

تمرین اول NLP

یاسمین مدنی

۹۷۵۳۲۲۶۵



فهرست

| | |
|----|---------------------------|
| ۲ | سوالات تئوری |
| ۲ | سوال ۱ |
| ۲ | Natural Language Tool Kit |
| ۲ | fastText |
| ۲ | SpaCy |
| ۳ | learn-Scikit |
| ۳ | CoreNLP |
| ۴ | سوال ۲ |
| ۶ | سوال ۳ |
| ۸ | سوالات عملی |
| ۸ | سوال ۱ |
| ۸ | سوال ۲ |
| ۹ | سوال ۳ |
| ۱۰ | سوال ۴ |
| ۱۱ | سوال ۵ |
| ۱۳ | سوال ۶ |

سوالات تئوری

سوال ۱

Natural Language Tool Kit

یکی از محبوب ترین کتابخانه های NLP در پایتون است. از تعداد زیادی کار پشتیبانی می کند و می تواند برای انجام هر کاری از تکنیک های پیش پردازش متن مانند توقف حذف کلمه، توکن سازی، ریشه یابی و واژه سازی تا ساخت n-gram استفاده شود.

NLTK رابط کاربری آسانی را برای بیش از ۵۰ مجموعه و منبع واژگانی فراهم می کند. این ابزار دارای عملکردهای ضروری مورد نیاز برای تقریباً همه انواع وظایف پردازش زبان طبیعی با پایتون است و تقریباً هر مؤلفه ای از NLP را که نیاز دارید اجرا می کند، مانند طبقه بندی، نشانه گذاری، ریشه یابی، برچسب گذاری، تجزیه و استدلال معنایی. اغلب بیش از یک پیاده سازی برای هر کدام وجود دارد، بنابراین می توانید الگوریتم یا روش دقیقی را که می خواهید استفاده کنید انتخاب کنید. همچنین از بسیاری از زبان ها پشتیبانی می کند.

fastText

یک کتابخانه منبع باز، رایگان و سبک برای یادگیری جاسازی کلمات و طبقه بندی متن است که توسط آزمایشگاه تحقیقاتی هوش مصنوعی فیس بوک ایجاد شده است. این مدل به فرد امکان می دهد یک الگوریتم یادگیری بدون نظارت یا یادگیری نظارت شده برای بدست آوردن نمایش های برداری برای کلمات ایجاد کند. فیس بوک مدل های از پیش آموزش دیده را برای ۲۹۴ زبان در دسترس قرار می دهد.

SpaCy

سریع، آسان برای استفاده، به خوبی مستند شده، و برای پشتیبانی از حجم زیادی از داده ها طراحی شده است، دارای مجموعه ای از مدل های NLP از پیش آموزش دیده است که کار شما را آسان تر می کند. برخلاف NLTK یا CoreNLP که تعدادی الگوریتم را برای هر کار نمایش می دهند، SpaCy منوی خود را کوتاه نگه می دارد و بهترین گزینه موجود را برای هر کار در دست ارائه می دهد.

اگر می خواهید متنی را برای یادگیری عمیق آماده کنید، این کتابخانه یک گزینه عالی است و در کارهای استخراج عالی است. در حال حاضر، فقط به زبان انگلیسی در دسترس است.

Scikit-learn

یک کتابخانه NLP منبع باز رایج در میان دانشمندان داده به دلیل مستندات عالی است. علاوه بر این، روش های کلاس بصری را ارائه می دهد و الگوریتم های متعددی را برای ساخت مدل های یادگیری ماشین ارائه می دهد. با این حال، Scikit-learn شبکه های عصبی را برای پردازش متن ارائه نمی دهد.

CoreNLP

هدف آن این است که استفاده از ابزارهای تحلیل زبانی را برای یک متن آسان و کارآمد کند. با CoreNLP، می توانید انواع ویژگی های متن (مانند تشخیص هویت نام، برچسب گذاری بخشی از گفتار، و غیره) را تنها در چند خط کد استخراج کنید.

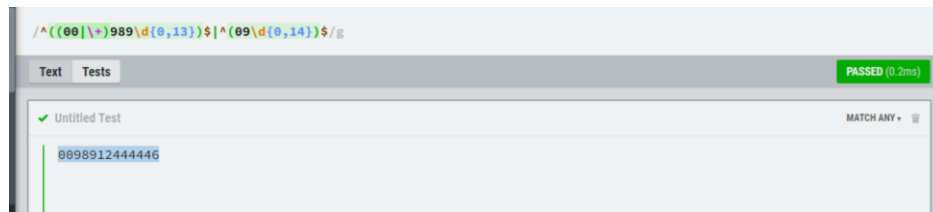
از آنجایی که CoreNLP به زبان جاوا نوشته شده است، نیاز به نصب جاوا بر روی دستگاه شما دارد. با این حال، رابط های برنامه نویسی را برای بسیاری از زبان های برنامه نویسی محبوب از جمله پایتون ارائه می دهد. این ابزار شامل چندین ابزار NLP استندفورد مانند تجزیه کننده، تجزیه و تحلیل احساسات، یادگیری الگوی بوت استرپ، برچسب بخشی از گفتار (POS) و... است.

علاوه بر این، CoreNLP از زبان های عربی، چینی، آلمانی، فرانسوی و اسپانیایی علاوه بر انگلیسی پشتیبانی می کند.

سوال ۲

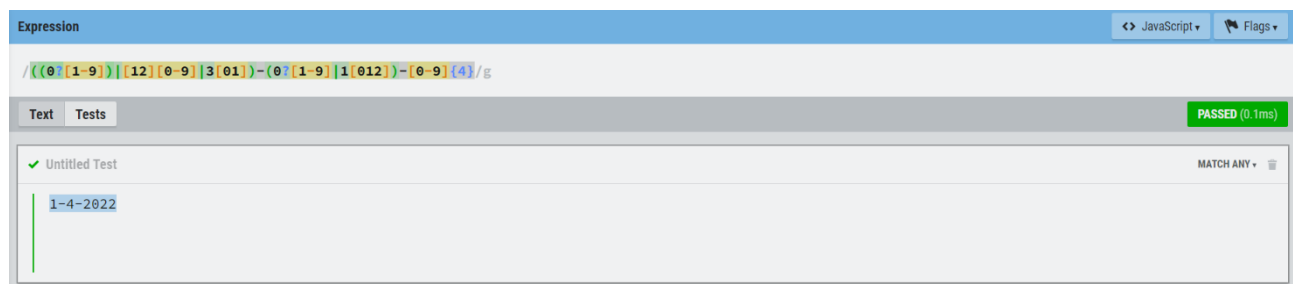
(الف)

$\wedge((00|\backslash+)989\d{0,13})\$|\wedge(09\d{0,14})\$$



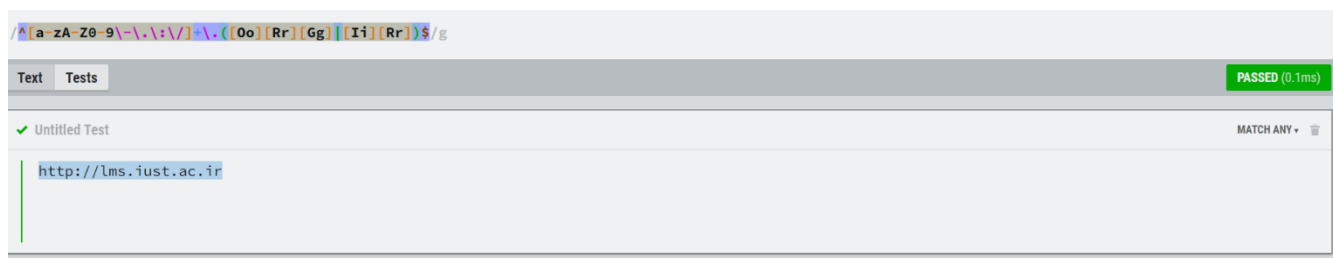
(ب)

$\wedge((0?[1-9])|[12][0-9]|3[01])-(0?[1-9]|1[012])-[0-9]{4}\$$



(ج)

$\wedge[a-zA-Z0-9\-\.\:\backslash]+ \backslash. ([Oo][Rr][Gg]|[Ii][Rr])\$$



(٥

`^[1-9][1-9][A-Z][1-9][1-9][1-9]IR[1-9][0-9]$`

`/^[1-9][1-9][A-Z][1-9][1-9][1-9]IR[1-9][0-9]$/g`

TextTests

PASSED (0.1ms)

✓Untitled Test

MATCH ANY ▾

54M235IR44

سوال ۳

روش Maximal Matching

یکی از راه‌های جلوگیری از تطابق کوتاه‌ترین کلمه، یافتن طولانی‌ترین دنباله کاراکترها در فرهنگ لغت است. این روش یک الگوریتم حریصانه است که با طولانی‌ترین کلمه مطابقت دارد. مثلاً در زبان انگلیسی ما این سری کاراکتر را داریم:

“themendinehere”

برای کلمه اول کلمات زیر را می‌یابیم و بعد از آن کلمه‌ای پیدا نمی‌شود.

the, them, theme

حالا ما طولانی‌ترین را انتخاب می‌کنیم که «theme» است، سپس دوباره از «n» شروع می‌کنیم. اما در حال حاضر ما هیچ کلمه‌ای در رشته زیر نداریم.

“ndineh...”

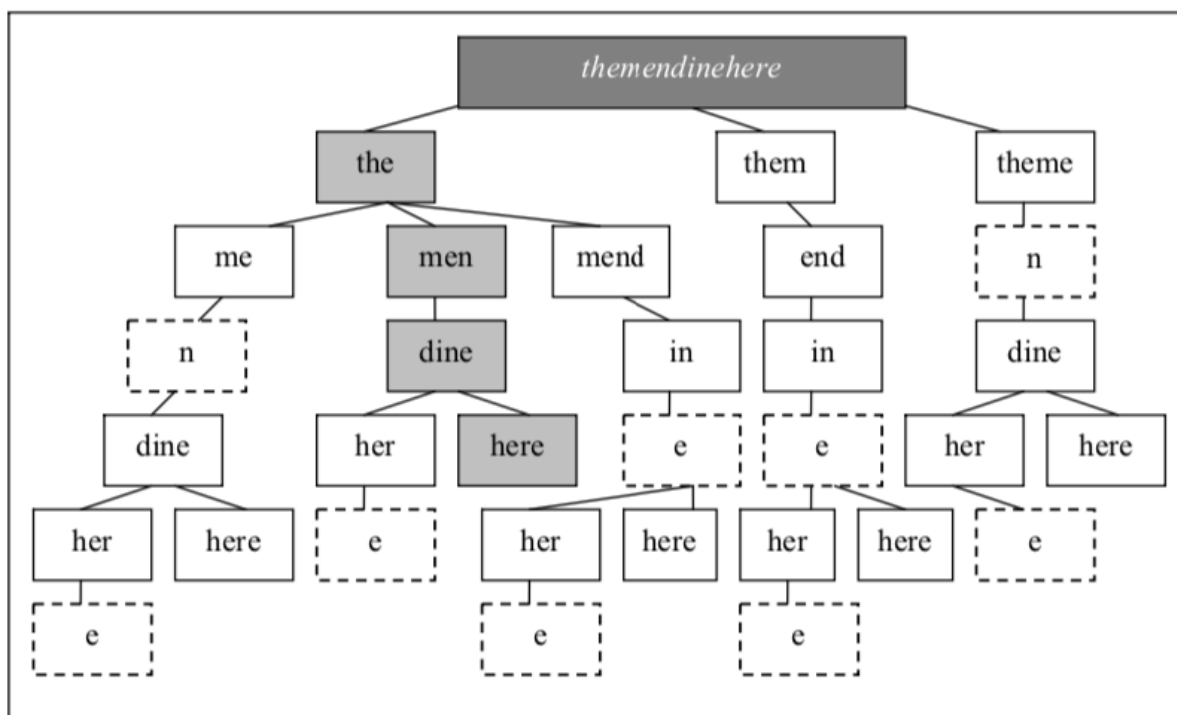
وقتی نمی‌توانیم یک کلمه را مطابقت دهیم، فقط اولین کاراکتر را به عنوان ناشناخته علامت گذاری می‌کنیم. پس اینجا n را به عنوان ناشناخته علامت گذاری و جدا می‌کنیم و کلمه بعد را با d شروع می‌کنیم. بافرض آنکه "din" یا "re" در فرهنگ لغت ما نباشد، مجموعه کلمات را به عنوان "theme n dine" به دست می‌آید.

اما همانطور که می‌بینید طولانی‌ترین کلمه می‌تواند تقسیم‌بندی نادرستی ایجاد کند. این امر منجر به گسترش بیش از حد کلمه اول "theme" به بخشی از کلمه دوم "men" و ناشناخته شدن کلمه بعدی می‌شود.

روش Maximum Matching

روش دیگر برای حل ماهیت حریصانه Maximal Matching الگوریتمی به نام «تطبیق حداکثر» یا Maximum Matching است.

این رویکرد چندین احتمال را تقسیم‌بندی می‌کند و گزینه‌ای را انتخاب می‌کند که کلمات کمتری در جمله داشته باشد. همچنین داشتن کلمات ناشناخته کمتر را در اولویت قرار می‌دهد. با استفاده از این رویکرد، تقسیم‌بندی صحیح را از متن مثال مانند شکل زیر بدست می‌آوریم.



اولین کلمه می تواند "the", "them", "theme" باشد. از هر یک از این گره ها، چندین انتخاب وجود دارد که همانطور که مشاهده می شود سایر مسیرها به تولید کلمات ناشناخته منجر شده و کارآمد نیستند. این رویکرد از طریق تمام ترکیبات مختلف بر اساس فرهنگ لغت ما انجام می شود. بنابراین کلمات ناشناخته هنوز مشکلی است که به آن رسیدگی نکرده ایم.

علاوه بر این، این روش نیز می تواند دارای مشکل باشد اگر متن مورد بررسی تعداد کلمات بیشتری داشته باشد اما الگوریتم مسیری با تعداد کلمات کمتر پیدا کرده باشد.

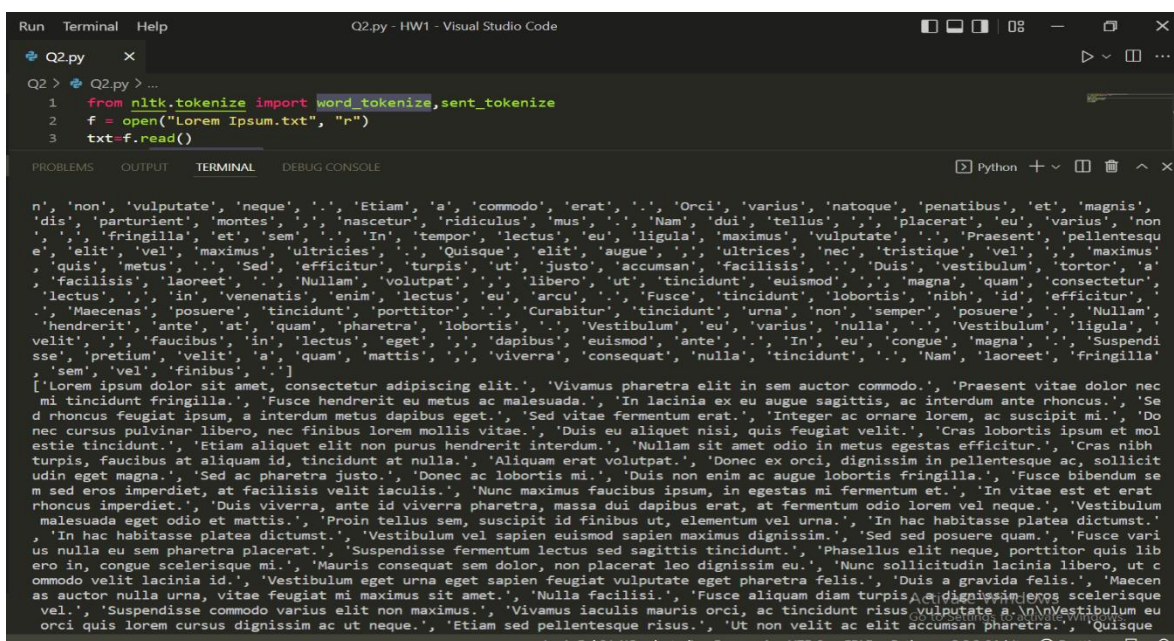
سوالات عملی

سوال ۱

کلمات مورد نظر در فایلی با نام MyFile.txt در پوشه Q1 ذخیره شده اند. پاسخ ها که همگی مثبت اند نمایش دهنده پذیرفته شدن عبارات توسط برنامه است.

سوال ۲

فایل Lorem Ipsum.txt فایلی متشکل از کلمات و جملات بی معنی و تصادفی است که قالب پاراگرافی دارد و در پوشه Q2 قرار دارد. استفاده از (`word_tokenize()`) رشته را بر اساس کلمات(نقاط پایانی و فاصله ها و...) تقسیم می کند اما استفاده از (`sent_tokenize()`) موجب می شود تا جملات متن جدا شوند. با اجرای فایل خروجی برنامه قابل مشاهده است.



```
Run Terminal Help Q2.py - HW1 - Visual Studio Code
Q2.py
Q2 > Q2.py > ...
1 from nltk.tokenize import word_tokenize,sent_tokenize
2 f = open("Lorem Ipsum.txt", "r")
3 txt=f.read()

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Python
n','non','vulputate','neque','Etiam','a','commodo','erat','Orci','varius','natoque','penatibus','et','magnis','dis','parturient','montes','nascetur','ridiculus','mus','Nam','dui','tellus','placemat','eu','varius','non','fringilla','et','sem','In','tempor','lectus','eu','ligula','maximus','vulputate','Praesent','pellentesque','elit','vel','maximus','ultrices','Quisque','elit','augue','ultrices','nec','tristique','vel','maximus','quis','metus','Sed','efficitur','turpis','ut','justo','accumsan','facilisis','Duis','vestibulum','tortor','a','facilisis','laoreet','Nullam','volutpat','libero','ut','tincidunt','eismod','magna','quam','consectetur','lectus','in','venenatis','enim','lectus','eu','arcu','Fusce','tincidunt','lobortis','nibh','id','efficitur','Maecenas','posuere','tincidunt','porttitor','Curabitur','tincidunt','urna','non','semper','posuere','Nullam','hendrerit','ante','at','quam','pharetra','lobortis','Vestibulum','eu','varius','nulla','Vestibulum','ligula','velit','faucibus','in','lectus','eget','dapibus','eismod','ante','In','eu','congue','magna','Suspendisse','pretium','velit','a','quam','mattis','viverra','consequat','nulla','tincidunt','Nam','laoreet','fringilla','sem','vel','finibus']
[Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus pharetra elit in sem auctor commodo. Praesent vitae dolor nec mi tincidunt fringilla. Fusce hendrerit eu metus ac malesuada. In lacinia ex eu augue sagittis, ac interdum ante rhoncus. Sed rhoncus feugiat ipsum, a interdum metus dapibus eget. Sed vitae fermentum erat. Integer ac ornare lorem, ac suscipit mi. Donec cursus pulvinar libero, nec finibus lorem mollis vitae. Duis eu aliquet nisi, quis feugiat velit. Cras lobortis ipsum et mollis estie tincidunt. Etiam aliquet elit non purus hendrerit interdum. Nullam sit amet odio in metus egestas efficitur. Cras nibh turpis, faucibus at aliquam id, tincidunt at nulla. Aliquam erat volutpat. Donec ex orci, dignissim in pellentesque ac, sollicitudin eget magna. Sed ac pharetra justo. Donec ac lobortis mi. Duis non enim ac augue lobortis fringilla. Fusce bibendum sem sed eros imperdiet, at facilisis velit iaculis. Nunc maximus faucibus ipsum, in egestas mi fermentum et. In vitae est et erat rhoncus imperdiet. Duis viverra, ante id viverra pharetra, massa dui dapibus erat, at fermentum odio lorem vel neque. Vestibulum malesuada eget odio et mattis. Proin tellus sem, suscipit id finibus ut, elementum vel urna. In hac habitasse platea dictumst. In hac habitasse platea dictumst. Vestibulum vel sapien eismod sapien maximus dignissim. Sed sed posuere quam. Fusce varius nulla eu sem pharetra placerat. Suspendisse fermentum lectus sed sagittis tincidunt. Phasellus elit neque, porttitor quis libero in, congue scelerisque mi. Mauris consequat sem dolor, non placerat leo dignissim eu. Nunc sollicitudin lacinia libero, ut commodo velit lacinia id. Vestibulum eget urna eget sapien feugiat vulputate eget pharetra felis. Duis a gravida felis. Maecenas auctor nulla urna, vitae feugiat mi maximus sit amet. Nulla facilisi. Fusce aliquam diam turpis, a dignissim eros scelerisque vel. Suspendisse commodo varius elit non maximus. Vivamus iaculis mauris orci, ac tincidunt risus vulputate a. Vestibulum eu orci quis lorem cursus dignissim ac ut neque. Etiam sed pellentesque risus. Ut non velit ac elit accumsan pharetra. Quisque
```

سوال ۳

کد مربوط به تمرین در پوشه Q3 قرار دارد.

در این سوال ما لیستی از `incorrect_words` را تعریف می کنیم که برای آنها به املای صحیح نیاز داریم. سپس یک حلقه برای هر کلمه در لیست کلمات نادرست اجرا می کنیم که در آن فاصله ژاکارد کلمه نادرست را محاسبه می کنیم که هر کلمه املای صحیح همان حرف اولیه را به شکل `bi-gram` کاراکترها داشته باشد. سپس آنها را به ترتیب صعودی مرتب می کنیم تا کمترین فاصله در بالا باشد و کلمه مربوط به آن را استخراج کرده و چاپ می کنیم.

```
incorrect_words=['looove', 'correct', 'happy']
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
```

```
PS E:\uni\term8\nlp\Hws\HW1\Q2> & C:/Users/ASUS/AppData/Local/Programs/Python/Python38-32/Python.exe C:/Users/ASUS/AppData/Local/Programs/Python/Python38-32/Scripts/ipython.exe
love
correct
happy
PS E:\uni\term8\nlp\Hws\HW1\Q2>
```

نتایج موارد پ و ت در دو عکس زیر آمده است.

[illegible]

```
PS E:\uni\term8\nlp\Hws\HW1\Q4> & C:\Users\ASUS\AppData\Local\Programs\Python\Python39\python.exe e:\uni\term8\nlp\Hws\HW1\Q4/Q4.py  
['14', '1879', '1896', '1901', '1905', '1908', '1909', '1911', '1914', '1914', '1933', '1940', '1945']  
['Hello', '!', 'Hope', 'you', '"re', 'doing', 'well', '.', 'It', 'is', 'a', 'sample', 'short', 'text', '.This', 'is', 'our', 'I', 'st'  
, 'assignment', '.']  
['.', 'وتصا' و'ام' و'ا' و'نَیَرَمَتْ و'نَیا' و'تصا' و'هَنوْمَن و'هَاتَوک و'نَجْم و'کِی و'نَیا و'. و'دیشاب' و'بُوح و'مَراودِم و'ا' و'عَالِی'  
PS E:\uni\term8\nlp\Hws\HW1\Q4>
```

پایاده سازی WhitespaceTokenizer با استفاده از RegexpTokenizer در کد پایاده سازی شده است. تصویر زیر نتیجه حاصل است.

```
58 # WordPunct_Tokenizer=WordPunctTokenizer()
59 # ShortSampleEnglish.tokenize=word_tokenize(word_tokenize(ShortSampleEnglish.txt))
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE Python + - [] [X] [Y] [Z]

```
PS E:\uni\term8\nlp\Hws\HW1\Q4> & C:/Users/ASUS/AppData/Local/Programs/Python/Python39/python.exe e:/uni/term8/nlp/Hws/HW1/Q4/Q4.py
['Hello!', 'Hope', 'you're', 'doing', 'well.', 'It', 'is', 'a', 'sample', 'short', 'text.This', 'is', 'our', '1', 'st', 'assignment.']
['Hello!', 'Hope', 'you're', 'doing', 'well.', 'It', 'is', 'a', 'sample', 'short', 'text.This', 'is', 'our', '1', 'st', 'assignment.']
PS E:\uni\term8\nlp\Hws\HW1\Q4>
```

این تابع برحسب علائم نگارشی رشته را توکنایز میکند و عبارت منظم معادل آن $\backslash w+|[^w\backslash s]+$ می باشد، نتیجه در تصویر زیر قابل مشاهده است

```
PS E:\uni\term8\nlp\Hws\HW1\Q4> & C:/Users/ASUS/AppData/Local/Programs/Python/Python39/python.exe e:/uni/term8/nlp/Hws/HW1/Q4/Q4.py
['Hello', '!', 'Hope', 'you', '', 're', 'doing', 'well', '.', 'It', 'is', 'a', 'sample', 'short', 'text', '.', 'This', 'is', 'our', '1', 'st', 'assignment', '.']
PS E:\uni\term8\nlp\Hws\HW1\Q4>
```

سوال ۵

Porter stemmer

پنج مرحله کاهش کلمات در این روش استفاده می شود که هر کدام مجموعه ای از قوانین نقشه برداری خاص خود را دارد. غالباً ریشه حاصل کلمه کوتاه تری با همان ریشه است.

LancasterStemmer

لنکستر ساده است، اگرچه اغلب با ریشه گیری بیش از حد ریشه های غیر زبانی یا بی معنی را ارائه می دهد.

نتیجه اجرای کد تمرین ۵ در شکل زیر آمده است.

```
PS E:\uni\term8\nlp\Hws\HW1> cd Q5
PS E:\uni\term8\nlp\Hws\HW1\Q5> & C:/Users/ASU
Stemming using PorterStemmer
was==> wa
Germany==> germani
weeks==> week
later==> later
family==> famili
moved==> move
Italy==> itali
-----
Stemming using LancasterStemmer
was==> was
Germany==> germany
weeks==> week
later==> lat
family==> famy
moved==> mov
Italy==> ita
PS E:\uni\term8\nlp\Hws\HW1\Q5>
```

لمتایزر با آرگومان های پیش فرض به خوبی پاسخ گو نیست چرا که تمامی واژگان را مانند یک اسم در نظر می گیرد و از این رو در ارائه پاسخ مناسب برای فعل ها یا سایر انواع کلمات بازمی ماند. اما با استفاده از تشخیص pos کلمه و استفاده از آن می توانیم قدرت آن را بیشتر کنیم. برای مثال در تصویر زیر پیش و پس از استفاده از pos را مشاهده می کنیم.

```
lemmatize with deaafult values
Waves==>Waves
fishing==>fishing
rocks==>rock
was==>wa
corpora==>corpus
better==>better
ate==>ate
broken==>broken
-----
lemmatize with giving apropriate POS
Waves==>Waves
fishing==>fish
rocks==>rock
was==>be
corpora==>corpora
better==>good
ate==>ate
broken==>broken
PS E:\uni\term8\nlp\Hws\HW1\Q5>
```

البته با وجود استفاده از pos کلمات باز هم می بینیم بعضی از کلمات به خوبی نتیجه نداده اند چرا که pos آن ها به گونه ای متفاوت از انتظار ما تعیین شده است.

('Waves', 'NNS'), ('fishing', 'VBG'), ('rocks', 'NNS'), ('was', 'VBD'), ('corpora', 'RB'), ('better', 'JJR'), ('ate', 'NN'), ('broken', 'NN')

برای مثال broken یک اسم تلقی می شود.

سوال ۶

به دلیل داده‌های زیاد و زمان بر بودن نمونه‌های موجود در پوشه و عکس‌ها مربوط به ۲۰۰۰۰ داده است که البته به راحتی می‌توان با برداشتن شرط تعداد و یا افزایش تعداد مجاز کل داده‌ها را بررسی کرد.

به ترتیب مراحل گفته شده در داک تمرین و به کمک عبارات منظم بخشی از پیش پردازش را مانند حذف کاراکترهای اضافی و.. صورت می‌پذیرد.

سپس با توکنایز کردن هر توییت اعمال سری دیگری از شرط‌های مربوط به کلمات و ریشه‌گیری از آن‌ها توییت پاکسازی شده را از جویین کردن کلمات به‌دست آورده و در فایل `final` به همراه توییت اولیه می‌نویسیم. `word_tokenize` رشته را بر اساس علائم نگارشی جمله تقسیم می‌کند.

`stop words` کلماتی در هر زبانی هستند که معنای زیادی به جمله اضافه نمی‌کنند. آن‌ها را می‌توان با خیال راحت بدون قربانی کردن معنای جمله نادیده گرفت. معمولاً به رایج‌ترین کلمات در یک زبان اشاره دارد. هیچ لیست جهانی از "کلمات توقف" وجود ندارد که توسط همه ابزارهای NLP مشترک استفاده شود. کلمات توقف اغلب قبل از آموزش مدل‌های یادگیری عمیق و یادگیری ماشین از متن حذف می‌شوند، این کلمات به تعداد زیاد در متون استفاده می‌شود بنابراین اطلاعات منحصر به فرد کمی ارائه می‌کنند که می‌تواند برای طبقه‌بندی یا خوشه‌بندی استفاده شود.

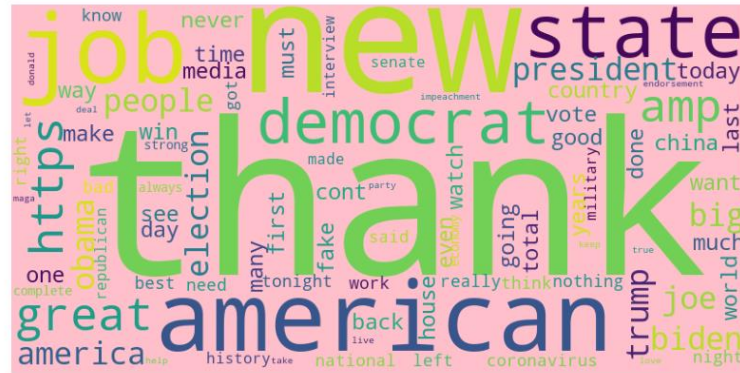
همان‌طور که پیشتر با روش‌های پورتر و لنکستر و تفاوت‌های آن‌ها در سوال‌های قبل آشنا شدیم از آنجا که پورتر ملایم‌تر عمل می‌کند و محبوب‌تر و رایج‌تر است در این تمرین از پورتر برای یافتن ریشه کلمات استفاده می‌کنیم.

نمونه توییت‌ها در حین پردازش:

1. The threshold identification of Ballots is turning out to be even bigger than originally anticipated. A very large number of Ballots are impacted. Stay tuned!
2. The threshold identification of Ballots is turning out to be even bigger than originally anticipated. A very large number of Ballots are impacted. Stay tuned!
3. the threshold identification of ballots is turning out to be even bigger than originally anticipated. a very large number of ballots are impacted. stay tuned!

1. RT @RealRLimbaugh: Winning, winning, winning! @realDonaldTrump
2. RT : Winning, winning, winning!
3. rt : winning, winning, winning!
4. win win win

بیشترین کلمات تکرار شده در تصویر cloud به نمایش گذاشته شده اند. نتایج زیر تگ های ترند و تعداد آن هم چنین بیشترین کلمات تکرار شده به همراه تعداد می باشد.



Comon_tags:

[('maga', 241), ('timetogettough', 95), ('covid19', 67), ('trumpvlog', 63), ('kag2020', 41), ('2a', 41), ('coronavirus', 38), ('1', 31), ('trump', 24), ('paycheckprotectionprogram', 24)]

Comon_words:

[('https', 5433), ('great', 2108), ('http', 1901), ('amp', 1834), ('president', 1631), ('trump', 1555), ('people', 1172), ('thank', 1036), ('biden', 876), ('new', 857), ('obama', 810), ('get', 744), ('news', 725), ('would', 703), ('america', 695), ('big', 681), ('like', 670), ('joe', 662), ('election', 650), ('country', 647), ('democrats', 621), ('today', 619), ('never', 614), ('vote', 600), ('one', 591), ('american', 575), ('china', 556), ('time', 543), ('many', 535), ('going', 520), ('good', 516), ('fake', 494), ('must', 483), ('want', 480), ('make', 477), ('even', 469), ('years', 454), ('job', 452), ('thanks', 452), ('see', 444), ('state', 444), ('back', 438), ('cont', 432), ('states', 416), ('win', 414), ('house', 409), ('media', 399), ('last', 397), ('jobs', 390), ('way', 389), ('total', 388), ('much', 383), ('first', 355), ('watch', 353), ('world', 346), ('done', 343), ('day', 340), ('ever', 331), ('nothing', 329), ('national', 329), ('know', 328), ('best', 328), ('need', 327), ('coronavirus', 326), ('really', 325), ('night', 323), ('work', 322), ('said', 318), ('tonight', 310), ('got', 310), ('right', 308), ('bad', 308), ('think', 307), ('left', 306), ('history', 297), ('republican', 295), ('made', 292), ('interview', 284), ('senate', 283), ('military', 283), ('complete', 283), ('strong', 282), ('always', 281), ('economy', 280), ('also', 279), ('impeachment', 273), ('donald', 271), ('endorsement', 271), ('americans', 269), ('let', 266), ('love', 263), ('maga', 263), ('democrat', 262), ('take', 262), ('deal', 262), ('keep', 259), ('live', 257), ('help', 257), ('true', 254), ('party', 253)]