گزارش پروژه پایانی – فاز اول

استخراج رابطه در زبان فارسی با بررسی وابستگی جهانی *Seraji* و *PerDT*

مبانی پردازش زبان و گفتار

استاد: دکتر بهروز مینایی بیدگلی

دانیال بازمانده – محمدحسین کریمیان

در این پروژه هدف اصلی یافتن روابط معنایی در زبان فارسی است به این صورت که بین دو *entity*، یک رابطه استخراج می-کنیم.

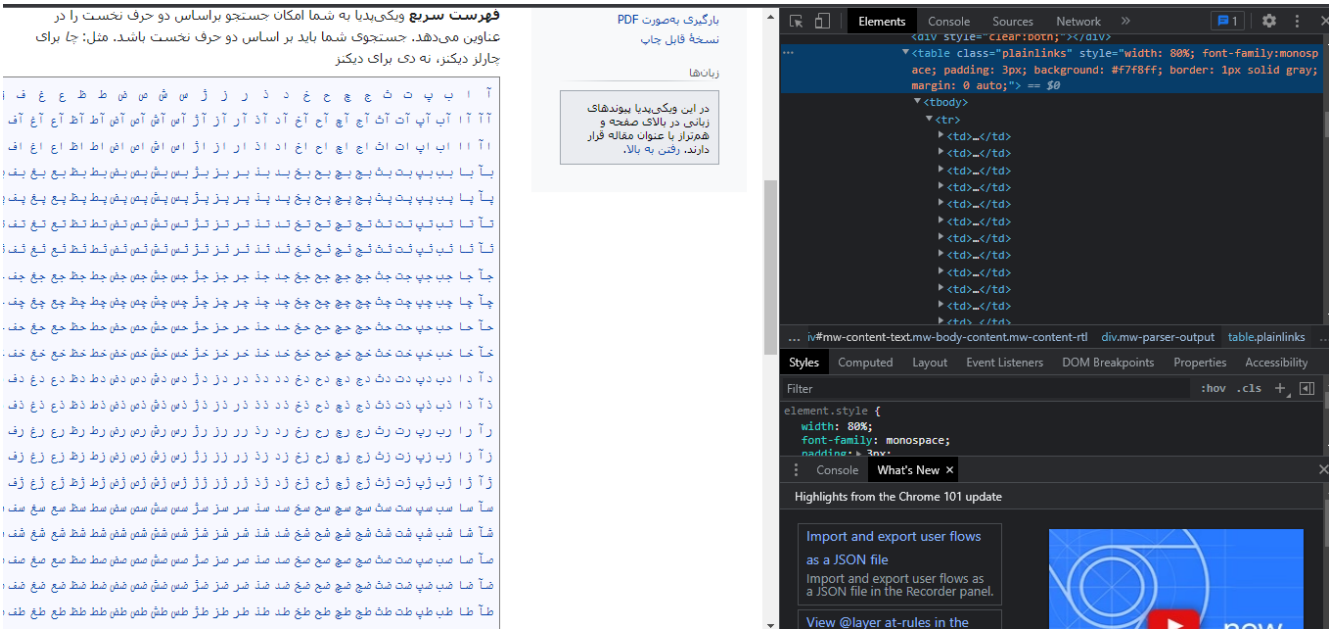
در این گام که گام اول پروژه است، ابتدا باید داده های زیادی به دست بیاوریم زیرا برای رسیدن به هدف نهاییمان، به مجموعه بزرگی از داده ها نیاز داریم.سپس باید این داده ها را *normalize* کنیم و با حذف *stop words* ها و پیدا کردن ریشه کلمات، تعداد تکرار شدن هر کلمه را بدست می آوریم و جمله ها را هم مشخص می کنیم.

برای به دست آوردن داده ها، باید *Crawl* کنیم که به معنای دنبال کردن لینک ها یا اصطلاحا "خزیدن" در وب سایت ها است. برای استخراج داده ها از اینترنت، از کتابخانه *Beautiful Soup* استفاده می کنیم که برای استخراج داده از فایل های *html* کاربرد دارد.

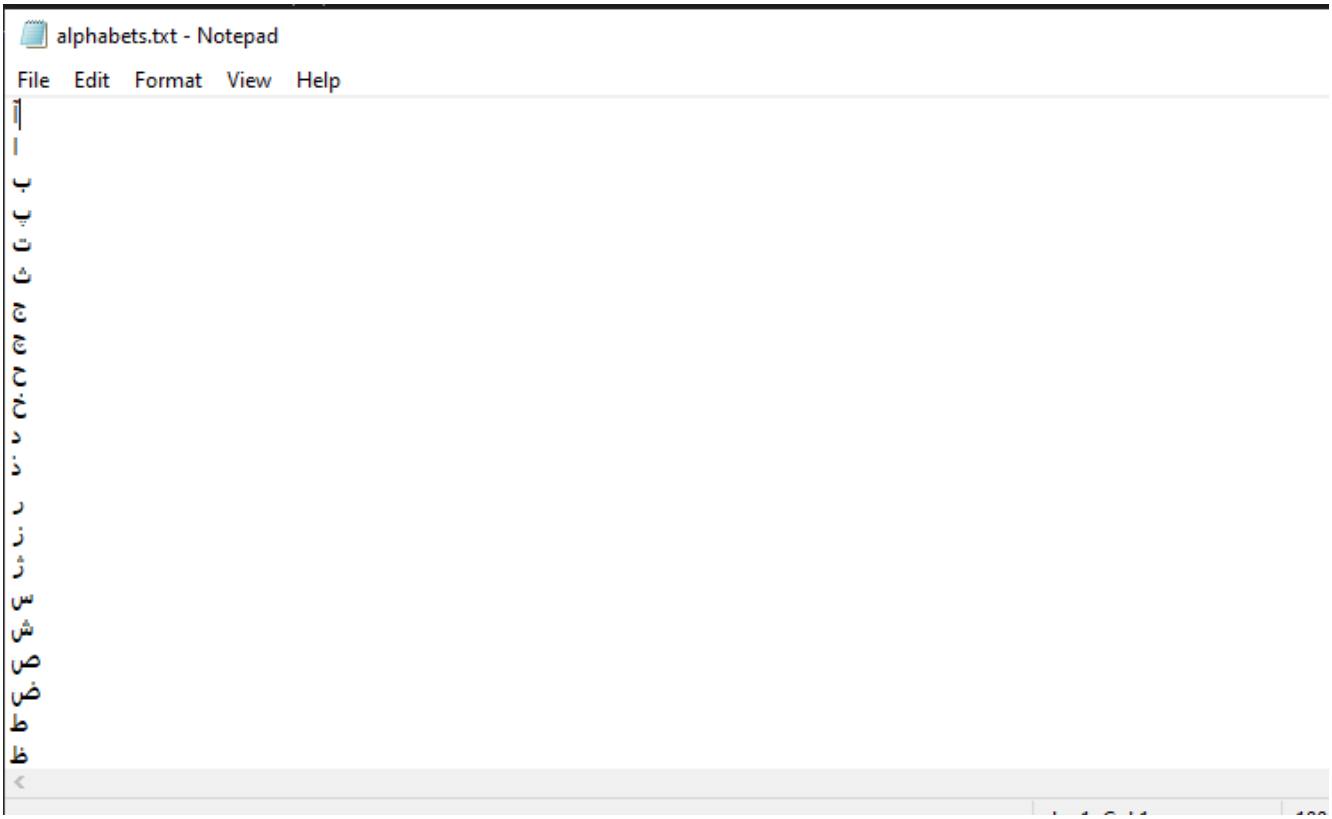
در این تحقیق از منابع سایت *Wikipedia* فارسی برای جمع آوری *data* استفاده کردیم به این صورت که ابتدا به فهرست منابع ویکی پدیا یک ریکوئست می فرستیم تا بتوانیم فهرست الفبای آن را پیدا کنیم و مطابق قطعه کد زیر ، که در فایل *collect_alphabets.py* می باشد، الفبای مورد نظر را که در ادامه قراره به لینک های مربوط به آن ها درخواست بزنیم، در یک فایل به نام *alphabets.txt* ذخیره می کنیم.

```python
from bs4 import BeautifulSoup, Tag
import requests

resultsFile = open("../data/raw/alphabets.txt", "w", encoding="utf8")
url = "https://fa.wikipedia.org/wiki/ویکی‌پدیا:فهرست_سریع"
web = requests.get(url)

soup = BeautifulSoup(web.content, "html5lib")
table = soup.find('table', attrs={'style': 'width: 80%; font-family:monospace; padding: 3px; background: #f7f8ff;
for tr in table.contents[1].contents:
    if isinstance(tr, Tag):
        tds = tr.find_all("td")
        for td in tds:
            resultsFile.write(td.text)
```

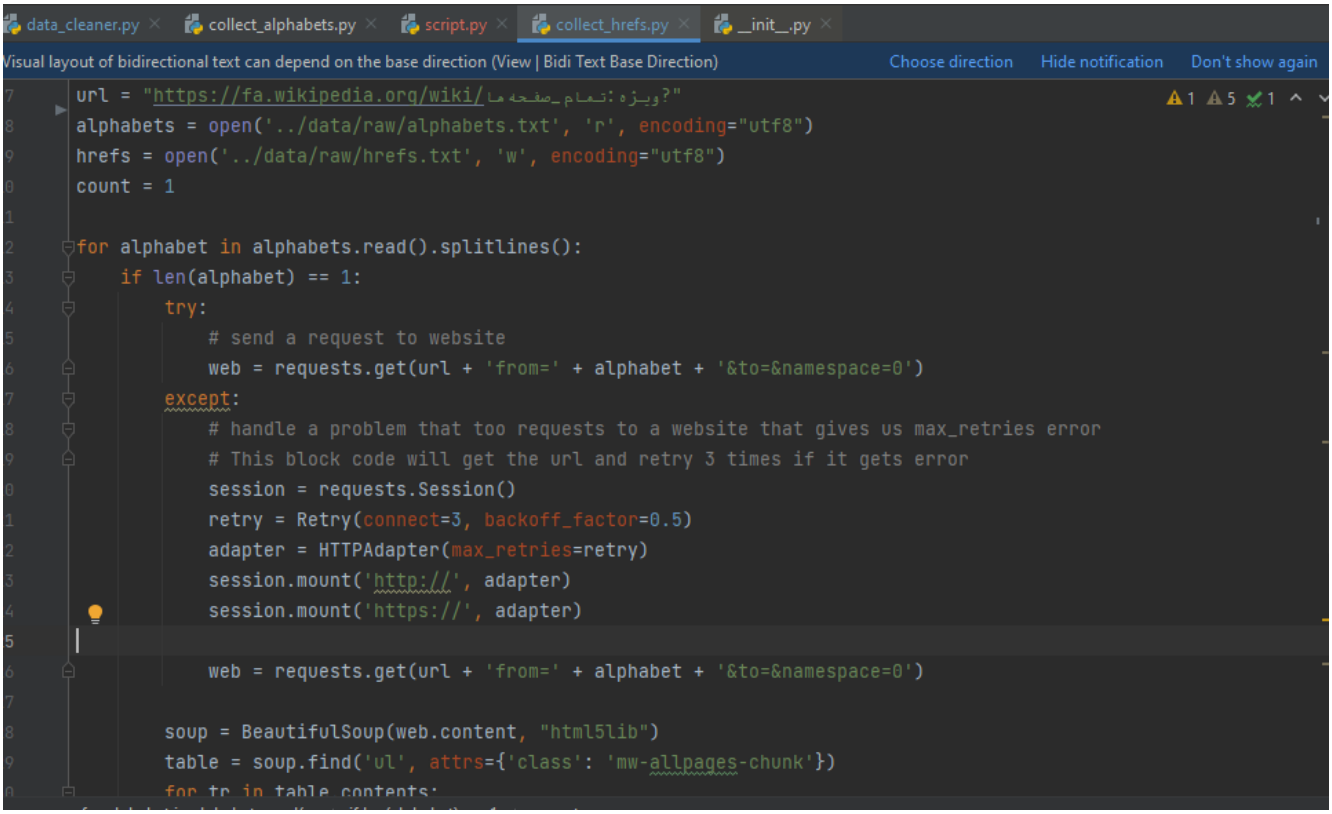
مطابق شکل بالا وقتی به *url* مربوطه درخواست می دهیم، در صفحه *html*ی که وارد آن می شویم به دنبال جدولی می گردیم که دارای استایل مشخص شده باشد و همه موارد آن که حروف الفبای مورد نظر ما می باشند را ، در فایل *alphabets.txt* می نویسیم. استایلی که برای جدول مشخص کردیم با بررسی کردن تگ های این صفحه مانند شکل زیر به دست می آید.



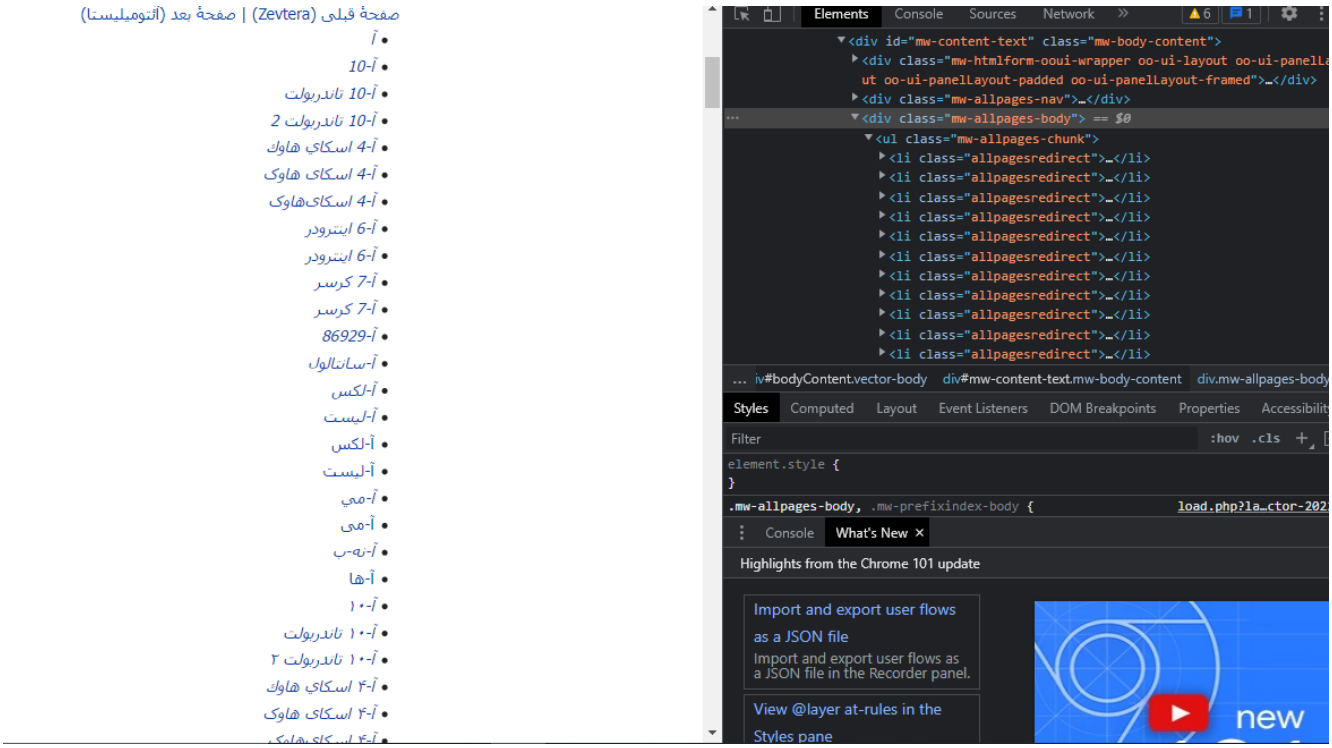
حال در پوشه *data* که پوشه مربوط به داده هایمان است، در پوشه *raw* فایل *alphabets.txt* را چک می کنیم تا از اطلاعات ثبت شده مطمئن شویم.



حال که لیست حروفی از الفبا داریم، باید به صفحه هر حرف رفته و لینک های موضوعاتی که با آن حرف شروع می شوند را به دست می آوریم. به این خاطر که داده های مورد نیاز ما حداکثر سی هزار جمله می باشد، فقط موضوعاتی که در صفحه اول خر حرف هستند را در نظر می گیریم و همچنین حروفی از فهرست که بیش از یک حرف دارند را هم حذف می کنیم.سپس مانند قطعه کد زیر که در فایل *collect_hrefs.py* می باشد ، به این صفحات یک ریکوئست می فرستیم و آدرس صفحات مرتبط با آن ها را به دست می آوریم و در فایل *hrefs.txt* می نویسیم.



مطابق کد بالا ابتدا فهرست الفبایی که از فایل آن می خوانیم را با *UTF8* اینکود می کنیم تا فرمت آن به دلیل فارسی بودن به هم نریزد.سپس به صفحه اول هر کدام از حروف الفبا در ویکیپدیا درخواست داده و با یافتن جدولی که کلاس مورد نظر ما را دارد و لیست موضوعات صفحه های ویکیپدیا در آن است ، هر کدام از این موضوعات که در اسم آن حروفی مانند خط فاصله یا پرانتز است، یا درباره موضوع ابهام زدایی می باشد را نادیده گرفته و بقیه را *decode* می کنیم و در فایل *hrefs.txt* می نویسیم.کلاسی که برای جدول مشخص کرده ایم با بررسی آن صفحه به صورت زیر به دست آمده است.

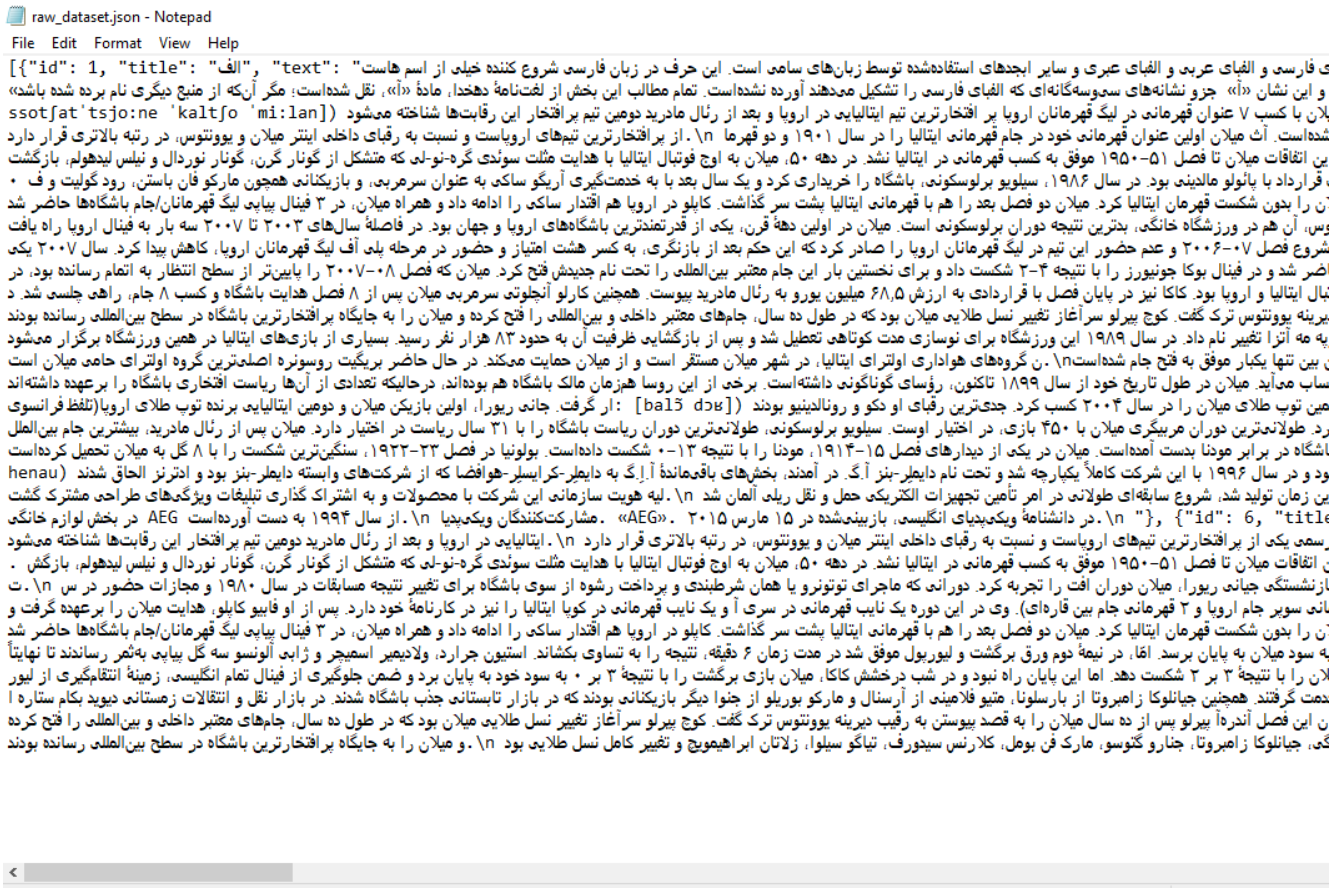


حال برای اطمینان از نتایج، قسمتی از فایل *hrefs.txt* را در زیر می بینیم:

حال که لینک هایی که می خواهیم از آن ها اطلاعات بگیریم را دخیره کردیم، باید به هر کدام جداگانه یک ریکوئست بزنیم و متنی که داخل آن نوشته شده است را استخراج کنیم و در پوشه *raw*، در فایل *raw_dataset.json* ذخیره کنیم. برای این کار با *encode* کردن لینک های ذخیره شده در فایل *hrefs.txt* در مرحله قبل، به آن ها در خواست می دهیم و با کاوش کردن در فایل *html* سایت ها همه تگ های *p* آن ها که مربوط به متن است را به دست می آوریم. مطابق شکل زیر بعضی جمله ها که در همه مقاله ها می آید را نادیده می گیریم:

```
final_text = ''
for p_tag in div.findChildren("p", recursive=False):
    p_text = ''
    for element in p_tag.contents:
        if isinstance(element, Tag):
            if element.name != 'sup' and element.name != 'br':
                if 'می‌توانید با گسترش آن به ویکی‌پدیا کمک کنید' not in element.text \
                        and element.text.replace('\n', '') != '':
                    p_text += element.text
        else:
            if 'می‌توانید با گسترش آن به ویکی‌پدیا کمک کنید' not in element \
                    and element.replace('\n', '') != '':
                p_text += element

    text = p_text.replace('\n', '')
    if p_text != '':
        final_text += p_text + ' '

dataset.append({'id': count, 'title': title, 'text': final_text})
count += 1
```

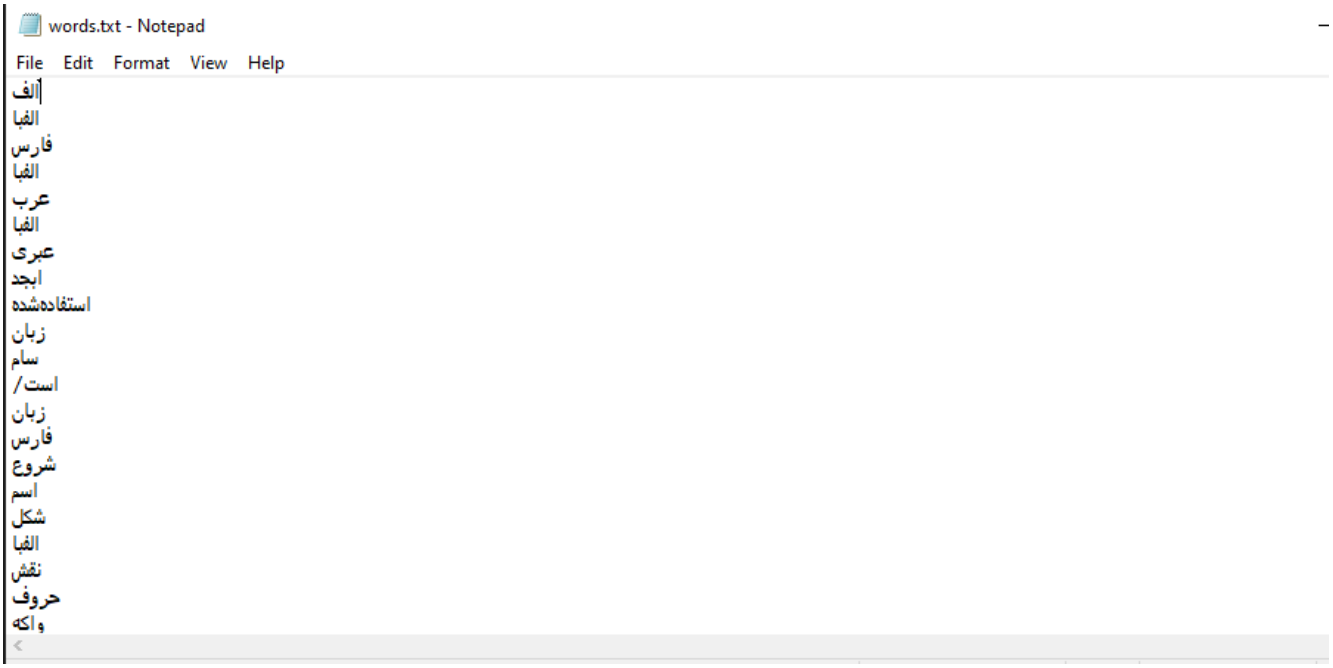
نتیجه آن را در فایل *raw_dataset.json* مشاهده می کنیم:



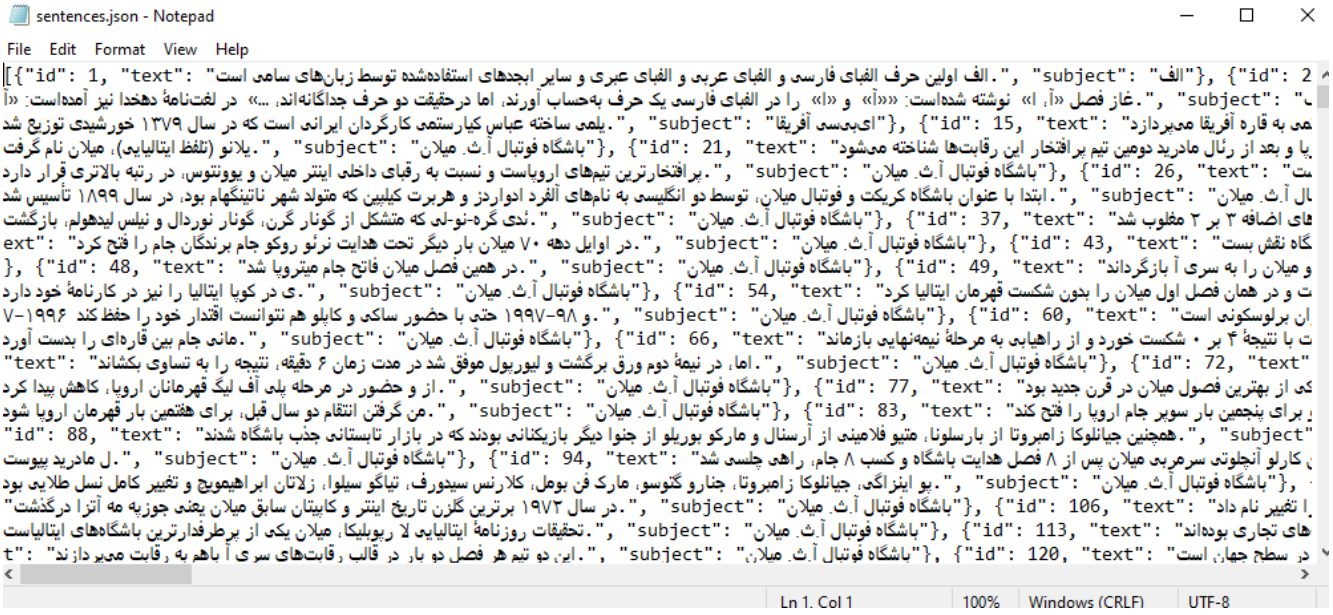
بعد از این کار زمان *normalize* کردن این دیتا می رسد.در فایل *data_cleaner.py*، از کتابخانه *Hazm* استفاده می کنیم که کتابخانه ای است تقریبا مشابه *nltk* که برای پردازش زبان فارسی کاربرد دارد و *documentation* آن از این لینک https://www.roshan-ai.ir/hazm/docs قابل مشاهده است. ابتدا با دستور *pip install hazm*، این کتابخانه را نصب می کنیم، سپس با استفاده از توابع *normalizer* و *lemmatizer*، داده هایی که در *raw_dataset.json* ذخیره کرده بودیم را نرمالایز می کنیم.

```
for row in raw_dataset:
    cleaned_text = normalizer.normalize(row.get('text'))
    cleaned_text_data.append({
        'id': row.get('id'),
        'text': cleaned_text,
        'subject': row.get('title')
    })

    for sentence in sent_tokenize(cleaned_text):
        sentences_data.append({
            'id': sentence_counter,
            'text': sentence,
            'subject': row.get('title')
        })
        sentence_counter += 1
    for word in word_tokenize(cleaned_text):
        new_word = lemmatizer.lemmatize(word)
        if new_word not in stop_words:
            if '#' in new_word:
                new_word = new_word.replace('#','/')
            if new_word in counts:
                counts[new_word] += 1
            else:
                counts[new_word] = 1
```

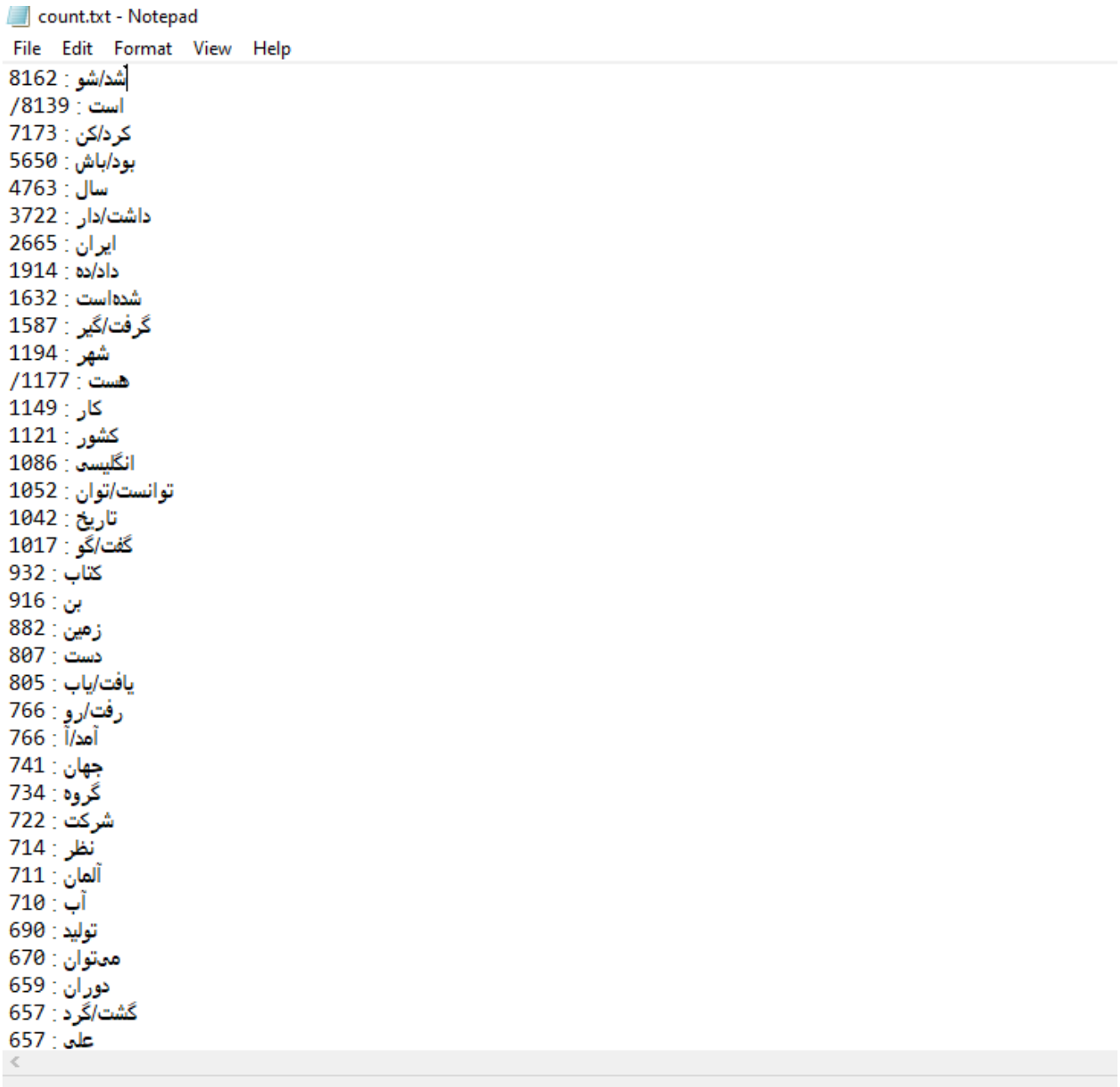
سپس با *stemming* ریشه اصلی آن ها را به دست می آوریم و در نهایت چک می کنیم که این کلمات بین *stop words* های زبان فارسی هستند یا خیر و اگر در فایل *stop words* نبود، آن را در لیست کلمات در فایل *words.txt* می نویسیم که در شکل زیر این فایل را می بینیم.



لیست *stop words* ها در پوشه *raw* در فایل *stopwords-fa.txt* قرار دارد.سپس هر جمله که توسط این توابع جدا شده اند را در *sentences.json*، همراه با آیدی و موضوع آن می نویسیم.فرمت ذخیره شده را در شکل زیر می بینیم.



همچنین برای این که هر کلمه را با تعداد تکرارش داشته باشیم، یک دیکشنری تعریف می کنیم و هر بار که کلمه ای تکرار شد یکی به تعداد تکرار آن اضافه می کنیم و در نهایت آن را پس از *sort* کردن بر اساس تکرار بیشتر، در فایل *count.txt* ذخیره می کنیم.



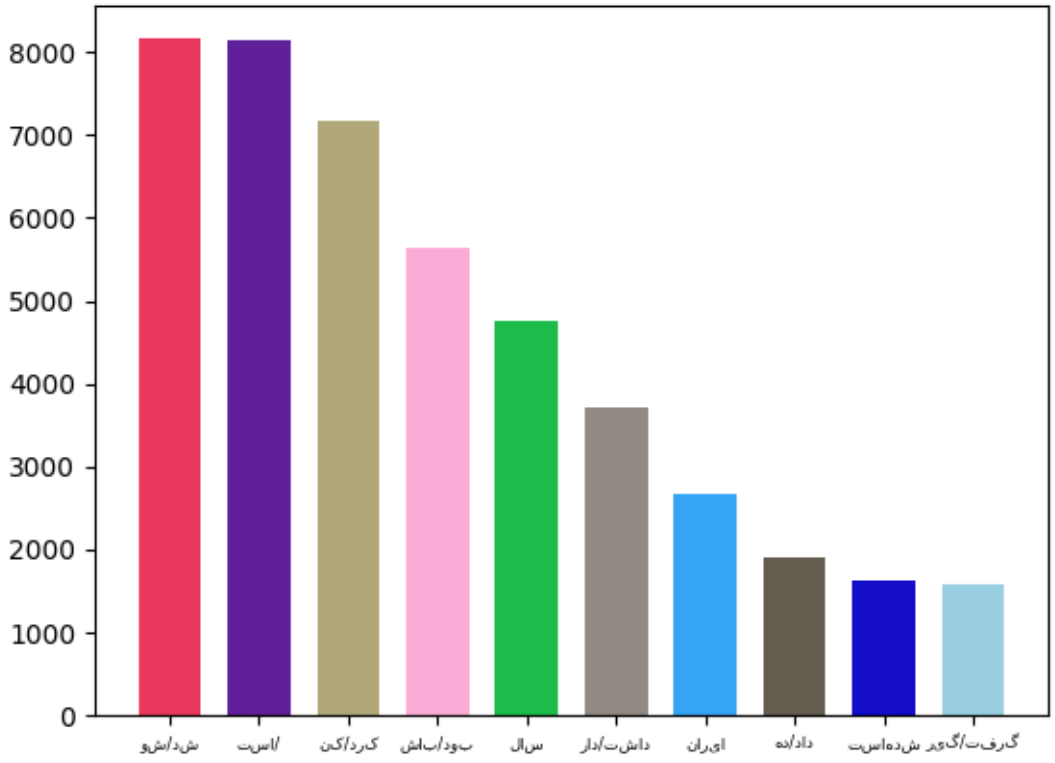
برای این که براساس تعداد مرتب کنیم، از قطعه کد زیر استفاده می کنیم:

```
words_count = open('../data/word_count/count.txt', 'w', encoding='utf8')

for i in sorted(counts, key=counts.get, reverse=True):
    words_count.write(i + " : "+str(counts[i])+"\n")

cleaned_dataset_file.write(json.dumps(cleaned_text_data, ensure_ascii=False))
sentence_broken_file.write(json.dumps(sentences_data, ensure_ascii=False))
```

حال که کلماتی که بیشترین تکرار را دارند پیدا کرده ایم، ده تا از آن ها که بیشترین تکرار را داشته اند در یک نمودار نمایش می دهیم.در فایل *plot.py* ، ما با خواندن فایل *count.txt* و انتخاب ده مورد اول آن نمودار کلمات و تعداد تکرار آن ها را با رنگ های *random* رسم می کنیم و عکس نمودار را در پوشه *plot_img* ذخیره می کنیم.

```
from __future__ import unicode_literals

import matplotlib.pyplot as plt
import numpy as np
import random
from bidi.algorithm import get_display
wordcount = open('../data/word_count/count.txt', 'r', encoding='utf8').read().splitlines()
x = []
y = []
for i in range(10):
    row = wordcount[i].split(' : ')
    y.append(int(row[1]))
    x.append(get_display(row[0]))

rnd = []
for i in range(10):
    r = random.random()
    b = random.random()
    g = random.random()
    rnd.append((r, g, b))
plt.bar(x, y, width=0.7, bottom=None, align='center', data=None, color = rnd )
plt.xticks(fontsize=6)

plt.savefig('../data/plot_img/plot.png')
```

در نهایت در شکل زیر عکس نمودار را مشاهده می کنیم.



در انتها فایل *script.py*، همه توابع موجود برای گرفتن داده و نرمالایز کردن را انجام می دهد که کد آن در زیر نشان داده شده است.

```python
import data_cleaner
import collect_alphabets
import collect_hrefs
import collect_articles

if __name__ == "__main__":
    collect_alphabets.func()
    collect_hrefs.func()
    collect_articles.func()
    data_cleaner.func()
```