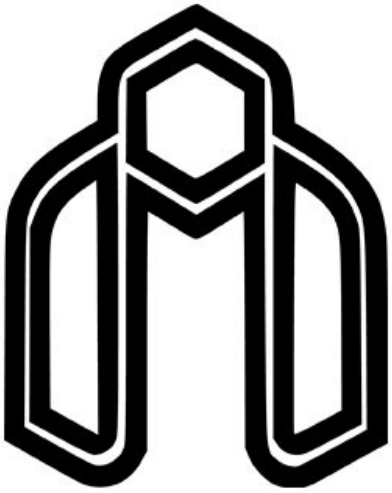
بنام آنکه هستی نقش از اوست



دانشگاه صنعتی شاهرود

Shahrood University of Technology

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

گزارش حل تمرین (داده کاوی)

استاد : دکتر مرتضی زاهدی

دانشجو : احسان پایدار
شماره دانشجویی : ۴۰۱۰۳۸۳۴

نیم سال اول ۱۴۰۲-۱۴۰۱

تمرین :

- از سایت های خبری ۱۰۰۰ خبر اقتصادی و ۱۰۰۰ خبری ورزشی (متن کامل) از هر کدام ۲۰۰ تا برای تست استفاده کنیم.
- اندازه بگیریم بین ۱۶۰۰ تا متن هر کدام از کلمات چند بار تکرار شده اند و شماره گذاری بشوند.
- کلمات StopWord شناسایی و حذف کنیم.
- برای هر متن (ورزشی / اقتصادی) فیچر وکتور درست کنیم.
- میانگین واریانس را بدست می آوریم.

کد اصلی تمرین

پاسخ) متن کاوی یکی از شاخه‌های مهم پردازش زبان طبیعی محسوب می‌شود. ابزارهای مختلف پردازش زبان طبیعی به ترتیب (متداول) استفاده برای پیش‌پردازش متن بهمراه نمونه کد برای آنها معرفی می‌شوند. منظور از ابزارهای پردازش متن، کتابخانه‌هایی است که برای آماده‌سازی متن جهت متن کاوی و استخراج دانش از متن بکار می‌روند.

لازم به ذکر است که برای اجرای نمونه کدهای مربوط به زبان فارسی ابتدای فایل شبه کد ذیل را اضافه نمایید و قبل از اجرا لازم است تا عبارت YOUR_API_KEY (خط ۱۴ شبه کد زیر) را با کلید ای.پی.آی واقعی خودتان جایگزین نمایید. شما می‌توانید در کمتر از دو دقیقه از این سایت:

https://scikit-learn.org/stable/modules/classes.html

کلید API رایگان برای استفاده از کلیه امکانات متن کاوی فارسی تهیه کنید.

```
1. import requests
2. import json
3.
4. def callApi(url, data, tokenKey):
5. headers = {
6. 'Content-Type': "application/json",
7. 'Authorization': "Bearer " + tokenKey,
8. 'Cache-Control': "no-cache"
9. }
10. response = requests.request("POST", url, data=data.encode("utf-8"),
    headers=headers)
11. return response.text
12. ##################### Get Token by Api Key ##########################
13. url = "http://api.text-mining.ir/api/Token/GetToken"
14. querystring = {"apikey":"YOUR_API_KEY"}
15. response = requests.request("GET", url, params=querystring)
16. data = json.loads(response.text)
17. tokenKey = data['token']
```

نرمالساز متن(Normalizer)

هدف این ابزار، تمیز و مرتب کردن متن و یکسان‌سازی کاراکترها با جایگزین کردن کاراکترهای استاندارد در متن ورودی است. در واقع قبل از پردازش متون جهت استانداردسازی حروف و فاصله‌ها بایستی پیش‌پردازش‌هایی روی آنها انجام شود. در واقع در این مرحله بایستی همه‌ی نویسه‌های (حروف) متن با جایگزینی با معادل استاندارد آنها، یکسان‌سازی گردند. در پردازش رسم الخط زبان فارسی، با توجه به شباهتی که با رسم الخط عربی دارد، همواره در نگارش تعدادی از حرف‌ها مشکل استفاده از کاراکترهای عربی معادل وجود دارد؛ که از جمله‌ی آنها می‌توان به حروف "ک"، "ی" و همزه اشاره نمود. در اولین گام باید مشکلات مربوط به این حروف را با یکسان‌سازی آنها برطرف کرد.

علاوه بر این، اصلاح و یکسان‌سازی نویسه‌ی نیم‌فاصله و فاصله در کاربردهای مختلف آن و همچنین حذف نویسه‌های اعراب، تشدید، تنوین و «ـ» که برای کشش نویسه‌های چسبان مورد استفاده قرار می‌گیرد و مواردی مشابه برای یکسان‌سازی متون، از اقدامات لازم قبل از شروع پردازش متن می‌باشد.

در ابزار طراحی شده و موجود در سامانه متن کاوی حدود هزار کاراکتر (حرف) با معادل صحیح آن در صفحه کلید استاندارد فارسی جایگزین می‌شود.

سپس مطابق با یک سری قاعده دقیق و مشخص، فاصله‌ها و نیم‌فاصله‌های موجود در متن برای وندهایی نظیر "ها"، "تر" و "ی" غیرچسبان (در انتهای لغات) و همچنین پیشوندها و پسوندهای فعل‌ساز نظیر "نمی"، "می"، "ام"، "ایم"، "اید" و موارد مشابه نیز اصلاح می‌گردند.

نمونه کد نرمال‌سازی فارسی با جایگزینی بیش از ۱۰۰۰ کاراکتر غیراستاندارد و اصلاح فاصله و نیم‌فاصله‌ها بصورت ذیل می‌باشد:

```
url = "http://api.text-mining.ir/api/PreProcessing/NormalizePersianWord"
payload = u"{\"text\":\"ولی اگر دکمه مکث رو لمس کنیم کلا متن چندین صفحه جابه جا میشه و دیگه نمیشه فهمید کدوم آیه تلاوت می شود باید چی کنیم؟.\", \"refineSeparatedAffix\":true}"
print(callApi(url, payload, tokenKey))
```

خروجی :

```
output :
ولی اگر دکمه مکث رو لمس کنیم کلا متن چندین صفحه جابه جا میشه و دیگه نمیشه فهمید کدوم آیه تلاوت می‌شود باید چی کنیم؟
```

تقطیع جملات و واژه‌ها(Sentence Splitter and Tokenizer)

هدف این ابزار شخیص جملات را در متن ورودی است. به عبارت دیگر جداسازی جملات ساده و مرکب (غیرتودرتو) و واژه‌ها و عبارات خاص (توکن‌ها) از یکدیگر هست. پس از پایان مرحله‌ی نرمالسازی متن، ابزار تشخیص‌دهنده‌ی جملات با استفاده از نمادهای (علامت‌های) پایانی جمله از قبیل: ".","؛"، "!"، "؟"، "?"، ... و بکارگیری برخی علائم، قواعد دستوری زبان و در نظرگرفتن حروف ربط یا برخی لغات آغاز کننده‌ی جملات مرکب (از قبیل حروف ربط مانند "که"، "تا"، "اما"، "ولی"، "زیرا"، "سپس"، "و"، "یا"، ... در زبان فارسی)، مرز جمله‌ها را تعیین می‌نماید. تشخیص‌دهنده‌ی لغات (توکن) نیز با استفاده از علامت‌های فضای خالی، "،"، ","، "-" و غیره و در نظر گرفتن اصلاحات اعمال شده در مورد پیشوندها و پسوندها در فاز قبلی، واحدهای با معنی مانند واژه‌ها را شناسایی می‌نماید.

در ابزار طراحی شده و موجود در سامانه متن کاوی، علاوه بر شناسایی واژه‌ها، پردازش ویژه‌ای برای درنظر گرفتن توکن واحد برای کلمات اختصاری (از قبیل A.T.R یا بی.بی.سی)، تاریخ و زمان (از قبیل ۵:۳۵ یا ۲۵/۲/۲۰۱۵)، اعداد اعشاری (از قبیل ۱۷/۵ یا ۵/۱۷)، آدرس صفحات وب، ایمیل و سایر عبارات و علائم خاص (جایگزینی کلمه "ا... " با کلمه اصلی آن) انجام می‌شود.

با توجه به پایه بودن جمله یا توکن در بسیاری از پردازش‌های زبانی، خروجی دقیق این ابزار از درجه‌ی اهمیت بالایی برخوردار است.

نمونه کد این ابزار برای زبان فارسی:

```
url = "http://api.text-mining.ir/api/PreProcessing/SentenceSplitterAndTokenize"
payload = u'''{\"text\": \"من با دوستم به مدرسه می رفتیم و در آنجا مشغول به تحصیل بودیم. سپس به دانشگاه راه یافتیم\",
 \"checkSlang\": true,
 \"normalize\": true,
 \"normalizerParams\": {
 \"text\": \"don't care\",
 \"replaceWildChar\": true,
 \"replaceDigit\": true,
 \"refineSeparatedAffix\": true,
 \"refineQuotationPunc\": false
 },
 \"complexSentence\": false}'''
print(callApi(url, payload, tokenKey))
```

خروجی :

```
output:
[["من","با","دوستم","به","مدرسه","می‌رفتیم"],["و","در","آنجا","مشغول","به","تحصیل","بودیم","."],["سپس","
به","دانشگاه","راه","یافتیم"]]
```

تشخیص کسره اضافه

با آنکه کسرهٔ اضافه در زبان فارسی بازنمایی صوری ندارد و به بیانی در صورت‌بندی زبانی، وزنه‌ای به شمار نمی‌رود، اما به لحاظ کارکردی بسیار ضروری و حائز اهمیت است. از این ابزار در ابزارهای تشخیص موجودیت‌های نامی، قطعه‌بند جملات و ... می‌توان استفاده کرد. شناسایی کسره اضافه از دو رویکرد زبان‌شناسی (بوسیله تعیین نقش کلمات و بدست آوردن درخت تجزیه جملات) و یادگیری ماشین (با استفاده از پیکره برچسب خورده) میسر است.

حذف کلمات توقف(Stop Word Removal)

منظور از حذف کلمات توقف، حذف علائم، اعداد، کلمات عمومی و بدون ارزش معنایی (از قبیل: از، در، با، به، است، پس، ...) در جمله است. در بسیاری از کاربردهای بازیابی اطلاعات، حذف لغات کم‌اهمیت که شاخصه متن نیستند، می‌تواند بدون از بین بردن معنا باعث بهبود دقت و سرعت الگوریتم‌های متن‌کاوی شوند. لیست کلمات توقف وابسته به کاربرد مورد نظر باید تهیه شود. برای مثال: کلمات "هست" و "نیست" برای دسته‌بندی موضوعی متن حائز اهمیت نیستند ولی در تحلیل حس، می‌توانند حس جمله را معکوس کنند.

نمونه کد این ابزار برای زبان فارسی:

```
url = "http://api.text-mining.ir/api/InformationRetrieval/StopWordRemoval"
 payload = u''' "تیم متن کاوی فارسی‌یار با مجموعه‌ای از فارغ التحصیلان دانشگاه‌های صنعتی شریف، تربیت مدرس و فردوسی مشهد از سال ۱۳۹۰ بصورت تخصصی در زمینه پردازش متون برای زبان فارسی، این گروه با آزمایشگاه متن کاوی و یادگیری ماشین پژوهشگاه علوم و فناوری اطلاعات ایران (ایرانداک) همکاری تنگاتنگی داشته است."  '''
print(callApi(url, payload, tokenKey))
```

خروجی:

```
# result:  تیم متن کاوی فارسی‌یار مجموعه‌ای فارغ التحصیلان دانشگاه‌های صنعتی شریف، تربیت مدرس فردوسی مشهد سال ۱۳۹۰ تخصصی پردازش
زبان طبیعی مشغول فعالیت. سال ۱۳۹۶ فعالیت پژوهشی عمیق‌تر پردازش متون زبان فارسی، گروه آزمایشگاه متن کاوی یادگیری ماشین پژوهشگاه علوم فناوری
اطلاعات ایران (ایرانداک) همکاری تنگاتنگی.
```

نمونه کد این ابزار با استفاده از NLTK

```
input_str = "NLTK is a leading platform for building Python programs to work with
human language data."
stop_words = set(stopwords.words('english'))
from nltk.tokenize import word_tokenize
tokens = word_tokenize(input_str)
result = [i for i in tokens if not i in stop_words]
print (result)
```

خروجی:

```
['NLTK', 'leading', 'platform', 'building', 'Python', 'programs', 'work', 'human',
'language', 'data', '.']
```

ریشه‌یابی کلمات یا بُن‌واژه‌یاب(Stemmer and Lemmatizer)

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.

برخلاف زبان انگلیسی، چالش‌های مختلفی هنگام ریشه‌یابی کلمات زبان فارسی وجود دارد از جمله اینکه ضمایر می‌توانند به دو صورت جدا و متصل در جمله ظاهر شوند. البته در مورد افعال مسئله کمی پیچیده‌تر است، بطوری که علاوه بر وندهای فعلی، شخص (فاعل) و زمان جمله نیز بر روی حالت فعل تاثیرگذار هستند. معروفترین الگوریتم ریشه یابی در انگلیسی porter می باشد.

در روش‌های ریشه‌یابی رایج، بعد از حذف انواع وندها (اشتقاقی، تصریفی و واژه‌بست) ممکن است معنای کلمه تغییر یابد. ولی در بُن‌واژه‌یاب (یا Lemmatizer) سعی در ریشه‌یابی بُن کلمه بدون تغییر مفهوم اصلی کلمه در جمله شده است .

همچنین در بن‌واژه‌یاب تولید شده در سامانه متن کاوی، قابلیت تعیین ریشه در چند سطح را دارد. برای مثال ریشه کلمه "دانشجویان" به ترتیب در سطوح مختلف : "دانشجو"، "دانش" و "دان" است. این سطوح مختلف ریشه می‌توانند در کاربردهای مختلف پردازش زبان طبیعی مورد استفاده قرار گیرند. در ابزار ریشه‌یاب موجود در سامانه متن کاوی از دو رویکرد مبتنی بر فرهنگ لغات و قواعد ریخت‌شناسی بهره گرفته شده است. بر پایه تحلیل‌های آماری انجام شده، برای این ابزار از پنج فرهنگ لغت مختلف استفاده شده است.

نمونه کد بن‌واژه‌یابی در زبان فارسی:

```
url = "http://api.text-mining.ir/api/Stemmer/LemmatizeText2Text"
payload = u'"من با شما کارهای زیادی دارم\nمن با دانشجویان دیگری برخورد کردم. سپس به آنها گفتم"'
print(callApi(url, payload, tokenKey))
```

خروجی:

```
من با دانشجو دیگر بر خورد کرد. سپس به آن گفت
من با شما کار زیاد داشت
```

نمونه کد ریشه‌یاب انگلیسی بوسیله کتابخانه NLTK:

```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
stemmer= PorterStemmer()
input_str=”There are several types of stemming algorithms.”
input_str=word_tokenize(input_str)
for word in input_str:
 print(stemmer.stem(word))
```

خروجی :

```
There are sever type of stem algorithm.
```

نمونه کد بن‌واژه‌یابی در زبان انگلیسی بوسیله NLTK:

```
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
lemmatizer=WordNetLemmatizer()
input_str=”been had done languages cities mice”
input_str=word_tokenize(input_str)
for word in input_str:
 print(lemmatizer.lemmatize(word))
```

خروجی :

```
be have do language city mouse
```

توضیحات کد :

```
1. import numpy as np
2. import re
3. import nltk
4. from sklearn.datasets import load_files
5. nltk.download('stopwords')
6. import pickle
7. from nltk.corpus import stopwords
8. import matplotlib.pyplot as plt
9. movie_data = load_files(r"/Users/epsoft/Downloads/datasetv/780429/001.txt")
10. X, y = movie_data.data, movie_data.target
11. documents = []
```

۱. import numpy as np

NumPyیک کتابخانه برای زبان برنامه نویسی پایتون (Python) است. با استفاده از این کتابخانه امکان استفاده از آرایه‌ها و ماتریس‌های بزرگ چند بعدی فراهم می‌شود. هم‌چنین می‌توان از تابع‌های ریاضیاتی سطح بالا بر روی این آرایه‌ها استفاده کرد.

۲. import re

ماژول. Re یکی از پر کاربردترین ماژول ها در پایتون می باشد که تابع های زیر فراخوانی میکند و عمل جستجو در متن را انجام می دهد.

- search
- match
- fullmatch
- findall
- finditer

۳. import nltk

NLTKیک بستر پیشرو برای ساختن برنامه های پایتون برای کار با داده های زبان انسانی است. این رابط کاربری آسان برای بیش از ۵۰ شرکت بزرگ و منابع واژگانی مانند WordNet ، به همراه مجموعه ای از کتابخانه های پردازش متن برای طبقه بندی ، رمزگذاری ، نشانه گذاری ، برچسب زدن ، تجزیه و استدلال معنایی ، بسته های مربوط به کتابخانه های NLP با قدرت صنعتی ، و یک انجمن گفتگوی فعال NLTK به لطف راهنمایی مفید در معرفی اصول برنامه نویسی در کنار مباحث مربوط به زبان شناسی محاسباتی ، به علاوه اسناد جامع API ، به طور یکسان برای زبان شناسان ، مهندسان ، دانشجویان ، آموزگاران ، محققان و کاربران صنعت مناسب است.

۴. scikit-learn

به همراه چند دیتاست استاندارد ارائه شده است، به عنوان مثال: مجموعه داده iris و اعداد برای طبقه بندی و مجموعه داده دیابت برای رگرسیون.

۵. nltk.download ‘stopword”

تعریف و لود stopword

۶. import pickle

بعنوان یک توسعه دهنده ، ممکن است شما نیاز داشته باشید که یک آبجکت پیچیده را از طریق شبکه بفرستید یا وضعیت داخلی آبجکت‌های خود را برای استفاده بعدی در دیسک یا دیتابیس ذخیره کنید. برای تحقق این امر می توانید از فرایندی به نام **serialization** استفاده کنید که به لطف ماژول **pickle** پایتون به طور کامل توسط کتابخانه استاندارد پشتیبانی می شود.

۷. import nltk.corpus

برای کار با NLTK لازم است تا در ابتدا مجموعه‌ای از متون را دانلود کنیم. این مجموعه متون که با نام corpus نیز شناخته می‌شوند، از طریق NLTK قابل دستیابی هستند. یک – corpus که صورت جمع آن corpora است – در Wikipediaبه صورت زیر تعریف می‌شود:

به مجموعه‌ای خام از داده‌های زبانیِ نوشتاری یا گفتاری گفته می‌شود که می‌توان در توصیف و تحلیل زبان از آن بهره گرفت.

بنابراین می‌توان گفت یکcorpus ، حجم وسیعی از فایل‌های متنی را شامل می‌شود.

۸. import matplotlib.pyplot

یکی از راه‌های رسم نمودار در پایتون ، استفاده از کتابخانه matplotlib در پایتون است. کتابخانه‌های دیگری نیز برای رسم نمودار در پایتون وجود دارند، اما matplotlib پیشکسوت همه آن‌ها است. با استفاده از matplotlib شما می‌توانید انواع نمودارها را رسم کنید. نمودارهای دوبعدی، سه بعدی، نمودار میله ای، نمودار هیستوگرام، scatter plotو ... . در ادامه خواهیم گفت matplotlibچیست و چگونه می‌توان با کمک این کتابخانه رسم نمودار در پایتون را انجام داد.

۹. import matplotlib.pyplot as plt

برای رسم یک خط در کتابخانه‌ی مت پلات مراحل زیر را داریم:

- تعریف محور x و مقادیر متناظر در محور y به صورت لیست‌های جداگانه
- رسم آنها بر روی صفحه با تابعplot
- اختصاص نام به محورهای x و y با توابع xlabel وylabel
- دادن عنوان به نمودار با تابعtitle
- در پایان تمامی کدها در matplotlib بکارگیری تابعplt.show

۱۰. import WordNetLemmatizer

این پکیج بسته به نوع استفاده از آن به عنوان اسم یا فعل، شکل اصلی کلمه را استخراج می کند.

۱۱. X, y = movie_data.data, movie_data.target

movie_data.data در متغیر x و movie_data.taget در متغیر y تعریف میکنیم.

```
from nltk.stem import WordNetLemmatizer

stemmer = WordNetLemmatizer()

for sen in range(0, len(X)):
    # Remove all the special characters
    document = re.sub(r'\W', ' ', str(X[sen]))

    # remove all single characters
    document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

    # Remove single characters from the start
    document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

    # Substituting multiple spaces with single space
    document = re.sub(r'\s+', ' ', document, flags=re.I)

    # Removing prefixes d 'b'
    document = re.sub(r'^b\s+', '', document)

    # Converting to Lowercase
    document = document.lower()

    # Lemmatization
    document = document.split()

    document = [stemmer.lemmatize(word) for word in document]
    document = ' '.join(document)

    documents.append(document)
```

کد بالا با استفاده از هشتگ کامنت گذاری شده (استاپ ورد ورودی شناسایی میکند)

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(max_features=1500, min_df=5, max_df=0.7,
stop_words=stopwords.words('english'))
X = vectorizer.fit_transform(documents).toarray()
from sklearn.feature_extraction.text import TfidfTransformer
tfidfconverter = TfidfTransformer()
X = tfidfconverter.fit_transform(X).toarray()

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
import sklearn.ensemble as se
classifier = se.RandomForestClassifier(n_estimators=1000, random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
plt.plot(X_test,y_pred)
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print(accuracy_score(y_test, y_pred))
```

الگوریتم TF-IDF یکی از الگوریتم‌های محاسبه امتیاز ارتباط (relevance score) است که در زمینه‌ی جستجو و تحلیل داده‌های متنی مورد استفاده قرار می‌گیرد. نام این الگوریتم مخفف عبارت‌های زیر است:

TF = Term Frequency

IDF = Inverse Document Frequency

این الگوریتم امتیاز میزان ارتباط یک term (T) در یک document (D) را طبق فرمول زیر محاسبه می‌کند:

score(D، T) = termFrequency(T، D) * log(N / docFrequency(T))

در این فرمول فاکتورهای زیر موثر است:

termFrequency: این فاکتور تعداد تکرار term در یک document را محاسبه می‌کند.

N: تعداد کل document های موجود در مجموعه‌ی هدف برای جستجو

docFrequency:این فاکتور تعداد تکرار term در کل document های مجموعه‌ی مورد جستجو را محاسبه می‌کند.

همانطور که از فرمول بالا مشخص است، عامل termFrequency تاثیر مثبت در میزان اهمیت (ارتباط) یک term در یک document داشته و عامل docFrequency اثر منفی در اهمیت خواهد داشت. دلیل استفاده از عبارت Inverse) Document Frequencyمعکوس تعداد تکرار در document ها) در نام این الگوریتم نیز همین مساله است.

از جمله نواقص این الگوریتم می‌توان به موارد زیر اشاره کرد:

ـاثر نامطلوب term های مورد جستجو بر یکدیگر: زمانی که عبارت مورد جستجو شامل چندین term باشد، تکرار بیش از اندازه‌ی یکی از term ها می‌تواند باعث افزایش امتیاز نهایی یک document شود در حالیکه ممکن است اهمیت آن term در document زیاد نباشد. برای مثال زمانی که عبارت "آموزش با سکان‌آکادمی" جستجو شود، تکرار بسیار زیاد کلمه‌ی "با" می‌تواند اثر نامطلوب در رتبه‌بندی نتایج جستجو داشته باشد و نتایجی که شامل هر سه term مورد نظر هستند در رتبه‌های پایین‌تر نسبت به document هایی قرار گیرند که تعداد زیادی کلمه‌ی "با" در آن‌ها تکرار شده است !

ـعدم در نظر گرفتن طول متن :document با درنظر گرفتن این حقیقت که طولانی‌تر بودن یک متن به صورت ضمنی شانس بیشتری را برای تکرار یک term در آن ایجاد می‌کند، می‌توان گفت که در برخی موارد ممکن است تکرار بیشتر یک term در document الزامی به ارتباط بیشتر آن document با term مورد نظر نداشته باشد. برای مثال ۲ مرتبه تکرار کلمه‌ی "آموزش" در یک متن با طول ۱۰۰ اهمیت بیشتری نسبت به ۲ بار تکرار این کلمه در یک متن با طول ۵۰۰ خواهد داشت.

در ادامه کد بالا داده تست 0.2 قرار دادیم یعنی 200 داده ی تست از 1000 تا برای تست قرار دادیم.

خروجی :

	precision	recall	f1-score	support
0	0.86	0.87	0.86	208
1	0.85	0.84	0.85	192
accuracy			0.85	400
macro avg	0.85	0.85	0.85	400
weighted avg	0.85	0.85	0.85	400

0.855

خروجی :