فاز اول

مجموعه داده

ابتدا فایل مجموعه داده را لود میکنیم. نمونه از داده خوانده شده را می توانید در تصویر زیر مشاهده کنید.

```
import ...

f = open('IR_data_news_12k.json')
# f = open('IR_data_news_5k 2.json')
json_data = json.load(f)
contents, titles, urls = zip(*[(item.get("content", ""), item.get("title", ""),
                                item.get("url", "")) for item in json_data.values()])

# Example:
print('1st doc:')
print("Title:", titles[0])
print("URL:", urls[0])
print("Content:", contents[0])

```

```
1st doc:
Title: اعلام زمان قرعه کشی جام باشگاه های فوتسال آسیا
URL: https://www.farsnews.ir/news/14001224001005/اعلام-زمان-قرعه-کشی-جام-باشگاه-های-فوتسال-آسیا
Content:
به گزارش خبرگزاری فارس، کنفدراسیون فوتبال آسیا (AFC) در نامه ای رسمی به فدراسیون فوتبال ایران و باشگاه گیتی پسند زمان قرعه کشی جام باشگاه های فوتسال آسیا را رسماً اعلام کرد. بر این اساس 25 فروردین ماه 1401 مراسم قرعه کشی جام باشگاه های فوتسال آسیا در مالزی برگزار می شود. باشگاه گیتی پسند بعنوان قهرمان فوتسال ایران در سال 1400 به این مسابقات راه پیدا کرده است. پیش از این گیتی پسند تجربه 3 دوره حضور در جام باشگاه های فوتسال آسیا را داشته که هر سه دوره به فینال مسابقات راه پیدا کرده و یک عنوان قهرمانی و دو مقام دومی بدست آورده است. انتهای پیام/
```

پیشپردازش اسناد

نرمال سازی

در ابتدا چندین لیست را مقداردهی میکنیم تا در ادامه برای بخش normalization از آنها بهره ببریم. نام هر یک از لیست ها بیانگر کاربردی است که برای آن مد نظر است و در ادامه به هر یک از نرمال سازی ها می پردازیم.

```
# lists and parameters initializing
with open('verbs.txt', 'r', encoding='utf-8') as verbs_file:
    for v in verbs_file:
        verbs.append(v[:-1])

punctuations_replacements = [...]

diacritics_replacements = [...]

specials_chars_replacements = [...]

unicodes_replacements = [...]

punc_after, punc_before = r"\.:!،؛؟»\]\)\}", r"«\[\(\{"

spacing_patterns = [...]

extra_space_replacements = [...]

punctuation_spacing_replacements = [...]

number_replacements = {...}
```

تابع نرمال سازی چنین است:

```python
def normalize(text):
    text = replace(punctuations_replacements, text)
    text = replace(specials_chars_replacements, text)
    text = replace(diacritics_replacements, text)
    text = replace(unicodes_replacements, text)
    text = separate_mi(text)
    text = correct_spacing(text)
    text = replace(number_replacements, text)

    return text
```

به ترتیب موارد زیر در نرمال سازی انجام میشوند:

- حذف علائم نگارشی
- حذف کرکترهای خاص مانند کرکترهای زبان عربی
- حذف مانند فتحه و کسره و ...
- تبدیل Unicode ها و آ و ی و ک به فرمت واحد
- اعمال نیم فاصله میان افعال دارای می یا نمی
- اصلاح فاصله ها مانند حالتی که اسپیس های اضافه وجود دارد یا ...
- تبدیل شماره های انگلیسی به فارسی

توابع فراخوانی شده در تابع نرمالایز:

```python
def replace(patterns, text):
    for pattern, repl in patterns:
        text = re.sub(pattern, repl, text)
    return text


def separate_mi(text):
    words = text.split()
    for i in range(len(words)):
        if re.match(r"\b[آابپتثجچحخدذرزژسشصضطظعغفقکگلمنوهی]ن?می+", words[i]):
            prefix = re.sub("(ن?می)^", r"\1ZWNJ", words[i])
            if prefix in verbs:
                words[i] = prefix
    return ' '.join(words)


def correct_spacing(text):
    text = replace(extra_space_replacements, text)
    text = replace(punctuation_spacing_replacements, text)
    return replace(spacing_patterns, text)
```

تابع replace پترن های یک لیست را با جایگزین آنها جایگزین میکند و بارها از آن در نرمال سازی استفاده میکنیم.

Separate_mi به دنبال کلماتی در متن میگردد که با می یا نمی بصورت پیوسته شروع شوند و در لیست افعال باشند. در این صورت نیم فاصله اعمال میشود.

Correct_spacing نیز انواع فواصل را اصلاح میکند و از ۳ بار فراخوانی replace استفاده میکند. replace اول برای حالاتی مثل اسپیس اضافی و .. است. مورد دوم برای اصلاح فواصل مثلا با پرانتز و .. است. مورد آخر نیز برای اصلاح فاصله برای تر، ترین، و ... است.

نمونه ورودی و خروجی نرمال سازی:

```
texts = []
texts.append('0123456789%٠١٢٣٤٥٦٧٨٩')
texts.append('پیامبر اکرم ﷺ')
texts.append('حذف إعراب')
texts.append('به طول ۹متر و عرض۶')
texts.append('سلام دنیـــــا می ایم میروم')
texts.append('جمعهها که کار نمیکنم مطالعه میکنم')
texts.append('')
texts.append('')
for text in texts:
    print(normalize(text))
```

```
٠١٢٣۴۵۶٧٨٩٠١٢٣۴٥٦٧٨٩
پیامبر اکرم
حذف اعراب
به طول ۹ متر و عرض ۶
سلام دنیا میایم میروم
جمعه‌ها که کار نمی‌کنم مطالعه می‌کنم
```

استخراج توکن

از تابع نرمال سازی پیش از تولید توکن ها استفاده میکنیم تا متن بهینه شود. تابع زیر برای تولید توکن ها استفاده میشود. این تابع محتواها را پیمایش میکند، نرمال میکند، توکن ها را استخراج میکند، و در نهایت stemming را بر روی خروجی ها اعمال میکند.

```
def generate_tokens(contents):
    overall_tokens = []
    stemmer = Stemmer()
    for c in contents:
        c = normalize(c)
        tokens = tokenize(c)
        for i in range(len(tokens)):
            tokens[i] = stemmer.stem(tokens[i])
        overall_tokens.append(tokens)
    return overall_tokens


docs = generate_tokens(contents)
```

تابع tokenize ابتدا در صورت نیاز اسپیس میان علائم نگارشی اضافه میکند. البته این کار در صورت عدم نرمال سازی کاربرد دارد. در نرمال سازی انجام شده این علائم را حذف کردیم. اما حالت کلی فارغ از نرمال سازی در نظر گرفته شده است. در ادامه توکن ها با توجه به فاصله ها استخراج می شوند. در نهایت نیز افعال مرکب ۲ کلمه ای به یک توکن مشترک تبدیل می شوند.

```
def tokenize(text):
    pattern = re.compile(r'([؟!?]+|[\d.:]+|[:.:»\])}"«\[({/\\])')
    text = pattern.sub(r" \1 ", text.replace("\n", " ").replace("\t", " "))
    tokens = [word for word in text.split(" ") if word]
    tokens = join_verb_parts(tokens)

    return tokens
```

```
after_verbs = {...}

before_verbs = {...}

verbe = set(
    [verb + "ه" for verb in verbs]
    + ["ن" + verb + "ه" for verb in verbs],
)

def join_verb_parts(tokens):
    if len(tokens) == 1:
        return tokens

    result = [""]
    for token in reversed(tokens):
        if token in before_verbs or (
                result[-1] in after_verbs and token in verbe
        ):
            result[-1] = token + "_" + result[-1]
        else:
            result.append(token)
    return list(reversed(result[1:]))
```

join_verb_parts که در تابع tokenize فراخوانی میشود برای استخراج افعال مرکب است. دو لیست از کلمات اول و دوم در افعال ۲ تایی داریم و با پیمایش توکن ها به ترتیب، آنها را استخراج کرده و تبدیل به توکن میکنیم.

در بخش بعدی می خواهیم ۵۰ توکن پرتکرار را حذف کنیم تا حجم index و هزینه تولید خروجی query ها کمتر شود.

```
def remove_most_frequent_tokens(docs):
    flattened_list = [item for sublist in docs for item in sublist]
    item_frequencies = Counter(flattened_list)
    sorted_frequent = sorted(item_frequencies.items(), key=lambda x: x[1], reverse=True)[:50]
    print('> frequent_terms:')
    for frq in sorted_frequent:
        print(frq)
    frequent_terms = [x for x, _ in sorted_frequent]
    print(frequent_terms)
    new_docs=[]
    for doc in docs:
        dd=[]
        for t in doc:
            if not t in frequent_terms:
                dd.append(t)
        new_docs.append(dd)
    return new_docs


docs = remove_most_frequent_tokens(docs)
```

در تابع بالا ابتدا لیست توکن همه متن ها را به یک لیست واحد تبدیل میکنیم. سپس با استفاده از counter تعداد هر element لیست شمرده می شود و مرتب سازی انجام میشود. در ادامه لیست جدید برای doc های جدید در نظر میگیریم. همه لیست ها را بررسی میکنیم و توکن های آنها را در صورتی که عضو ۵۰ تا پرتکرار نبودند در نظر میگیریم. به این صورت این ۵۰ توکن از لیست ها حذف می شوند.

لیست ۵۰ پرتکرار:

```
> frequent_terms:
('و', 234741)
('در', 165012)
('به', 136382)
('از', 92924)
('این', 84204)
('که', 76243)
('را', 69659)
('با', 69230)
('اس', 46515)
('برا', 30975)
('ه', 28174)
('کرد', 26725)
('یک', 23902)
('کشور', 22369)
('ت', 22289)
('ما', 19729)
('خود', 18859)
('بر', 18829)
('', 18579)
('شد', 17098)
('باز', 16821)
('باید', 16197)
('تا', 15919)
('اما', 15197)
('گف', 14751)
('فارس', 14173)
('مرد', 13944)
('بود', 13234)
('گزار', 13166)
('دو', 12874)
('پیا', 12811)
('کار', 12807)
('ایر', 12728)
('سال', 12638)
('مل', 12464)
('خبرگزار', 12416)
('انت', 12352)
('رئیس', 11628)
('دول', 11589)
('ب', 11212)
('دارد', 10782)
('بازیکن', 10625)
('دا', 10440)
('شود', 10260)
('اینکه', 10052)
('قرار', 9492)
('ان', 9304)
('انقلاب', 9168)
```

ساخت شاخص مکانی

در ادامه باید index را تولید کنیم.

```
def generate_positional_index(docs):
    postings = {}
    for doc_id, doc in enumerate(docs):
        for pos, token in enumerate(doc):
            if token not in postings:
                postings[token] = {'doc_freq': 0, 'posting': {}, 'tf_idf': {}}
            if doc_id in postings[token]['posting']:
                postings[token]['posting'][doc_id].append(pos)
            else:
                postings[token]['posting'][doc_id] = [pos]
                postings[token]['doc_freq'] += 1
    return postings


postings = generate_positional_index(docs)
```

تمام لیست توکن های اسناد را پیمایش میکنیم و در صورت نبودن هر توکن در دیکشنری، آن را اضافه میکنیم. در غیر این صورت لیست آن را به روز میکنیم. Doc_freq در واقع تعداد اسناد دارای توکن است. در صورتی که یک سند پیشتر بررسی شده بود و توکن در آن دیده شده بود، وارد else می شویم و doc_freq تغییر نمیکند. بصورت کلی هر توکن علاوه بر doc_freq، لیست posting ها بصورت لیست doc_id : [positions] دارد و مقدار دیکشنری tf_idf ان نیز در ادامه محاسبه میشود.

چند خط اول postings کلمه تیم:

```
postings['تیم']
```

```
{'doc_freq': 2736,
 'posting': {3: [57],
  4: [8],
  8: [286],
  9: [13, 18, 29],
  11: [10],
  14: [57],
  17: [53],
  24: [99],
  28: [113],
  29: [589],
```

فاز دوم

Tf_idf

تابع رو به رو برای محاسبه tf_idf است. تمام توکن های درون ایندکس بررسی میشوند. Idf هر یک با توجه به تعداد کل اسناد و doc_freq که از پیش برای آن ذخیره شده بود قابل محاسبه است.

برای محاسبه tf نیز تعداد position های مربوط به یک سند شمرده میشود. در نهایت ضرب tf و idf محاسبه می شود و در ایندکس نگهداری میشود. توان دوم همه tf_idf های اسناد نیز در یک دیکشنری ذخیره می شوند تا در نهایت طول بردار هر یک محاسبه شود و برای نرمال سازی برداری بتوان از آنها استفاده کرد.

```python
def calculate_tf_idf(postings, docs):
    document_length = defaultdict(float)

    for token, posting in postings.items():
        idf = math.log10(len(docs) / posting['doc_freq'])

        for doc_id, positions in posting['posting'].items():
            tf = math.log10(len(positions)) + 1
            tf_idf = tf * idf
            posting['tf_idf'][doc_id] = tf_idf
            document_length[doc_id] += tf_idf ** 2

    for doc_id, length in document_length.items():
        document_length[doc_id] = math.sqrt(length)

    return dict(document_length)
```

Query

توابع رو به رو نیز مربوط به query هستند. تابع اول بطور کلی توکن های کوئری را استخراج میکند.

تابع دوم نیز tf را برای توکن های استحراج شده query محاسبه می کند تا در محاسبه شباهت مورد استفاده قرار گیرد.

```python
def tokenize_query(query):
    placeholders = []
    def repl(m):
        placeholders.append(m.group())
        return f'___{len(placeholders) - 1}___'

    query = re.sub(r'"[^"]*"', repl, query)

    tokens = query.split()

    for i, ph in enumerate(placeholders):
        tokens = [t.replace(f'___{i}___', ph) for t in tokens]

    return tokens


def calculate_query_tf(query):
    query_tf = {}

    for token in tokenize_query(query):
        if token in query_tf.keys():
            query_tf[token] += 1
        else:
            query_tf[token] = 1

    for token in query_tf.keys():
        query_tf[token] = 1 + math.log10(query_tf[token])

    tf_sum = sum(math.pow(tf, 2) for tf in query_tf.values())
```

Champions

تابع زیر برای محاسبه لیست champions است تا عملکرد IR بهتر شود. در این تابع برای هر کلمه، tf_idf آن به ازای همه اسناد محاسبه میشود. سپس این نتایج نزولی مرتب می شوند و در صورت وجود، نهایتا ۶۰ تای برتر در دیکشنری champions_list برای آن کلمه ذخیره می شوند. در این صورت می توان برای محاسبه شباهت و تولید خروجی برای کوئری ها ابتدا به سراغ این لیست آمد که سرعت را بهبود می بخشد.

```
def generate_champions(postings_lists):
    champion_lists = {}
    for term in postings_lists:
        tf_idf_scores = {}
        for doc_id in postings_lists[term]['posting'].keys():
            if doc_id in tf_idf_scores.keys():
                tf_idf_scores[doc_id] += postings_lists[term]['tf_idf'][doc_id] / doc_len[doc_id]
            else:
                tf_idf_scores[doc_id] = postings_lists[term]['tf_idf'][doc_id] / doc_len[doc_id]

        sorted_docs = sorted(tf_idf_scores.items(), key=lambda x: x[1], reverse=True)
        r = min(60, len(sorted_docs))
        champion_lists[term] = [doc_id for doc_id, _ in
                                sorted_docs[:r]]

    return champion_lists
```

Cosine

برای محاسبه شباهت کسینوسی، ابتدا با استفاده از توابعی که پیش تر دیدیم tf و طول بردار را برای کوئری محاسبه میکنیم. در ادامه برای هر توکن درون کوئری:

- در صورت استفاده از champions:
  - در صورتی که توکن درون لیست champions باشد(درون دیکشنری بوده باشد و محاسبه لیست آن انجام شده باشد)، بر ر وی لیست قهرمانان آن پیمایش میکنیم. برای هر سند درون آن لیست، ضرب tf_idf آن را در tf کوئری محاسبه میکنیم.
  - در صورت عدم استفاده از لیست قهرمانان، برای هر توکن کوئری، در صورتی که در ایندکس وجود داشته باشد، ضرب مذکور را برای همه doc ها در لیست آن محاسبه میکنیم.

در ادامه حاصل ضربهای بدست آمده را بر طول بردار سند مربوطه و نیز کوئری تقسیم میکنیم تا به نوعی نرمال سازی انجام شده باشد و فرموا شباهت cosine کامل شده باشد. خروجی های بدست آمده بصورت مرتب شده نزولی return می شوند.

```python
def calculate_cosine_similarity(query, use_champions=True):
    query_tf, query_sum = calculate_query_tf(query)

    doc_cosine_score_dic = {}

    for token in query_tf.keys():
        if use_champions:
            if token in champion_lists.keys():
                for doc_id in champion_lists[token]:
                    if not doc_id in doc_cosine_score_dic.keys():
                        doc_cosine_score_dic[doc_id] = 0
                    doc_cosine_score_dic[doc_id] += query_tf[token] * postings[token]['tf_idf'][doc_id]

        else:
            if token in postings.keys():
                for doc_id in postings[token]['posting'].keys():
                    if doc_id not in doc_cosine_score_dic.keys():
                        doc_cosine_score_dic[doc_id] = 0
                    doc_cosine_score_dic[doc_id] += query_tf[token] * postings[token]['tf_idf'][doc_id]

    # calculate similarity
    doc_cosine_similarity = {}

    for doc_id in doc_cosine_score_dic.keys():
        doc_cosine_similarity[doc_id] = doc_cosine_score_dic[doc_id] / doc_len[doc_id]
        doc_cosine_similarity[doc_id] = doc_cosine_similarity[doc_id] / query_sum
    sorted_doc_cosine_similarity = sorted(doc_cosine_similarity.items(), key=lambda x: x[1], reverse=True)

    return sorted_doc_cosine_similarity
```

Jaccard

شباهت jaccard نیز در زیر آمده است. اساس کار آن حلقه ای مشابه بخش قبل است. تفاوت در محاسبه مقدار امتیاز است که برابر است با اشتراک سند و کوئری تقسیم به اجتماع آنها. نتایج مشابه بخش قبل بصورت مرتب شده برگردانده میشوند.

```
def calculate_jaccard_similarity(query, use_champions=True):
    query_tokens = set(tokenize_query(query))
    doc_jaccard_scores = {}

    for term in query_tokens:
        if use_champions:
            if term in champion_lists:
                for doc_id in champion_lists[term]:
                    if doc_id not in doc_jaccard_scores.keys():
                        doc_tokens = set([token for token in docs[doc_id]])
                        jaccard_score = len(query_tokens.intersection(doc_tokens)) / len(query_tokens
                         .union(doc_tokens))
                        doc_jaccard_scores[doc_id] = jaccard_score
        else:
            if term in postings:
                for doc_id in postings[term]['posting'].keys():
                    if doc_id not in doc_jaccard_scores.keys():
                        doc_tokens = set([token for token in docs[doc_id]])
                        intersection = len(query_tokens.intersection(doc_tokens))
                        union = len(query_tokens.union(doc_tokens))
                        jaccard_score = intersection / union
                        doc_jaccard_scores[doc_id] = jaccard_score

    sorted_doc_jaccard_scores = sorted(doc_jaccard_scores.items(), key=lambda x: x[1], reverse=True)
    return sorted_doc_jaccard_scores
```

Results

در نهایت این تابع نیز برای چاپ خروجی استفاده می شود.

```
def print_res(scores):
    counter = 0
    for doc_id, score in scores:
        counter += 1
        if counter > 10:
            break

        content = contents[doc_id]
        print()
        print('DocID: ', doc_id, 'Title:', titles[doc_id], 'URL:', urls[doc_id])
        print('Score: ', score)
        print('Content:\n ', content)
```

Test(using champions list)

الف) ساده : ایران.

تعدادی از برترین خروجی ها:

```
DocID:  5492 Title:  تمجید اکانت رسمی لیگ قهرمانان اروپا از آزمون؛ سردار ایرانی اروپا را متحیر می
کند+فیلم URL: https://www.farsnews
.ir/news/14001014000112/تمجید-اکانت-رسمی-لیگ-قهرمانان-اروپا-از-آزمون-سردار-ایرانی-اروپا-را
Score:  0.1620591935312246

DocID:  4751 Title: زاهدی برای اولین بار به تیم ملی دعوت شد+عکس URL: https://www.farsnews
.ir/news/14001023000614/زاهدی-برای-اولین-بار-به-تیم-ملی-دعوت-شد-عکس
Score:  0.15520395543115656

DocID:  3292 Title: استقبال صفحه رسمی بوندس لیگا از سردار آزمون+عکس URL: https://www.farsnews
.ir/news/14001110000986/استقبال-صفحه-رسمی-بوندس-لیگا-از-سردار-آزمون-عکس
Score:  0.15472514600789533

DocID:  3506 Title: معاون رئیس جمهور:دولت از تیم‌ملی فوتبال حمایت می کند/قلب میلیون‌ها ایرانی شاد شد
URL: https://www.farsnews
.ir/news/14001107000682/معاون-رئیس-جمهور-دولت-از-تیم‌ملی-فوتبال-حمایت-می-کند-قلب-میلیون‌ها
Score:  0.15395506950376028

DocID:  1714 Title: طارمی به ترکیب اصلی پورتو بازگشت+عکس URL: https://www.farsnews
.ir/news/14001201001183/طارمی-به-ترکیب-اصلی-پورتو-بازگشت-عکس
Score:  0.14522470467149248
```

محتوای سند اول:

```
DocID:  5492 Title:  تمجید اکانت رسمی لیگ قهرمانان اروپا از آزمون؛ سردار ایرانی اروپا را متحیر می
کند+فیلم URL: https://www.farsnews
.ir/news/14001014000112/تمجید-اکانت-رسمی-لیگ-قهرمانان-اروپا-از-آزمون-سردار-ایرانی-اروپا-را
Score:  0.1620591935312246
Content:

به گزارش خبرگزاری فارس، اکانت رسمی لیگ قهرمانان اروپا با انتشار ویدئویی از گل های سردار آزمون در
این رقابت ها از مهاجم ایرانی تمجید کرد و نوشت: او می تواند برای سال های آینده ستاره واقعی فوتبال
ایران و آسیا باشد. ستاره ایرانی اروپا را متحیر می کند. *این فیلم را ببینید انتهای پیام/
```

کلماتی مانند ایران و ایرانی در محتوای آن زیاد آمده اند و امتیاز آن را بالا برده اند.

با مشاهده محتوای خروجی اول مشاهده میکنیم که کلمات ایران و ایرانی و ... چندین بار تکرار شده اند. همچنین با توجه به پرتکرار بودن کلمه، امتیاز خروجی ها به نسبت نزدیک است

ب) ساده و مرکب : باشگاه والیبال

تعدادی از برترین خروجی ها:

```
DocID:  1116 Title: میزبان مسابقات والیبال جام باشگاه های آسیا مشخص شد URL: https://www.farsnews
.ir/news/14001209000462/میزبان-مسابقات-والیبال-جام-باشگاه های-آسیا-مشخص-شد
Score:  0.2613948532903056

DocID:  4895 Title: داورزنی: حضورم در پرسپولیس شایعه است URL: https://www.farsnews
.ir/news/14001021000947/داورزنی-حضورم-در-پرسپولیس-شایعه-است
Score:  0.24467415784729635

DocID:  4805 Title: دیدار سرمربیان والیبال و فوتبال پیکان در فرودگاه کرمان URL: https://www
.farsnews.ir/news/14001022000908/دیدار-سرمربیان-والیبال-و-فوتبال-پیکان-در-فرودگاه-کرمان
Score:  0.20960841323962676

DocID:  2805 Title: رقابت‌های والیبال جام باشگاه های آسیا با حضور چند تیم برگزار می‌شود؟ URL:
https://www.farsnews
.ir/news/14001117000319/رقابت‌های-والیبال-جام-باشگاه های-آسیا-با-حضور-چند-تیم-برگزار-می‌شود
Score:  0.1786146294514291

DocID:  495 Title: تکلیف میزبان مسابقات والیبال جام باشگاه های آسیا مشخص شد URL: https://www
.farsnews.ir/news/14001217000551/تکلیف-میزبان-مسابقات-والیبال-جام-باشگاه های-آسیا-مشخص-شد
Score:  0.1758932867840487
```

محتوای سند اول:

```
DocID:  1116 Title: میزبان مسابقات والیبال جام باشگاه های آسیا مشخص شد URL: https://www.farsnews
.ir/news/14001209000462/میزبان-مسابقات-والیبال-جام-باشگاه های-آسیا-مشخص-شد
Score:  0.2613948532903056
Content:

 به گزارش خبرنگار ورزشی خبرگزاری فارس، پس از اعلام انصراف ارومیه از میزبانی مسابقات والیبال جام
 باشگاه های آسیا، شهرهای تهران و اصفهان به عنوان جدی‌ترین گزینه‌های میزبانی این دوره از مسابقات معرفی
 شدند. با بررسی درخواست باشگاه های پیکان و سپاهان سرانجام قرعه به نام خودروسازان افتاد و به این
 ترتیب رقابت‌های والیبال جام باشگاه های آسیا به میزبانی تهران و باشگاه پیکان برگزار خواهد شد. پیکان
 پرافتخارترین تیم ایران و آسیا به شمار می‌رود و این میزبانی می‌تواند به شاگردان پیمان اکبری کمک کند
 تا افتخار دیگری در کارنامه تیم پیکان ثبت کنند.  انتهای پیام/
```

این سند هر دو کلمه کوئری را دارد. اما هر دو به نسبت ساده و رایج هستند و در ترکیب با کلمات مختلفی علاوه بر یکدیگر به کار میروند. برای همین در سند بالا می توان دید که این دو کلمه در کنار یکدیگر نیامده اند. برای همین احتمال برگردانده شدن اسناد غیر مرتبط با کل کوئری وجود دارد.

پ) دشوار تک کلمه ای: کمیسیون

تعدادی از برترین خروجی ها:

```
DocID:  8274 Title: دستور جلسات هفته آینده کمیسیون‌های مجلس URL: https://www.farsnews
.ir/news/14001108000634/دستور-جلسات-هفته-آینده-کمیسیون‌های-مجلس
Score:  0.20882753754496872

DocID:  10084 Title: اعضای کمیسیون فرهنگی به طبس و رفسنجان سفر می‌کنند URL: https://www.farsnews
.ir/news/14000916000703/اعضای-کمیسیون-فرهنگی-به-طبس-و-رفسنجان-سفر-می‌کنند
Score:  0.1870492270100534

DocID:  9807 Title: نوروزی، دلخوش و موحد عضو کمیسیون تلفیق بودجه ۱۴۰۱ شدند URL: https://www
.farsnews.ir/news/14000923000737/نوروزی-دلخوش-و-موحد-عضو-کمیسیون-تلفیق-بودجه-۱۴۰۱-شدند
Score:  0.18388023086656977

DocID:  9084 Title: کمیسیون تلفیق کلیات لایحه بودجه را تصویب کرد URL: https://www.farsnews
.ir/news/14001014001128/کمیسیون-تلفیق-کلیات-لایحه-بودجه-را-تصویب-کرد
Score:  0.1802447482377456

DocID:  11247 Title: تأکید اعضای کمیسیون فرهنگی مجلس بر اصلاح آیین نامه سازمان امور سینمایی URL:
https://www.farsnews
.ir/news/14000818000811/تأکید-اعضای-کمیسیون-فرهنگی-مجلس-بر-اصلاح-آیین-نامه-سازمان-امور-سینمایی
Score:  0.17680376105610696
```

محتوای سند اول:

```
DocID:  8274 Title: دستور جلسات هفته آینده کمیسیون‌های مجلس URL: https://www.farsnews
.ir/news/14001108000634/دستور-جلسات-هفته-آینده-کمیسیون‌های-مجلس
Score:  0.20882753754496872
Content:

 به گزارش گروه سیاسی خبرگزاری فارس، دستور جلسات هفتگی کمیسیون‌های مجلس شورای اسلامی ( از شنبه ۹
بهمن تا پنج شنبه ۱۴ بهمن ۱۴۰۰) از شنبه اعلام شد. دستور جلسات هفتگی کمیسیون‌های مجلس را در
ادامه مشاهده کنید: دستور جلسات کمیسیون‌های مجلس انتهای پیام/
```

با توجه به خاص بودن کلمه تا حدی، امتیاز اسناد برگردانده شده نسبت به بخش الف تغییرات بیشتری دارند، چرا که doc freq در این حالت کم است. همچنین سند های برگردانده شده میتوانند مرتبط بودن بهتری نسبت به الف داشته باشند، چرا که کلمه رایج نیست و محدوده جست و جو کم است و ارتباط اسناد برتر با آن بیشتر می تواند باشد.

ت) دشوار چند کلمه ای: کمیسیون مجلس

تعدادی از برترین خروجی ها:

```
DocID:  8274 Title: دستور جلسات هفته آینده کمیسیون‌های مجلس URL: https://www.farsnews
.ir/news/14001108000634/دستور-جلسات-هفته-آینده-کمیسیون‌های-مجلس
Score:  0.14766336789653572

DocID:  10084 Title: اعضای کمیسیون فرهنگی به طبس و رفسنجان سفر می‌کنند URL: https://www.farsnews
.ir/news/14000916000703/اعضای-کمیسیون-فرهنگی-به-طبس-و-رفسنجان-سفر-می‌کنند
Score:  0.13226377683451065

DocID:  9807 Title: نوروزی، دلخوش و موحد عضو کمیسیون تلفیق بودجه ۱۴۰۱ شدند URL: https://www
.farsnews.ir/news/14000923000737/نوروزی-دلخوش-و-موحد-عضو-کمیسیون-تلفیق-بودجه-۱۴۰۱-شدند
Score:  0.13002295817189938

DocID:  9084 Title: کمیسیون تلفیق کلیات لایحه بودجه را تصویب کرد URL: https://www.farsnews
.ir/news/14001014001128/کمیسیون-تلفیق-کلیات-لایحه-بودجه-را-تصویب-کرد
Score:  0.12745228375217194

DocID:  11247 Title: تأکید اعضای کمیسیون فرهنگی مجلس بر اصلاح آیین نامه سازمان امور سینمایی URL:
https://www.farsnews
.ir/news/14000818000811/تأکید-اعضای-کمیسیون-فرهنگی-مجلس-بر-اصلاح-آیین-نامه-سازمان-امور-سینمایی
Score:  0.12501913838205925
```

محتوای سند اول:

```
DocID:  8274 Title: دستور جلسات هفته آینده کمیسیون‌های مجلس URL: https://www.farsnews
.ir/news/14001108000634/دستور-جلسات-هفته-آینده-کمیسیون‌های-مجلس
Score:  0.14766336789653572
Content:

 به گزارش گروه سیاسی خبرگزاری فارس، دستور جلسات هفتگی کمیسیون‌های مجلس شورای اسلامی ( از شنبه ۹
بهمن تا پنج شنبه ۱۴ بهمن ۱۴۰۰) از شنبه اعلام شد. دستور جلسات هفتگی کمیسیون‌های مجلس را در
ادامه مشاهده کنید: دستور جلسات کمیسیون‌های مجلس انتهای پیام/
```

مشابه بخش قبل کاهش امتیاز ها معمولا در این حالت سریع تر و واضح است و حتی نسبت به بخش قبل بیشتر است، چرا که کلمات خاص بیشتری داریم. همانطور که مشاهده می توان کرد، هر دو کلمه پشت سر هم در خروجی اول آمده اند. چرا که هر دو خاص هستند و در ترکیب های کمتری با کلمات دیگر به کار می روند و احتمال مشاهده آنها در کنار هم زیاد است. بنابراین اسناد برتر برگردانده شده ارتباط بالایی دارند و این ارتباط در رنک های پایین تر به مرور کاهش می یابد.