۱- ساخت شاخص مکانی

۱-۱ مجموعه داده

از مجموعه داده 12k ای اخبار خبرگذاری فارس برای ساخت شاخص استفاده خواهیم کرد. از اطلاعات هر داکیومنت، فقط متن خبر، عنوان خبر و url خبر را استفاده خواهیم کرد.

تابع load_docs

تابع برای خواندن فایل داده و ساخت dict از داکیومنت ها صدا زده میشود. key همان آیدی داک ها و value ها شامل content، title و url خبرند. تابع سه خروجی برمیگرداند که شامل دیکشنری docs (همه اطلاعات)، contents (متن خبرها) و urls (لینک خبرها) است.

```
def load_docs():
    docs = {}
    contents = []
    urls = []
    with open("IR_data_news_12k.json", 'r') as file:
        docs = json.load(file)
        for key in docs.keys():
            # index of files
            idx = str(key)
            # extract and save url, title and content of each doc
            docs[idx] = {'title': docs[idx]['title'],
                         'content': docs[idx]['content'],
                         'url': docs[idx]['url'],
                        }

            contents.append(docs[idx]['content'])
    return docs, contents, urls
```

نمونه ای از خروجی تابع و محتوای آن

```
docs, contents, urls = load_docs()
docs['6']
```

```
{'title': 'اعلام برنامه نشست خبری گل محمدی/ مجیدی هم باید به محل تمرین پرسپولیس برود!',
 'content': 'سایت باشگاه پرسپولیس، نشست خبری پیش از مسابقه سرمربیان دو تیم پرسپولیس و استقلال از هفته ب\xa0به گزارش خبرگزاری فارس و به نقل از\n\
برگزار ه\xa0باشگاه میزبان (پرسپولیس) در ورزشگاه شهید کاظمی به شرح زیر\xa0مدیریت سازمان لیگ و هماهنگی\xa0یست و سوم لیگ برتر (جام خلیج فارس) با
مسابقه دو تیم روز پنجشنبه در ورزش\xa0 شود: چهارشنبه ۲۵ اسفند ساعت ۱۳ فرهاد مجیدی سرمربی استقلال ساعت ۱۳:۳۰ یحیی گل محمدی سرمربی پرسپولیس\u200cی
به گزارش خبرگزاری فارس، پیش از این باشگاه استقلال اعلام کرده بود قرار است نشست خبری فرهاد مجیدی، سرمربی این تی\xa0 .شود\u200cگاه آزادی برگزار می
م از ساعت ۱۵ در سازمان لیگ برگزار شود. انتهای پیام /\n\n\n',
 'url': 'https://www.farsnews.ir/news/14001224000865/اعلام-برنامه-نشست-خبری-گل-محمدی-مجیدی-هم-باید-به-محل-تمرین-پرسپولیس'}
```

۱-۲ پیش‌پردازش اسناد

کلاس DataNormalization

در این کلاس توابعی برای نرمالسازی متن پیاده شده. در تابع __init__ این کلاس مقداردهی های این کلاس صورت میگیرد. برای مثال در پیش‌پردازش فعل‌ها نیاز داریم "می" و "نمی" را از افعال با نیم فاصله جدا کنیم. برای اینکار از بن‌های ماضی و مضارع مصادر استفاده میکنیم. این بن‌ها در فایل verbs قرار دارند و ماض و مضارع با علامت # جدا شده‌اند.

نمونه این از محتوای فایل verbs.dat

```
آخت#آهاز
آخت#آهنج
آخت#آهنج یا آهاز
آراست#آرای
```

قطعه کد زیر در تابع __init__ بن‌های ماضی و مضارع را جدا و ذخیره میکند.

```
# bons of verbs
with Path('verbs.dat').open(encoding="utf8") as verbs_file:
        verbs = list(
            reversed([verb.strip() for verb in verbs_file if verb]),
        )
        self.present_bons = {verb[1:].split("#")[0].strip() for verb in verbs[1:]}
        self.past_bons = {verb.split("#")[1] for verb in verbs}
```

یک تابع پر استفاده در این کلاس regex_replac است. این تابع جفتی از پترن رجکس ها را دریافت میکند و در text ورودی، پترن اول را با دومی جایگزین میکند و باز میگرداند.

```
@staticmethod
def regex_replace(patterns: list, text: str) -> str:
    for pattern, repl in patterns:
        text = re.sub(pattern, repl, text)
    return text
```

تابع normalize

این تابع یک string به عنوان text ورودی دریافت میکند و در ۵ گام نرمال‌سازی را اعمال میکند. در گام اول کاراکترهای خاص از متن حذف میشوند(مانند علائم نگارشی). در گام دوم "می" و "نمی" از افعال جدا میشوند. در گام سوم اعداد غیر فارسی به فارسی تبدیل میشوند. در گام چهارم برخی Unicode ها مثل "ریال" جایگزین میشوند و در گام آخر فاصله گذاری ها تصحیح میشود. در ادامه به توضیح هرگام و تابع متناظرش میپردازیم.

```
  # general normalization method to perform all above functions
  def normalize(cls, text:str) -> str:
      text = cls.remove_special_chars(text)
      text = cls.seperate_mi(text)
      text = cls.persian_number(text)
      text = cls.unicode_replacement(text)
      text = cls.spacing_correction(text)
      return text
```

تابع remove_special_chars

```
  # remove puctuation marks and arabic chars
  def remove_special_chars(cls, text: str) -> str:
      text = cls.remove_punc_marks(text)
      text = cls.remove_arabic_chars(text)
      return text

  # remove some arabic chars
  def remove_arabic_chars(cls, text: str) -> str:
      return cls.regex_replace(cls.arabic_patterns, text)

  # remove puctuation marks
  def remove_punc_marks(cls, text: str) -> str:
      return re.sub(cls.all_punc_marks, "", text)
```

این تابع در دو مرحله به حذف کاراکترهای خاص میپردازد. در مرحله اول با استفاده از تابع remove_punc_mark علائم نگارشی را از text حذف میکنیم. پترن‌های مورد بررسی در این بخش به شکل زیرند.

```
self.all_punc_marks = r"[\.:!،؛؟»\`\]\)\}|«\[\(\/\{><+\-?!=_]"
```

در مرحله دوم علائم متعلق به زبان عربی حذف یا جایگزین میشوند. فتحه، کسره، سکون و ... از متن حذف میشوند. میدانیم برخی حروف مانند ی به دو شکل ی و ی استفاده می‌شوند، در نرمالسازی از نسخه فارسی حروف استفاده میکنیم و با حرف عربی جایگزینش میکنیم.
پترن‌های مورد بررسی در این بخش به شکل زیرند.

```
self.arabic_patterns = [
            ("[\u064b\u064c\u064d\u064e\u064f\u0650\u0651\u0652]", ""),
            ("[ك]",'ک'),
            ("[ی]","ی"),
            ("[ة]","ه"),
            ("[أ]","ا"),
```

تابع separate_mi

```
  # seperate mi in start of verbs
  def seperate_mi(cls, text:str) -> str:
      matches = re.findall(cls.mi_patterns, text)
      for m in matches:
          r = re.sub("^(می?ن)", r"\1", m)
          # remove mi from token to check it contains the bon of a verb or not
          x = re.sub("^(می?ن)", "", m)
          for verb in cls.present_bons:
              if verb in x:
                  text = text.replace(m, r)
          for verb in cls.past_bons:
              if verb in x:
                  text = text.replace(m, r)
      return text
```

در این تابع ابتدا تمام کلماتی که با "می" یا "نمی" شروع میشوند جدا میشوند. "می" و "نمی" از این کلمات حذف شده و بررسی میشود آیا بخش باقی مانده با یک بن فعل (ماضی یا مضارع) شروع میشود یا خیر. در صورتی که ادامه کلمه با بن فعل شروع شود یعنی با احتمال بالا آن کلمه فعل بود که "می" یا "نمی" به شکل چسبیده به آن نوشته شده بوده، بنابراین بین "می" و ادامه‌ی کلمه نیم فاصله قرار میدهیم.

تابع Persian_number

```
def persian_number(cls, text: str) -> str:
    translation_table = str.maketrans(
        cls.number_not_persian,
        cls.number_persian )
    translated_text = text.translate(translation_table)
    return translated_text
```

این تابع ارقام غیرفارسی را به فارسی برمیگرداند. برای این بازگردانی از دو رشته هم طول استفاده میکنیم که کاراکترهای هم‌مکان ترجمه یکدیگرند و باهم جایگزین میشوند. با استفاده از این دو رشته یک جدول ترجمه ساخته میشود و متن توسط این جدول ترجمه پردازش شده و در صورت match شدن یک حرف با این جدول، با ترجمه اش که عدد به فارسی است جایگزین میشود.

رشته های استفاده شده در ساخت جدول ترجمه به شکل زیرند.

```
self.number_not_persian = "0123456789٪٠١٢٣٤٥٦٧٨٩"
self.number_persian = "٠١٢٣۴۵۶٧٨٩٪٠١٢٣۴۵۶٧٨٩"
```

تابع Unicode_replacement

```
  def unicode_replacement(cls, text: str) -> str:
      for old, new in cls.unicode_replacements:
          text = re.sub(old, new, text)
      return text
```

در این تابع از لیستی شامل جفت‌هایی از Unicode و توکن متناظرش استفاده میشود تا Unicode ها جایگزین شوند. لیست مورد استفاده به شکل زیر است و برای مثال " ﷽" با " بسم الله الرحمن الرحیم " جایگزین می‌شود.

```
# some special unicodes which should be replaced by persian terms
self.unicode_replacements = [
        ("﷽", " بسم الله الرحمن الرحیم "),("﷼", " ریال "),("ﷺ|ﷹ", " صلی "),("ﷲ", " الله "),("ﷳ", " اکبر "),
        ("ﷴ", " محمد "),("ﷵ", " صلعم "),("ﷶ", " رسول "),("ﷷ", " علیه "),("ﷸ", " وسلم "),("ﻵ|ﻶ|ﻷ|ﻸ|ﻹ|ﻺ|ﻻ|ﻼ", " لا ")]
```

تابع spacing_correction

```
# fix spacings
def spacing_correction(self, text: str) -> str:
    text = self.regex_replace(self.extra_space_patterns, text)
    text = self.regex_replace(self.punctuation_spacing_patterns, text)
    text = self.regex_replace(self.spacing_patterns, text)
    return text
```

این تابع از سه لیست پترن استفاده میکند. هر لیست بخشی از مشکلات فاصله‌گذاری در متن را رفع میکند. برای مثال لیست spacing_pattern شامل موارد زیر است.

- مورد اول non-breaking char را با space جایگزین میکند.
- مورد دوم "ی" ربط را با نیم فاصله به کلمه قبل متصل میکند.
- مورد سوم "می" و "نمی" را با نیم فاصله به کلمه بعد متصل می‌کند.
- مورد چهارم پسوندهای خاص فارسی را با یک کاراکتر zero-width non-joiner قبل از خود پسوند جایگزین می‌کند مثلا "بزرگ تر" تبدیل به "بزرگ‌تر" می‌شود. "تر" و "ترین" و "های" و "گری" از مواردی هستند که توسط این پترن هندل میشوند.
- مورد پنجم در کلماتی مانند "خانه ام" کاربرد دارد که کلمه اصلی در انتهایش یک "ه" دارد و ضمیر متصل به آن با space قرار گرفته. در این شرایط ضمیر با نیم‌فاصله به کلمه قبل متصل خواهد شد.
- مورد اخر برای کلماتی مثل "کلمهها" است که بین دو "ه" نیم فاصله قرار می‌دهد و توکن به "کلمه‌ها" تبدیل خواهد شد.

```
    # space patterns for nimfasele
    self.spacing_patterns = [
        (r"\xa0"," "),  # remove no-break char
        (r"([^ ]) ی ", r"\1‌ی "),          # fix 'ی' space
        (r"(^| )(ن?می) ", r"\1\2‌"),         # fix 'می' and 'نمی' space
        (r"(?<=[^\n\d" + self.punc_after + self.punc_before + r"]{2}) (تر(ین?)?|گری?|های?)(?=[ \n" + self.punc_after + self.punc_before + r"]|$
        # fix suffix spacing
        (r"([^ ]ه) ((ا(م|یم|ش|ند|ی|ید|ت))(?=[ \n" + self.punc_after + r"]|$)", r"\1‌\2"),  # fix verb conjugation spacing
        (r"(ه)(ها)", r"\1‌\2"),
    ]
```

نمونه ای از تست normalizer

Normalization Test

```
normalizer = DataNormalization()
print(normalizer.seperate_mi("میرفتم"))
print(normalizer.spacing_correction("به نام های خدا ی  درباره ی مهربان ترین"))
print(normalizer.normalize("به نام های ﷲ خدا ی 99 % درباره ی ٌسلامَ ُ ِ ° +ئئئ  مقابله أ ﷽ مهربان <. ﷽  . ترین"))
print(normalizer.normalize("«العين التي تمتل بك لن تنظر لغيرك...»"))
print(normalizer.normalize("'ابقَ قويًا، خالهٔ فَقِصّتُكَ لم تَنتَهي بعد ..' "))

میرفتم
به نام‌های خدای درباره‌ی مهربان‌ترین
به نام‌های الله خدای ۹۹ ٪ درباره‌ی سلام ئئئ مقابله ا بسم الله الرحمن الرحیم مهربان بسم الله الرحمن الرحیم‌ترین
العین التی تمتل بک لن تنظر لغیرک
ابق قویا خالهه فقصتک لم تنتهی بعد
```

کلاس DataPreprocessing

در این کلاس توابعی برای اعمال پیش‌پردازش‌ها روی متون پیاده شده.

تابع preprocess

```python
# preprocess all given docs
def preprocess(self, docs):
    tokens = []
    self.print_top_k()
    counter = 0
    for idx in docs.keys():
        content = docs[str(idx)]['content']
        punctuated_content = self.Remove_Punctuations(content)
        normalized_content = self.Normalization(punctuated_content)
        all_tokens = self.Tokenization(normalized_content)
        stemmed_tokens = self.Stemming(all_tokens)
        docs[str(idx)]['content'] = stemmed_tokens
        tokens += stemmed_tokens
        counter += 1
        # print progress
        if counter % 1000 == 0:
            print(counter, ' docs processed')
    # save top k frequent
    self.top_k = self.Top_K_Frequent(tokens, 20)
    # remove stop words from doc tokens
    for doc_id, doc_content in docs.items():
        docs[doc_id]['content'] = [token for token in doc_content['content'] if token not in self.top_k]
    return docs
```

عملیات این تابع به طور خلاصه به ترتیب زیر است:

۱. یک لیست به نام tokens ایجاد می‌شود که توکن‌های نهایی همه‌ی داکیومنت ها در آن ریخته خواهد شد.

۲. متن‌های ورودی docs که یک دیکشنری است و کلیدهای آن id خبر است، یکی یکی بررسی می‌شوند.

۳. متن مربوط به هر id از docs ابتدا با تابع remove_punctuation علائم نگارشی‌اش حذف می‌شود.

۴. سپس متن نرمال‌سازی می‌شود.

۵. تمامی توکن‌های متن نرمال‌شده استخراج می‌شود.

۶. توکن‌ها با استفاده از کتابخانه parsivar ریشه‌یابی می‌شوند. 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.

۷. content هر docs با توکن‌های نهایی جایگزین می‌شود.

۸. توکن‌های هر داک به لیست tokens اضافه می‌شوند.

۹. بعد از پردازش هر ۱۰۰۰ متن، پیشرفت کار چاپ میشود.

۱۰. با استفاده از تابع Top_K_Frequent، توکن‌های با بیشترین تکرار مشخص می‌شود.

۱۱. توکن‌های بیشترین تکرار از لیست توکن‌های تمامی داکیومنت‌ها حذف می‌شوند.

در ادامه به شرح توابع استفاده شده در این بخش میپردازیم.

تابع Remove_Punctuation

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

وظیفه‌ی تابع حذف general علائم نگارشی از متن ورودی است. در پایتون، کتابخانه string مجموعه‌ای از علائم نگارشی را در متغیر punctuation تعریف می‌کند. با استفاده از این متغیر، می‌توان به طور خودکار علائم نگارشی را در یک متن شناسایی و حذف کرد.

تابع Normalization

```
#Normalization
@staticmethod
def Normalization(text):
    my_normalizer = DataNormalization()
    return my_normalizer.normalize(text)
```

این تابع با استفاده از کلاس DataNormalization که پیشتر تعریف کردیم به نرمالایز کردن متن ورودی میپردازد.

تابع Tokenization

```
@staticmethod
def Tokenization(text):
    text = DataPreprocessing.pattern.sub(r" \1 ", text.replace("\n", " ").replace("\t", " "))
    tokens = [word for word in text.split(" ") if word]
    tokens_cleaned = [token.strip('\xa0') for token in tokens if len(token.strip()) != 0]

    result = [""]
    # merge multi term verbs like خواهم رفت to خواهم_رفت
    for token in reversed(tokens_cleaned):
        if token in DataPreprocessing.before_verbs or (
            result[-1] in DataPreprocessing.after_verbs and token in DataPreprocessing.verbe
        ):
            result[-1] = token + "_" + result[-1]
        else:
            result.append(token)
    return list(reversed(result[1:]))
```

این تابع مراحل زیر را انجام می‌دهد:

۱. جایگزینی پترن خاصی که در DataPreprocessing.pattern تعریف شده‌اند با یک فاصله از هر طرف آنها. حذف tab و new_line و جایگزینی با space.

۲. تبدیل متن به توکن‌ها با استفاده از فاصله به عنوان مرز بین توکن‌ها.

۳. پاک‌سازی توکن‌های حاصل از مرحله قبل از کاراکتر `\xa0\` و حذف توکن‌های خالی.

۴. (قسمت اضافی) ترکیب توکن‌هایی که مربوط به فعل‌های چندبخشی هستند. برای مثال "خواهم" و "رفت" به "خواهم_رفت" تبدیل می‌شوند. این ترکیب با استفاده از دو لیست شامل افعال پیشوندی مثل "خواهم"،"خواهی" و ... و افعال پسوندی مانند "بودم"،"است" و ... صورت میگیرد که با توکن قبل و بعدشان مقایسه میشوند.

تابع top_k_frequent

```
def Top_K_Frequent(tokens,k):
    token_counts = Counter(tokens)
    sorted_tokens = sorted(token_counts.items(), key=lambda x: x[1], reverse=True)
    stopwords_to_remove = [token for token, count in sorted_tokens[:k]]
    report = {token: count for token, count in sorted_tokens[:k]}
    return report

# print top k frequent terms
def print_top_k(self):
    for token, count in self.top_k.items():
        print(f"Token: {token}, Count: {count}")
```

در این تابع روی توکن‌های تمامی داکیومنت ها پردازش صورت میگیرد و با استفاده از Counter تعداد تکرار هر توکن استخراج می‌شود. این لیست سورت شده و k تا توکن اول که بیشترین تکرار را دارند بازگردانده می‌شوند. تابعی برای print نتایج این بخش نیز به عنوان print_top_k تعریف شده است.

اعمال پیش‌پردازش روی داکیومنت‌ها

* در دستورکار گفته شده ۵۰ کلمه پر تکرار حذف شوند اما من با صحبت با تدریسار و بررسی لیست ۲۰ تا توکن پرتکرار را حذف کردم چون در ۳۰ تای دوم کلمات مهمی حذف می‌شدند مانند "فارس" و "ایران".

```
global preprocessor
preprocessor = DataPreprocessing()
```

Load and Preprocess Docs

```
docs, contents, urls = load_docs()
```

```
pre_processed_docs = preprocessor.preprocess(docs)
```

```
1000  docs processed
2000  docs processed
3000  docs processed
4000  docs processed
5000  docs processed
6000  docs processed
7000  docs processed
8000  docs processed
9000  docs processed
10000  docs processed
11000  docs processed
12000  docs processed
```

لیست ۲۰ stop word حذف شده

Print Stop Words

```
preprocessor.print_top_k()
```

```
Token: و, Count: 219205
Token: در, Count: 164329
Token: به, Count: 133277
Token: از, Count: 92930
Token: این, Count: 82976
Token: که, Count: 76240
Token: با, Count: 68994
Token: را, Count: 67489
Token: کرد&کن, Count: 45104
Token: اس, Count: 44652
Token: برای, Count: 30996
Token: داشت&دار, Count: 30052
Token: تیم, Count: 27692
Token: شد&شو, Count: 26575
Token: کرد, Count: 22936
Token: هم, Count: 22393
Token: کشور, Count: 21730
Token: ما, Count: 19717
Token: یک, Count: 18733
Token: بود&باش, Count: 18028
```

این کلمات به شدت پرتکرارند اما اطلاعات خاصی از آنها استخراج نمیشود بنابراین حذف میشوند. ضمنا در صورتی که حذف نشوند posting_list های بسیار طولانی برای آنها خواهیم داشت که حجم دیکشنری را بالا میبرد.

۳-۱ ساخت شاخص مکانی

در این مرحله با تعریف تابع postings_list به ساخت شاخص معکوس مکانی میپردازیم. در این شاخص لیست قهرمان را نیز با ۲۰ داکیومنت برای هر term ذخیره میکنیم.

وظایف تابع به دو بخش تقسیم میشود:

بخش اول) ساخت دیکشنری

```
my_dict = {}
for index in Docs:
    for position, token in enumerate(Docs[index]['content']):
        # if its not new token
        if token in my_dict:
            # if the doc has already been in posting list of that token
            if index in my_dict[token]['docs']:
                my_dict[token]['docs'][index]['positions'].append(position)
                my_dict[token]['docs'][index]['number_of_token'] += 1
            else:
                my_dict[token]['docs'][index] = {
                                                 'positions': [position],
                                                 'number_of_token': 1
                                                 }
            my_dict[token]['frequency'] += 1

        # add token to dictionary if its new
        else:
            my_dict[token] = {
             'frequency': 1,
             'docs': {
                   index: {
                       'positions': [position],
                       'number_of_token': 1
                       }
                }
            }
```

ورودی تابع ساخت شاخص مکانی کل اسناد پیش پردازش شده است.

روی اسناد پیمایش کرده و برای هر سند روی توکن های آن(که در قسمت contentهستند) عملیات زیر را انجام میدهیم:

1-اگر توکن در دیکشنری ما بود:

1-1-اگر سند در سند هایی که توکن در آن ها بوده از قبل در دیکشنری باشد تنها لازم است به جایگاه فعلی آن را به جایگاه توکن در سند اضافه کنیم و همچنین به تعداد تکرار کلمه در سند یکی اضافه کنیم.

2-1-اگر سند در posting list توکن هنوز اضافه نشده باشد آن را اضافه میکنیم و position اول را نیز برای ش قرار میدهیم و تعداد تکرار کلمه در سند را یک قرار میدهیم ضمنا به تعداد تکرار کلمه در کل اسناد یکی اضافه میکنیم.

2-اگر توکن در دیکشنری از قبل نباشد، تعداد تکرار کلمه در کل اسناد را یک قرار داده و در قسمت posting list آن سند را اضافه کرده و اطلاعات (جایگاه در سند و تعداد تکرار ۱) را ایجاد میکنیم.

نکات خروجی:

برای هر term در دیکشنری یک ’frequency‘ داریم که تعداد داکیومنت هایی است که این term در آنها قرار دارد.

برای هر doc در posting_list یک term، لیستی به نام positions داریم که مکانهایی که term در آن سند قرار گرفته را در آن ذخیره میکنیم.

برای هر doc در posting_list یک term، number_of_token که تعداد تکرار term در آن سند است را نگه می‌داریم.

بخش دوم) ساخت بردار tf_idf برای داکیومنت‌ها- ساخت لیست قهرمان برای term ها

```
    N = len(Docs)
    docs_vectors = {}

    for term in my_dict:
        term_docs = dict(my_dict[term]['docs'])
        n_t = len(term_docs)
        # calculting tf_idf for each doc in posting list
        for doc in my_dict[term]['docs']:
            tf = my_dict[term]['docs'][doc]['number_of_token']
            x = (np.log10( N / n_t ))*(1+np.log10(tf))
            my_dict[term]['docs'][doc]['tf_idf'] = x

            # add weight of this term to docs vector for future usages in query processing
            if doc not in docs_vectors:
                docs_vectors[doc] = {}
            docs_vectors[doc][term] = {'tf_idf':x,'tf':tf}


        # sort posting list and put it in champion list of each term
        sorted_term_docs = sorted(term_docs, key=lambda doc: term_docs[doc]['number_of_token'], reverse=True)
        my_dict[term]['champions_list'] = {}
        for doc_number in sorted_term_docs:
            my_dict[term]['champions_list'][doc_number] = {'number_of_token': my_dict[term]['docs'][doc_number]['number_of_token']
                                                           ,'tf_idf' : my_dict[term]['docs'][doc_number]['tf_idf']}

        # if champ_len is smaller than actual postings, we extract top champ_len of sorted postings
        if champ_len < n_t:
            my_dict[term]['champions_list'] = dict(list(my_dict[term]['champions_list'].items())[:champ_len])

    return my_dict , docs_vectors
```

۱. برای هر ترم:

- داکیومنت هایی که ترم در آن وجود دارد در term_docs و تعدادشان در n_t قرار داده می‌شود.

برای هر داکیومنت در term_docs:

- مقدار TF-IDF برای هر داکیومنت در term_docs با استفاده از فرمول TF-IDF محاسبه می‌شود و به هر داکیومنت در دیکشنری اضافه می‌شود.N در فرمول IDF تعداد داکیومنت هاست که با len(docs) به دست آمده.

- میخواستیم بردار TF-IDF برای هر داکیومنت را نیز بسازیم. اگر document بردارش ساخته نشده باشد آن را به لیست بردار ها اضافه میکنیم. TF_IDF مربوط به این term را در بردار داک را اضافه میکنیم. از این بردار ها در نرمال‌سازی طول بردار ها در محاسبه شباهت کسینوسی استفاده میشود.

۲. لیست قهرمانان (champions list) برای هر ترم ایجاد می‌شود:

- تمام داکیومنت ها براساس تعداد توکن‌های ترم در هر document از posting list مرتب می‌شوند.

- این لیست مرتب شده از داکیومنت ها در my_dict[term]['champions_list'] قرار می‌گیرد.

۳. اگر تعداد مستندهای مرتبط با یک ترم n_t بیشتر از champ_len (تعداد مستندهایی که در لیست قهرمانان برای هر ترم استفاده می‌شود) باشد، فقط به تعداد champ_len از داکیومنت های مرتب شده در لیست قهرمانان باقی می‌ماند و سایرین حذف می‌شوند.

در نهایت با فراخوانی این تابع، دیکشنری ساخته میشود.

Create Dictionary

```
global dictionary, docs_vectors
dictionary, docs_vectors = Postings_List(pre_processed_docs, 20)
```

تعداد term های درون دیکشنری

```
Dictionary Length

21]:  1 len(dictionary)

21]:  50440
```

محاسبه حجم دیکشنری با استفاده از کتابخانه pickle

```python
import pickle
db = {}
db['dictionary_12k'] = dictionary

# Its important to use binary mode
dbfile = open('dictionary_12k', 'ab')

# source, destination
pickle.dump(db, dbfile)
dbfile.close()
```

بعد از اجرای قطعه کد ذخیره ی دیکشنری، فایل dictionary_12k ایجاد میشود که حجم آن ۸۵,۷ مگابایت است.

نمونه‌ای از محتوای دیکشنری

```
23]:  1 dictionary['استراماچونی']

23]:  {'frequency': 87,
       'docs': {'143': {'positions': [5, 33, 127, 228],
         'number_of_token': 4,
         'tf_idf': 3.8248557751020287},
        '208': {'positions': [141],
         'number_of_token': 1,
         'tf_idf': 2.38746101632035},
        '325': {'positions': [7], 'number_of_token': 1, 'tf_idf': 2.38746101632035},
        '540': {'positions': [149],
         'number_of_token': 1,
         'tf_idf': 2.38746101632035},
        '599': {'positions': [48], 'number_of_token': 1, 'tf_idf': 2.38746101632035},
        '917': {'positions': [37], 'number_of_token': 1, 'tf_idf': 2.38746101632035},
        '951': {'positions': [52], 'number_of_token': 1, 'tf_idf': 2.38746101632035},
        '965': {'positions': [25, 73],
         'number_of_token': 2,
         'tf_idf': 3.106158395711893},
```

۲-پاسخ‌دهی به پرسمان در فضای برداری

در بخش ساخت ماتریس مکانی، vector های tf_idf برای داکیومنت ها را نیز ساختیم. در بخش پاسخ دهی به پرسمان‌ها برای query نیز بردار tf_idf را محاسبه میکنیم و این بردار را با بردار داکیومنت‌هایی که آن query را دارند مقایسه میکنیم. مقایسه توسط تابع شباهت کسینوسی صورت میگیرد. داکیومنتهایی که بیشترین شباهت کسینوسی را داشته باشند در rank های بالاتر برمیگردند.

تابع vector_length

```
def vector_length(vector_dict):
    length = math.sqrt(sum(tf_idf_value['tf_idf'] ** 2 for tf_idf_value in vector_dict.values()))
    return length
```

ورودی این تابع بردار tf_idf های یک داکیومنت است. برای نرمالسازی بردار ها حین ضرب دات نیاز است نتیجه را به طولشان تقسیم کنیم. محاسبه طول بردار داکیومنت ها توسط این تابع صورت میگیرد که اگر V = {v1,v2,...,vn} باشد طول طبق فرمول $\sqrt{\sum {v_i}^2}$ محاسبه می‌شود.

تابع calculate_tf_idf

```
import math
def calculate_tf_idf(f_td, N, n_t):
    tf = 1 + np.log10(f_td)
    idf = np.log10(N / n_t)
    return tf * idf
```

این تابع برای محاسبه tf_idf برای یک ترم در یک داکیومنت به کار میرود. ورودی های ان f_td (تعداد تکرار آن term در آن داکیومنت)، N (تعداد کل داکیومنت های دیتاست) و n_t (تعداد داکیومنت‌هایی که این term را دارند) است. Tf_idf طبق فرمول زیر محاسبه شده و بازگردانده میشود. بخش idf به محاسبه میزان نادر بودن کلمه میپردازد و هرچه کلمه کمتر رایج باشد مقدار بیشتری دارد. بخش tf به میزان تکرار term در داکیومنت میپردازد چون توقع داریم هرچه query در داکیومنتمان بیشتر تکرار شده باشد داکیومنت مرتبط تر باشد.

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

برای امتیاز دهی به داکیومنت ها بر اساس یککوئری ورودی از الگورتم زیر استفاده میشود که در تابع query_scoring پیاده سازی شده است.

Computing cosine scores

CosineScore(q)

```
1   float Scores[N] = 0
2   float Length[N]
3   for each query term t
4   do calculate w_{t,q} and fetch postings list for t
5       for each pair(d, tf_{t,d}) in postings list
6       do Scores[d] += w_{t,d} × w_{t,q}
7   Read the array Length
8   for each d
9   do Scores[d] = Scores[d]/Length[d]
10  return Top K components of Scores[]
```

```
[19]: def query_scoring(query, total_number_of_docs, dictionary, k, champion_list = False):
          cosine_scores = {}
          jaccard_scores = {}
          query_tokens = preprocessor.simple_preprocess(query)
          query_tokens_count = dict(collections.Counter(query_tokens))
          print(query_tokens_count)
          query_terms_num = sum(query_tokens_count.values())
          for term in query_tokens_count:
              if term in dictionary:
                  if champion_list:
                      term_docs = dictionary[term]['champions_list']
                  else:
                      term_docs = dictionary[term]['docs']
                  w_tq = calculate_tf_idf(query_tokens_count[term], total_number_of_docs, len(term_docs))
                  for doc in term_docs:
                      w_td = term_docs[doc]['tf_idf']
                      #update doc scores for cosines similarity
                      if int(doc) in cosine_scores:
                          #update doc scores for cosines similarity
                          cosine_scores[int(doc)] += w_td * w_tq
                          #update doc scores for jaccard similarity
                          jaccard_scores[int(doc)] += 1
                      else:
                          cosine_scores[int(doc)] = w_td * w_tq
                          #update doc scores for jaccard similarity
                          jaccard_scores[int(doc)] = 1
          # calculate cosine score by dividing by doc vector length
          for doc_number in cosine_scores:
              cosine_scores[doc_number] /= vector_length(docs_vectors[str(doc_number)])
          # calculate jaccard score
          for doc_number in jaccard_scores:
              jaccard_scores[doc_number] /= (len(pre_processed_docs[str(doc_number)]['content']) + query_terms_num - jaccard_scores[doc_number])
          # sort scores for top k
          sorted_doc_cosine = sorted(cosine_scores.items(), key=lambda x:x[1], reverse=True)
          sorted_doc_jaccard = sorted(jaccard_scores.items(), key=lambda x:x[1], reverse=True)
          return sorted_doc_cosine[:k], sorted_doc_jaccard[:k]
```

در دستور کار شباهت کسینوسی معیار قرار گرفته اما به عنوان تمرین بیشتر، هم شباهت کسینوسی و هم jaccard را محاسبه کردیم.

مراحل کار تابع query_scoring به شرح زیر است:

۱- توکن‌های کوئری با استفاده از تابع simple_preprocess از کلاس DataPreprocessing استخراج و تعداد تکرار هر کدام شمرده میشود و در query_tokens_count قرار میگیرد.

۲- به ازای هر توکن یکتا(term) در query اگر در دیکشنری‌مان قرار داشته باشد:

- Posting_list آن term را در term_docs قرار می‌دهیم. اگر در scoring مشخص شده باشد که از champions list استفاده شود، به جای posting_list از champions_list که ۲۰ داکیومنت با بیشترین تعداد تکرار term در آنهاست استفاده میکنیم.
- Tf_idf برای term در کوئری را توسط تابع calculate_tf_idf محاسبه کرده و در w_tq ذخیره میکنیم.

به ازای هر doc در term_docs:

- در w_td مقدار tf_idf برای term در doc را ذخیره میکنیم. به شباهت کسینوسی مقدار w_td*w_tq و به شباهت jaccard مقدار ۱ را اضافه میکنیم.

۳- ضرب w_td*w_tq در واقع همان ضرب دات بردارهاست. باید این بردارها نرمال سازی شوند یعنی به طول بردار tf_id برای داکیومنت و query تقسیم شوند. طول بردار کوئری ثابت است پس تاثیری در رتبه بندی ندارد اما برای هر داکیومنت طول بردار متفاوت است پس امتیاز هر داکیومنت را بر طول بردار آن داکیومنت تقسیم میکنیم.

۴- برای شباهت jaccard، اشتراک مجموعه term های کوئری و داکیومنت به اجتماعشان باید تقسیم شود. اجتماع همان تعداد term های کوئری به علاوه تعداد term های داکیومنت منهای اشتراکشان است.

۵- امتیاز ها را به ترتیب نزولی sort میکنیم و k داکیومنت برتر در هر نوع امتیاز را برمیگردانیم.

تابع query_search

```
def query_search(query, result_numbers = 5, champion_list = False):
    results1, results2 = query_scoring(query, len(docs), dictionary, result_numbers, champion_list)
    # print(results1)
    if len(results1) == 0 and len(results2) == 0:
        print("نتیجه ای یافت نشد")
    else:
        print("Cosine Scores:")
        print(100*'=')
        r1 = print_results(results1)
        print()
        print("Jaccard Scores:")
        print(100*'=')
        r2 = print_results(results2)
    return r1, r2
```

این تابع query، تعداد خروجی و استفاده از لیست قهرمان را به عنوان ورودی دریافت میکند. با استفاده از تابع query_scoring نتیجه امتیازدهی به داکیومنت ها را دریافت کرده و برای هر دو معیار کسینوسی و jaccard نتایج را چاپ میکند. چاپ نتایج با تابع print_results صورت میگیرد.

تابع print_results

```
def print_results(results):
    dict_result = {}
    for rank, result in enumerate(results):
        doc_id = result[0]
        if doc_id == None:
            continue
        print(100*'-' + '\n')
        print(f'Rank: {rank + 1}')
        print(f'ID: {doc_id}')
        print(f'{docs[f"{doc_id}"]["title"]}')
        print(f'{docs[f"{doc_id}"]["url"]}')
        dict_result[rank + 1] = {'docID': doc_id,
                                 'title': docs[str(doc_id)]["title"],
                                 'url'  : docs[str(doc_id)]["url"]}
    return dict_result
```

این تابع در لیست نتایج، عنوان و رتبه و لینک خبر را نمایش میدهد.

نمونه ای از نحوه استفاده و نتایج:

```
[20]: r1, r2 = query_search('وزیر', result_numbers = 10, champion_list = False)

      {'1 :'وزیر}
      Cosine Scores:
      ==========================================================================================
      ------------------------------------------------------------------------------------------

      Rank: 1
      ID: 12191
      آغاز نشست غیرعلنی مجلس با حضور امیرعبداللهیان
      https://www.farsnews.ir/news/14000724000697/آغاز-نشست-غیرعلنی-مجلس-با-حضور-امیرعبداللهیان
      ------------------------------------------------------------------------------------------

      Rank: 2
      ID: 10893
      زارعی کوشا استاندار کردستان شد
      https://www.farsnews.ir/news/14000826000505/زارعی-کوشا-استاندار-کردستان-شد
      ------------------------------------------------------------------------------------------

      Rank: 3
```

۳-تست جست و جو

در جست و جو ها از لیست قهرمان استفاده شده و ۵ سند برتر نمایش داده شده.

الف)پرسمان ساده از کلمه متداول

کلمه وزیر

```
[22]: query_search('وزیر', result_numbers = 5, champion_list = True)

      {'وزیر': 1}
      Cosine Scores:
      ==========================================================================================
      ------------------------------------------------------------------------------------------

      Rank: 1
      ID: 10268
      دستور کار کمیسیون‌های مجلس/ دیدار نمایندگان با وزیر خارجه و بررسی تخلفات روحانی و زنگنه در انعقاد قرارداد کرسنت
      https://www.farsnews.ir/news/14000910000920/دستور-کار-کمیسیون‌های-مجلس-دیدار-نمایندگان-با-وزیر-خارجه-و-بررسی
      ------------------------------------------------------------------------------------------

      Rank: 2
      ID: 10890
      سومین وزیر پیشنهادی آموزش و پرورش در ایستگاه بهارستان/ از سرعت عمل رئیس‌جمهور در معرفی تا گزینه‌ای از بدنه فرهنگیان
      https://www.farsnews.ir/news/14000826000559/سومین-وزیر-پیشنهادی-آموزش-و-پرورش-در-ایستگاه-بهارستان-از-سرعت-عمل
      ------------------------------------------------------------------------------------------

      Rank: 3
      ID: 7478
      اصلاحات بدون رتوش| مروری بر بدزبانی‌ها و اهانت‌های دولت روحانی به مردم و خبرنگاران
      https://www.farsnews.ir/news/14001203000851/اصلاحات-بدون-رتوش|-مروری-بر-بدزبانی‌ها-و-اهانت‌های-دولت-روحانی-به-مردم
      ------------------------------------------------------------------------------------------

      Rank: 4
      ID: 8171
      قوه قضاییه برای برخورد با اصلاح‌طلب هنای به اسلام و شهدا چه تصمیمی دارد؟
      https://www.farsnews.ir/news/14001110000053/قوه-قضاییه-برای-برخورد-با-اصلاح‌طلب-هنای-به-اسلام-و-شهدا-چه-تصمیمی
      ------------------------------------------------------------------------------------------

      Rank: 5
      ID: 7564
      هیات دولت با تشکیل «شورای سینما» موافقت کرد
      https://www.farsnews.ir/news/14001201001012/هیات-دولت-با-تشکیل-شورای-سینما-موافقت-کرد
```

نتیجه خبر اول

همانطور که مشاهده میشود در این خبر تعداد تکرارهای کلمه وزیر زیاد است چون از لیست قهرمان استفاده شده که در آن داکیومنت ها با بیشترین tf قرار دارندو اگر همین سرچ بدون لیست قهرمان انجام مید به احتمال زیاد رنک ۱ داکیومنتی کوتاه و با تعداد تکرار کمتر کلمه وزیر بود. خبر مستقیما به وزیر مرتبط نیست بلکه صرفا تعداد استفاده های وزیر در آن زیاد بوده. وزیر کلمه ای بسیار کلی است.

رسیدگی به گزارش دیوان محاسبات کشور درخصوص بودجه پیشنهادی سال ۱۴۰۱ شرکتهای دولتی، بانکها و موسسات انتفاعی وابسته به دولت با حضور رئیس کل دیوان محاسبات و معاونین وی، بحث و بررسی گزارش دیوان محاسبات درخصوص ثبت اطلاعات فردی و استخدامی کارکنان توسط دستگاههای اجرائی در سامانه پاکنا با حضور معاون رئیس جمهور و رئیس سازمان امور اداری استخدامی کشور بخشی از دستور کارهای کمیسیون برنامه و بودجه و محاسبات است.

دعوت از وزیر نفت جهت پاسخگویی به سوال جلیل مختار نماینده آبادان، بررسی مشکلات بیماران اس.ام.ای باحضور مسئولین وزارت بهداشت و نمایندگان بیماران، استماع برنامه های ریاست جمعیت هلال احمر و ادامه رسیدگی به طرح اصلاح اساسنامه جمعیت هلال احمر بخشی از دستور کارهای کمیسیون بهداشت و درمان است.

دعوت از وزیر صنعت، معدن و تجارت برای پاسخگویی به سئوالات نمایندگان، بررسی عملکرد شرکت ملی نفت ایران در خصوص توسعه میادین نفتی پارس جنوبی و میادین مشترک نفتی با حضور معاون وزیر و مدیر عامل شرکت بخشی از دستور کارهای کمیسیون صنایع و معادن است.

ادامه رسیدگی به طرح کاهش تعرفه بنگاه های معاملاتی، دعوت از وزیر نیرو در خصوص پاسخ به سوالات نمایندگان، رسیدگی به لایحه اصلاح قانون توسعه حمل و نقل عمومی و مدیریت مصرف سوختی بخشی از دستور کارهای کمیسیون عمران است.

دعوت از وزیر ورزش و جوانان جهت پاسخ گویی به سوالات نمایندگان، بحث و بررسی پیرامون چگونگی اجرای قانون جوانی جمعیت و حمایت از خانواده بخشی از دستور کارهای کمیسیون فرهنگی است.

ادامه بررسی طرح ممنوعیت خروج مسئولین و مدیران نظام جمهوری اسلامی پس از اتمام مسئولیت از کشور تا سپری شدن مراحل قانونی، ادامه بررسی طرح یک فوریتی حمایت مالی از افشاگران فساد بخشی از دستور کارهای کمیسیون قضائی و حقوقی بخشی از دستور کارهای کمیسیون قضائی و حقوقی است.

دعوت از وزیر نفت جهت شرکت در جلسه هم اندیشی تهاتر محصولات نفتی با محصولات کشاورزی، دعوت از وزیر جهاد کشاورزی جهت پاسخگویی به سئوالات نمایندگان، بررسی موضوع نحوه خرید و فروش و عرضه دام با حضور وزیر کشاورزی و مدیرعامل شرکت پشتیبانی امور دام بخشی از دستور کارهای کمیسیون کشاورزی، آب، منابع طبیعی و محیط زیست است.

به گزارش خبرنگار پارلمانی خبرگزاری فارس، دستور کارهای کمیسیون‌های مجلس شورای اسلامی در هفته جاری اعلام شد.

بر همین اساس، ادامه بررسی طرح اجرای اصل پانزدهم (۱۵) قانون اساسی جمهوری اسلامی ایران (تدریس زبانهای محلی و قومی در مدارس و دانشگاههای کشور) و ارائه گزارش وزیر علوم، تحقیقات و فناوری درخصوص اقدامات انجام شده آن وزارت در ۱۰۰ روز ابتدایی دولت سیزدهم بخشی از دستور کارهای کمیسیون آموزش، تحقیقات و فناوری است.

بررسی تغییر عنوان مصوبه کمیسیون در طرح تسری فوق العاده خاص کارمندان سازمان های پزشکی قانونی کشور و انتقال خون ایران به کارکنان قوه قضائیه، ادامه بررسی طرح ساماندهی استخدام کارکنان دولت، جلسه کمیته اشتغال، کارآفرینی و روابط کار کمیسیون اجتماعی به منظور بررسی طرح مدیریت تعارض منافع بخشی از دستور کارهای کمیسیون اجتماعی است.

بررسی مشکلات اقتصادی صنعت تولید و صادرات فرش، ادامه بررسی طرح مالیات بر عایدی سرمایه، نشست اعضای کمیسیون اقتصادی با دبیر شورای عالی مناطق آزاد تجاری- صنعتی و ویژه اقتصادی به منظور بررسی برنامه‌های پیشنهادی دبیرخانه شورای عالی مناطق آزاد تجاری- صنعتی کشور و ارائه گزارش کمیته تخصصی کمیسیون اقتصادی مجلس در خصوص طرح اصلاح قانون چگونگی اداره مناطق آزاد تجاری صنعتی

انتخاب اعضای هیأت تحقیق و تفحص از عملکرد بانک های دولتی و بانکهای واگذار شده در اجرای اصل ۴۴ قانون اساسی بخشی از دستور کارهای کمیسیون اقتصادی است.

نشست مشترک اعضای کمیسیون با وزیر امور خارجه و معاونان وی، بررسی طرح تشکیل پیمان دفاعی امنیتی گروه مقاومت بخشی از دستور کارهای کمیسیون امنیت ملی و سیاست خارجی است.

کارگروه بررسی طرح نحوه تشکیل اجتماعات و برگزاری راهپیمایی ها، بررسی طرح اصلاح جدول حوزه های انتخابیه مجلس، ادامه بررسی طرح اصلاح موادی از قانون انتخابات شوراهای اسلامی کشور و انتخاب شهرداران بخشی از دستور کارهای کمیسیون امورداخلی کشور و شوراهاست.

گزارش عملکرد معاون وزیر نفت و مدیرعامل شرکت ملی صنایع پتروشیمی درخصوص آخرین دستاوردها، فعالیتها و اقدامات و نیز برنامه های آتی این شرکت، بررسی و مکان سنجی صدور خدمات فنی و مهندسی جهت نیروگاهها و پالایشگاه های خارجی، نشست مشترک با شرکت ملی نفت ایران و شرکت ملی گاز ایران جهت بررسی وضعیت گازهای مشعل و ساماندهی آنها، جمع آوری، فروش و ظرفیت سازی گازهای LPG، بررسی سوالات نمایندگان از وزیر نفت، بررسی گزارش اعمال ماده ۲۳۴ قانون آیین نامه داخلی مجلس شورای اسلامی درخصوص استنکاف ریاست جمهوری سابق و وزارت نفت سابق از توسعه میادین مشترک نفت و گاز، بررسی درخواست تعدادی از نمایندگان جهت اعمال ماده ۲۳۴ قانون آیین نامه داخلی مجلس شورای اسلامی درخصوص استنکاف و عدم اجرای قانون و عدم رعایت شئونات توسط ریاست جمهوری سابق و وزارت نفت سابق پیرامون قرارداد فروش گاز به شرکت کرسنت بخشی از دستور کارهای کمیسیون انرژی است.

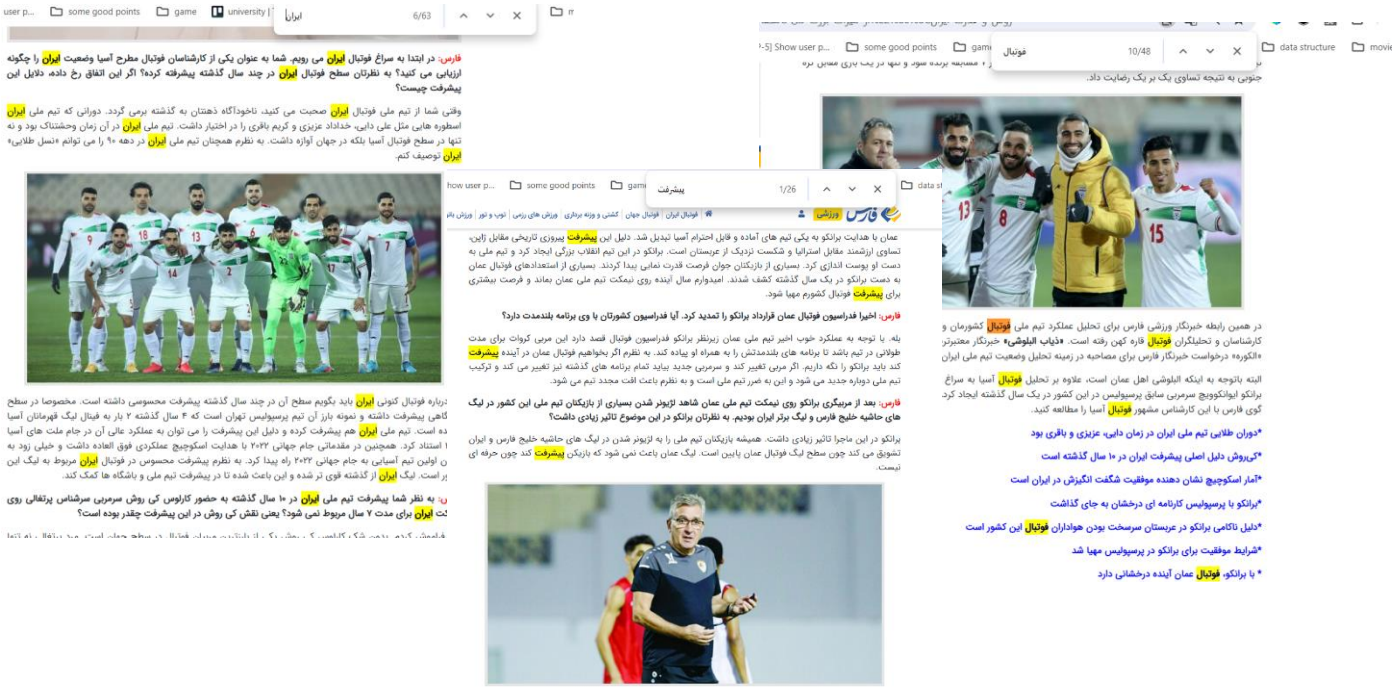
ب)پرسمان ساده از چند کلمه متداول

عبارت پیشرفت فوتبال ایران

```
[28]: query_search('پیشرفت فوتبال ایران', result_numbers = 5, champion_list = True)

{'پیشرفت': 1, 'فوتبال': 1, 'ایران': 1}
Cosine Scores:
================================================================================================
------------------------------------------------------------------------------------------------

Rank: 1
ID: 2098
مصاحبه فارس با کارشناس فوتبال آسیا| از میراث بزرگ کوروش و قدرت ایران با اسکوچیچ تا انقلاب برانکو در عمان
https://www.farsnews.ir/news/14001124000522/مصاحبه-فارس-با-کارشناس-فوتبال-آسیا|-از-میراث-بزرگ-کوروش-و-قدرت-ایران
------------------------------------------------------------------------------------------------

Rank: 2
ID: 9728
تولید واکسن ایرانی غربیها را آشفته کرد/ تضاد پیشرفتهای انقلاب اسلامی با تئوریهای غربی
https://www.farsnews.ir/news/14000924000638/تولید-واکسن-ایرانی-غربیها-را-آشفته-کرد-تضاد-پیشرفتهای-انقلاب-اسلامی
------------------------------------------------------------------------------------------------

Rank: 3
ID: 7682
عضو کمیسیون فرهنگی مجلس: دلسوزان زوایای پنهان دستاوردهای نظام را بر همگان روشن کنند
https://www.farsnews.ir/news/14001119001033/عضو-کمیسیون-فرهنگی-مجلس-دلسوزان-زوایای-پنهان-دستاوردهای-نظام-را-بر
------------------------------------------------------------------------------------------------

Rank: 4
ID: 11860
گزارش دیوان محاسبات از طرح اقدام ملی مسکن/ پیشرفت ناچیز سبد متنوع طرحهای مسکن
https://www.farsnews.ir/news/14000804000337/گزارش-دیوان-محاسبات-از-طرح-اقدام-ملی-مسکن-پیشرفت-ناچیز-سبد-متنوع
------------------------------------------------------------------------------------------------

Rank: 5
ID: 7795
انقلاب اسلامی؛ جهش علمی، نخبه‌پروری، تمدن اسلامی| هرجا مثل خودروسازی پول بر تعهد غلبه کرد عقب ماندیم
https://www.farsnews.ir/news/14001116000704/انقلاب-اسلامی-جهش-علمی-نخبه-پروری-تمدن-اسلامی|-هرجا-مثل-خودروسازی-پول
```

نتیجه خبر اول

تعداد کلمات فوتبال و ایران و پیشرفت در خبر بسیار زیاد است. موضوع خبر به طور کلی با پیشرفت فوتبال ایران مرتبط است و نتیجه جست و جو از این لحاظ مناسب است. خبر به طور کلی طولانی است و تکرار بالاست چون از لیست قهرمان استفاده شده.

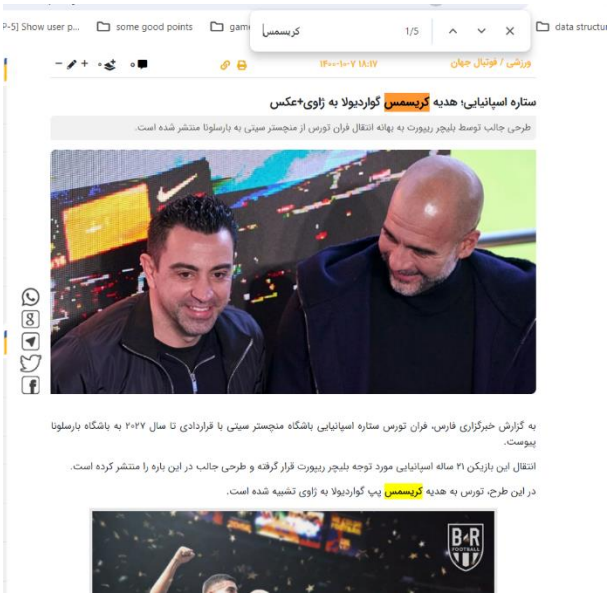


ج)پرسمان دشوار تک کلمه‌ای

کلمه کریسمس

```
[38]: query_search('کریسمس', result_numbers = 5, champion_list = True)

{'کریسمس': 1}
Cosine Scores:
==========================================================================================
------------------------------------------------------------------------------------------

Rank: 1
ID: 5933
ستاره اسپانیایی؛ هدیه کریسمس گواردیولا به زاوی+عکس
https://www.farsnews.ir/news/14001007000739/ستاره-اسپانیایی-هدیه-کریسمس-گواردیولا-به-زاوی-عکس
------------------------------------------------------------------------------------------

Rank: 2
ID: 6117
گیروش «دیکتاتور» لقب گرفت/اختلاف مرد پرتغالی با مصری‌ها به خاطر کریسمس+عکس
https://www.farsnews.ir/news/14001005000165/گیروش-دیکتاتور-لقب-گرفت-اختلاف-مرد-پرتغالی-با-مصری‌ها-به-خاطر-کریسمس
------------------------------------------------------------------------------------------

Rank: 3
ID: 6120
کشتار در ورزشگاه فوتبال در آستانه سال جدید
https://www.farsnews.ir/news/14001005000143/کشتار-در-ورزشگاه-فوتبال-در-آستانه-سال-جدید
------------------------------------------------------------------------------------------

Rank: 4
ID: 5926
مهاجم خارجی مس رفسنجان،4 کودک را به محل تحصیل برگرداند +عکس
https://www.farsnews.ir/news/14001007000809/مهاجم-خارجی-مس-رفسنجان4-کودک-را-به-محل-تحصیل-برگرداند-عکس
------------------------------------------------------------------------------------------

Rank: 5
ID: 5431
وقتی زیدان به رئال مادرید بازگشت+افتخارات
https://www.farsnews.ir/news/14001014000983/وقتی-زیدان-به-رئال-مادرید-بازگشت-افتخارات
```

نتیجه خبر اول



خبر مستقیما در رابطه با کریسمس نیست و صرفا کلمه به عنوان تشبیه به کار رفته. با توجه به دشوار بودن کلمه تعداد تکرار آن درداکیومنت ها کم است برای مثال در متن این داکیومنت ۱ بار تکرار شده فقط.

د) پرسمان دشوار چندکلمه ای

عبارت کمیسیون اجتهاد

```
query_search('کمیسیون اجتهاد', result_numbers = 5, champion_list = True)

{'کمیسیون': 1, 'اجتهاد': 1}
Cosine Scores:
==========================================================================================
------------------------------------------------------------------------------------------

Rank: 1
ID: 12141
«ماموستا عبدالسلام کریمی» مشاور رئیس جمهور در امور اقوام و اقلیت‌های دینی و مذهبی شد
https://www.farsnews.ir/news/14000725000846/ماموستا-عبدالسلام-کریمی-مشاور-رئیس-جمهور-در-امور-اقوام-و-اقلیت‌های
------------------------------------------------------------------------------------------

Rank: 2
ID: 9816
تحقیر زن در اندیشه غرب
https://www.farsnews.ir/news/14000923000557/تحقیر-زن-در-اندیشه-غرب
------------------------------------------------------------------------------------------

Rank: 3
ID: 12198
نقدی بر یادداشت «مرزبندی گفتمانی با طالبان»/ واروته‌نمایی گفتمانی اصلاح‌طلبان
https://www.farsnews.ir/news/14000724000611/نقدی-بر-یادداشت-مرزبندی-گفتمانی-با-طالبان-واروته-نمایی-گفتمانی
------------------------------------------------------------------------------------------

Rank: 4
ID: 11935
تفکر نهادسازی در اندیشه آیت‌الله مهدوی‌کنی و ایده ایشان برای تولید علوم انسانی اسلامی
https://www.farsnews.ir/news/14000801000312/تفکر-نهادسازی-در-اندیشه-آیت‌الله-مهدوی‌کنی-و-ایده-ایشان-برای-تولید
------------------------------------------------------------------------------------------

Rank: 5
ID: 11446
دستورکار کمیسیون‌های مجلس/ بررسی تفحص از شستا و اتاق بازرگانی و دیدار با ازه‌ای و مخبر
https://www.farsnews.ir/news/14000815000142/دستورکار-کمیسیون‌های-مجلس-بررسی-تفحص-از-شستا-و-اتاق-بازرگانی-و-دیدار
```

نتیجه خبر اول

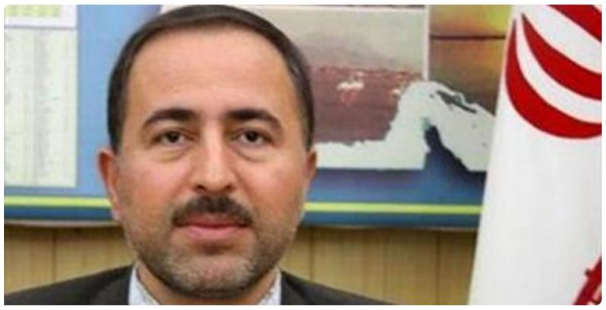
```
dictionary['کمیسیون']

{'frequency': 6364,
 'docs': {'13': {'positions': [81, 94],
```

```
dictionary['اجتهاد']

{'frequency': 16,
```

کلمه اجتهاد تعداد تکرار به شدت کمی دارد در نتیجه idf آن بسیار بالاتر است و به همین دلیل در محاسبه شباهت داکیومنتها وزن و تاثیر بیشتری میگیرد. به همین دلیل در چند خبر اول، داکیومنت ها به بود و نبود کلمه کمیسیون اهمیتی نداده اند و صرفا بودن کلمه اجتهاد برای بالا رفتن امتیاز کافی بوده. خبر ارتباط خاصی با اجتهاد ندارد و صرفا کلمه اجتهاد در آن آمده.

«ماموستا عبدالسلام کریمی» مشاور رئیس جمهور در امور اقوام و اقلیت‌های دینی و مذهبی شد

رئیس جمهور با صدور حکمی ماموستا عبدالسلام کریمی را به عنوان «مشاور رئیس جمهور در امور اقوام و اقلیت‌های دینی و مذهبی» منصوب کرد.



به گزارش خبرگزاری فارس، آیت الله سید ابراهیم رئیسی رئیس جمهور با صدور حکمی ماموستا عبدالسلام کریمی را به عنوان «مشاور رئیس جمهور در امور اقوام و اقلیت‌های دینی و مذهبی» منصوب کرد.

ماموستا عبدالسلام کریمی، متولد سال ۱۳۴۶، فرزند شهید، دانش‌آموخته دکترای زبان و ادبیات عرب و دارای گواهی افتا و اجتهاد از طرف مرکز بزرگ اسلامی غرب کشور در سال ۱۳۷۰ است.

فرماندار شهرستان دیواندره به مدت ۵ سال، رئیس دانشگاه‌های پیام نور مریوان و سنندج به مدت ۵ سال، معاون برنامه‌ریزی