستگاه محصولات مجازی و نمایشگاه کارتون و کاریکاتور نبض زمین از خدمات فرهنگی و هنری فرهنگسرای منتظر است که در نماز جمعه این هفته تهران به نمازگزاران ارائه می شود.

احتمالا چون در سند ها دو کلمه عکاسی و هنرستان را در یک داکيومنت نداشتیم، نتایج مرتبط نشده است. ولی به طور کلی اگر همین سندی میبود، باید مرتبط ترین میشد.