دانشگاه یزد

دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی مهندسی کامپیوتر

گرایش فناوری اطلاعات

تحلیل احساسات کاربران توییتر درباره رئیس جمهور ایالات متحده آمریکا در طول سال ۲۰۲۱ میلادی

نگارش:

پوریا بحری

استاد راهنما :

دکتر سید علیرضا صدرالسادات

بهمن ۱۴۰۰


بسم الله الرحمن الرحيم

تقدیم

تقدیم به پدر و مادر عزیزم
به پاس عاطفه سرشار و گرمای امیدبخش وجودتان
که در این سردترین روزگاران بهترین پشتیبان است
و به پاس محبت های بی دریغتان
که هرگز فروکش نمی کند
این مجموعه را به شما تقدیم می کنم

تشکر و قدردانی

تشکر قلبی و لسانی خود را از استاد عالی قدر جناب آقای دکتر صدرالسادات که زحمت راهنمایی این پایان نامه را عهده‌دار گردیدند و در تمامی مراحل انجام رساله از راهنمایی های مدبّرانه ایشان استفاده نمودم ابراز می دارم و توفیقات روز افزون ایشان را توأم با صحت و سعادت خواستارم.

چکیده

همزمان با پیروزی جو بایدن در انتخابات ریاست جمهوری ایالات متحده آمریکا در پایان سال ۲۰۲۰ میلادی ، دو دستگی شدیدی میان مردم پدید آمد که نمود آن در شبکه‌های اجتماعی قابل دیدن بود. اختلاف نظرها در سال ۲۰۲۱ نیز با اتفاقاتی همچون خروج نیروهای آمریکایی از افغانستان، ادامه‌ی شیوع ویروس کرونا، بحران مهاجران و... ادامه یافت. توییتر یکی از پر مخاطب‌ترین شبکه‌های اجتماعی در دنیا است. محبوبیت توییتر در کنار محتوای عمدتا متنیِ آن، باعث شده که مرجع خوبی برای تحلیل احساسات مردم درباره موضوعات مختلف باشد. در این پروژه با جمع‌آوری توییت‌های کاربران درباره جو بایدن (رئیس جمهور آمریکا) و تحلیل آن‌ها با استفاده از روش‌های پردازش زبان طبیعی، میزان محبوبیت وی را در طول سال ۲۰۲۱ بررسی می‌کنیم.

کلیدواژه : پایتون، پردازش زبان طبیعی، تحلیل احساسات، توییتر، بایدن

فهرست مطالب

فهرست شکل‌ها

۱ مقدمه

۱-۱ پیشگفتار

همزمان با پیروزی جو بایدن[۱] در پایان سال ۲۰۲۰ میلادی در انتخابات ریاست جمهوری ایالات متحده آمریکا، دو دستگی شدیدی میان مردم پدید آمد که نمود آن در شبکه‌های اجتماعی قابل دیدن بود. اختلاف نظرها در سال ۲۰۲۱ نیز با اتفاقاتی همچون خروج نیروهای آمریکایی از افغانستان، ادامه‌ی شیوع ویروس کرونا، بحران مهاجران و... ادامه یافت.

توییتر[۲] یکی از محبوب‌ترین شبکه‌های اجتماعی در دنیا است که این امکان را به کاربران می‌دهد تا صحبت‌های خود را در حداکثر ۲۸۰ کلمه بنویسند. محبوبیت توییتر در کنار محتوای عمدتا متنی و نسبتا کوتاه آن، باعث شده که مرجع خوبی برای تحلیل احساسات مردم درباره موضوعات مختلف باشد.[۱] تحلیل احساسات[۳] مبحثی در حوزه پردازش زبان طبیعی[۴] بوده که با اعمال دوقطبی (مانند خوب / بد) روی کلمات، سعی در طبقه‌بندی داده‌های موجود می‌کند. به عنوان مثال برای بررسی نظرات مردم درباره یک محصول، میتوان کامنت‌های ثبت شده درباره آن را مورد تحلیل قرار داد.

به‌طور کلی انجام این پروژه در سه بخش انجام شد. جمع‌آوری توییت‌ها، پیش‌پردازش[۵] متون و تحلیل احساسات و در نهایت مصورسازی و بررسی نتایج. در بخش اول با کتابخانه[۶] snscrape و اعمال کوئری[۷]های مناسب، دیتاهای مورد نیاز جمع‌آوری شد. در بخش دوم با استفاده از کتابخانه‌های re و spacy دیتاها منظم و سپس به کمک کتابخانه vaderSentiment تحلیل احساس انجام شد. در نهایت نیز با استفاده از کتابخانه matplotlib نتایج بدست آمده را به‌صورت بصری در چند نمودار نشان داده و بررسی می‌کنیم.

---

[۱] Joe Biden
[۲] Twitter
[۳] Sentiment analysis
[۴] Natural language processing
[۵] Preprocessing
[۶] Library
[۷] Query

۲-۱ هدف از انجام این پروژه

بدیهی است که در مقاطعی از سال، انتقاداتی به سیاست‌های جو بایدن وارد شد. از جمله پس از خروج آشفته از افغانستان و یا پس از ورود مهاجران پرشمار از کشورهای آمریکای مرکزی. بطور کلی هدفی که این پروژه دنبال می‌کند، بررسی نظرات افراد در فضای مجازی (شبکه اجتماعی توییتر) نسبت به رییس‌جمهور آمریکا و بررسی تغییرات مثبت و منفی آن‌ها در طول سال ۲۰۲۱ میلادی است. همچنین در ادامه سعی در تطبیق نتایج به‌دست آمده با مشاهدات دنیای واقعی خواهیم داشت و بررسی می‌کنیم که آیا شبکه‌های اجتماعی می‌توانند وضعیت جامعه را نشان دهند یا خیر؟

۲ فصل ۲ – مطالعات صورت گرفته و داده‌های مورد نیاز

۲-۱ مقدمه

پیش‌نیاز اصلی انجام پروژه‌ی تحلیل احساسات، دانستن اصول جبر خطی، هوش مصنوعی[۸] و همچنین روش‌های رایج در پردازش زبان طبیعی می‌باشد. بدون شک امکان استفاده از مدل‌ها و روش‌های از پیش پیاده‌سازی شده، کاملا میسر است. اما دانستن نحوه پیاده‌سازی هرکدام از مدل‌ها از این جهت اهمیت پیدا می‌کند که در مواقعی نیاز است که آن مدل را با توجه به نیازهای پروژه، شخصی‌سازی و بهینه‌سازی کنیم. همچنین برای انتخاب بهترین مدل، نیاز است تا دانش کلی از نحوه پیاده‌سازی هرکدام از گزینه‌های موجود داشته باشیم.

۲-۲ مطالعات صورت گرفته

همانطور که پیشتر نیز بیان شد، دانستن اصول و روش‌های رایج در علم پردازش زبان طبیعی، پیشنیاز اصلی برای انجام این پروژه بود. برای حل این مسئله علاوه بر دانش پیشین از مدل‌های یادگیری ماشین با نظارت[۹] ، نیاز بود تا دانشی در حوزه پردازش زبان طبیعی نیز حاصل شود. برای کسب این دانش در مقاطع ابتدایی از فیلم‌های آموزشی یوتوب و در ادامه از کتاب‌های برتر در این حوزه و دیگر منابع متنی استفاده شد. همچنین وبسایت kaggle برای بررسی پروژه‌های افراد فعال و باتجربه در حوزه علوم داده[۱۰] و پردازش زبان طبیعی نیز مورد استفاده قرار گرفت.

۲-۳ داده‌های مورد نیاز

هدف کلی پروژه بررسی نظرات کاربران فضای مجازی درباره جو بایدن بود. از آنجایی که شبکه اجتماعی توییتر برای اینکار انتخاب شد، نیاز بود تا به حجم زیادی از توییت‌های کاربران این شبکه اجتماعی درباره رییس جمهور ایالات متحده آمریکا دسترسی داشته باشیم. برای اینکار از کتابخانه snscrape استفاده شد. این کتابخانه متن‌باز[۱۱] برای اسکرپ کردن[۱۲] وبسایت‌های زیادی مناسب است. تعریف کلی وب اسکرپینگ، استخراج داده از وبسایت می‌باشد. مزیت اصلی این کتابخانه، امکان دسترسی به توییت‌های قدیمی بود که در بخش‌های بعدی بیشتر به آن خواهیم پرداخت.

---

[۸] Artificial intelligence

[۹] Supervised machine learning

[۱۰] Data science

[۱۱] Open source

[۱۲] Scrape

۳ فصل ۳ – معماری برنامه و کتابخانه‌های مورد استفاده

۳-۱ مقدمه

در این فصل به روش کلی پیاده‌سازی این پروژه، بخش بندی‌های مورد نیاز برای اینکار و معرفی محیط توسعه پرداخته می‌شود. سپس چند مورد از کتابخانه‌های مهم مورد استفاده در این پروژه معرفی میگردند و دلایل برتری‌شان نسبت به دیگر گزینه‌های موجود بررسی می‌شود.

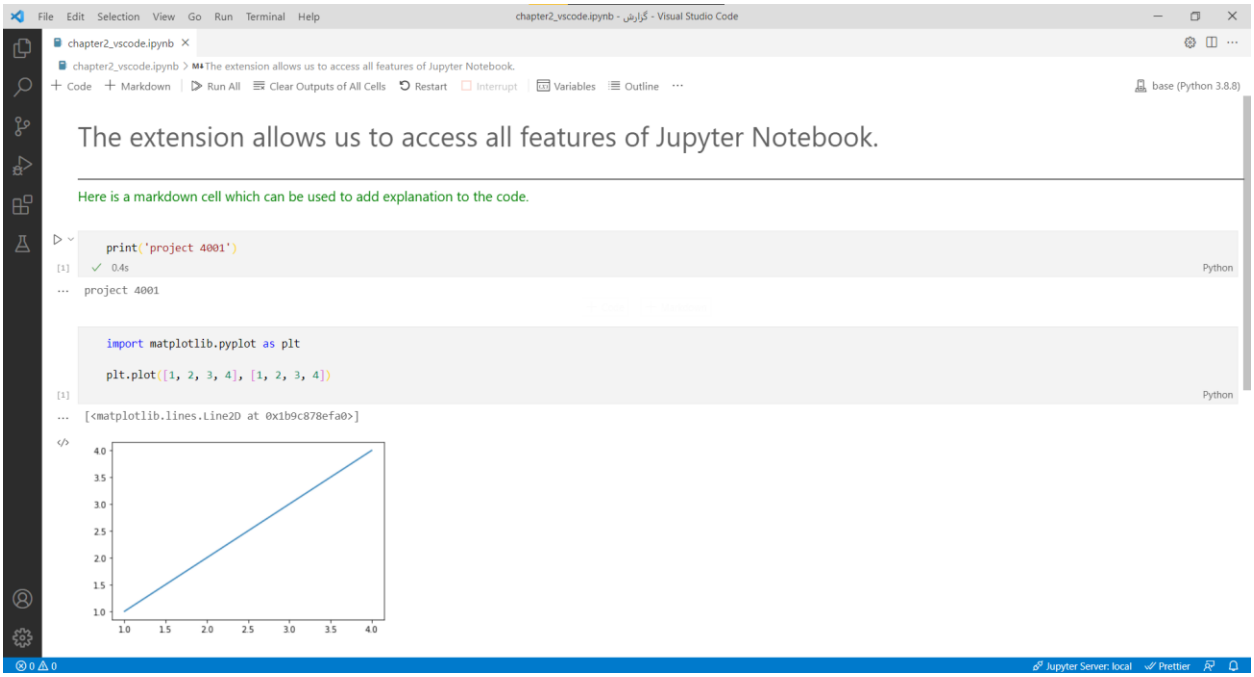
۳-۲ معماری برنامه

پیاده‌سازی این پروژه بطور کلی در ۳ بخش تعریف و انجام شد:

۱. جمع‌آوری توییت‌های هدف با اعمال فیلترهای مناسب
۲. پیش پردازش متن و تحلیل احساسات
۳. نتیجه‌گیری از تحلیل و مصورسازی نتایج بدست آمده

برای پیاده‌سازی این پروژه از محیط visual studio code بر مبنای Jupyter استفاده شد. از آنجایی که این نرم‌افزار هنگام نوشتن کد، با هوش مصنوعی (intellisense) خود پیشنهادهای مناسبی در زمان کوتاه می‌دهد، می‌تواند سرعت کد نویسی و پیاده‌سازی را به میزان قابل توجهی افزایش دهد. می‌توان گفت یکی از نقاط ضعف استفاده از jupyter notebook در محیط‌های مرورگر مثل گوگل کروم[۱۳]، سرعت بسیار پایین در نمایش پیشنهاد است که با استفاده از visual studio code این مشکل کاملا مرتفع می‌شود. برای انجام این‌کار کافی است افزونه Jupyter را در بخش extension های نرم‌افزار visual studio code دانلود و نصب نمود. این افزونه توسط شرکت Microsoft توسعه داده شده و با نصب آن، امکان استفاده از محیط jupyter notebook و فایل‌های با پسوند ipynb. را خواهیم داشت.

[۱۳] Google Chrome



شکل ۱-۳- اجرای Jupyter در محیط visual studio code

در شکل ۱-۳ ، نمایی از نرم‌افزار visual studio code را مشاهده می‌کنیم. مزیت اساسی استفاده از jupyter ، اجرای قسمت‌های مختلف کد بطور مجزا و بدون از دست دادن نتایج قسمت‌های دیگر می‌باشد. در این صورت میتوانیم بدون نیاز به شروع مجدد برنامه و از بین رفتن مقادیر متغییرهایمان، کد جدیدی را به راحتی اضافه کرده و یا قطعه کدی را ویرایش کنیم. همانطور که مشاهده می‌شود این برنامه از سه cell تشکیل شده‌است که میتوان هرکدام را بدون وجود تداخل با بقیه، ویرایش کرد. اولین cell به شکل markdown می‌باشد که به شکل گسترده در بسیاری از پروژه‌های ساخته‌شده توسط Jupyter به عنوان توضیحات اضافه برای کد، بکار می‌رود. این ویژگی در کنار بالا بردن خوانایی کد، به مرتب سازی کامنت‌های[۱۴] برنامه نیز کمک زیادی می‌کند.

هسته (کرنل) مورد استفاده برای انجام پروژه، Anaconda۳ بود. Anaconda یک توزیع از پایتون[۱۵] می‌باشد که با تمرکز روی علوم داده، یادگیری ماشین، پردازش داده‌های پرحجم[۱۶] و... سعی در ساده‌سازی مدیریت پکیج[۱۷] و پیاده‌سازیِ آسان آن‌ها می‌کند.

[۱۴] Comment
[۱۵] Python
[۱۶] Large-scale data processing
[۱۷] Package management

۳-۳ کتابخانه‌های مورد استفاده در پروژه

در پیاده‌سازی این پروژه چندین کتابخانه در بخش‌های مختلف انجام کار، مورد استفاده قرار گرفتند. در این بخش به معرفی این کتابخانه‌ها و کاربرد آنها در طول انجام پروژه می‌پردازیم و دلایل انتخاب آن‌ها را بررسی می‌کنیم.

۳-۳-۱ کتابخانه snscrape

کتابخانه snscrape کتابخانه‌ای متن‌باز به زبان برنامه‌نویسی پایتون، برای اسکرپ کردن شبکه‌های اجتماعی محبوبی همچون instagram ، reddit ، twitter و... می‌باشد. یکی از مزیت‌های اساسی این کتابخانه، امکان استخراج توییت‌ها، بدون داشتن اکانت توسعه‌دهنده[۱۸] در سایت توییتر است.[۲] بطور مثال کتابخانه دیگری که بطور گسترده برای استخراج از توییتر بکار میرود، کتابخانه tweepy است که امکان دسترسی به حساب کاربری توسعه دهنده را از طریق API توییتر برای افراد فراهم می‌کند که میتوان با آن فرایندهایی مثل ارسال توییت، لایک کردن، اسکرپ کردن و... را از طریق کد نویسی انجام داد.[۳] اما مشکل بزرگی که tweepy و دیگر کتابخانه‌های مبتنی بر API توییتر دارند، وجود محدودیت‌های API می‌باشد.[۴] این محدودیت‌ها عبارتند از:

- لزوم ثبت‌نام و ارسال درخواست ایجاد developer account : در روش‌های مبتنی بر API ، نیاز است تا درخواست متنی را برای قسمت پشتیبانیِ توییتر ارسال کرده و آن‌ها با تایید درخواست ما، امکان دسترسی به سطح اول developer account را فراهم کنند. این فرایند بسیار زمانبر بوده و حتی در مواردی تایید نمی‌شود.
- عدم امکان استخراج توییت‌های قدیمی: در صورت استفاده از API توییتر، حداکثر امکان دسترسی به توییت‌های ۳۰ روز پیش فراهم است و توییت‌های قدیمی‌تر در دسترس نیست.

پس از بررسی محدودیت‌های API توییتر، تصمیم بر آن شد که از snscrape برای استخراج توییت‌ها استفاده شود. هرچند که با استفاده از آن نمی‌توان حساب شخصی را مانند API مدیریت کرد و امکان نوشتن دستور برای عملیات‌هایی مثل لایک کردن، دنبال کردن و... وجود ندارد، اما استخراج توییت‌ها با استفاده از snscrape در همه زمان‌ها بدون محدودیت و بدون نیاز به ثبت هیچ درخواستی فراهم است. علاوه بر این، با استفاده از این کتابخانه، می‌توان به توییت‌های قدیمی نیز دسترسی پیدا کرد که اساس کار این پروژه می‌باشد.

[۱۸] Developer account

۲-۳-۳ کتابخانه spaCy

کتابخانه spaCy یک کتابخانه رایگان و متن باز قدرتمند برای پردازش زبان طبیعی است. این کتابخانه از آخرین تحقیقات به عمل آمده در حوزه NLP برای پیاده‌سازی الگوریتم‌های خود استفاده می‌کند و این ویژگی‌ها استفاده از آن را در حوزه پژوهش و همچنین در صنعت بسیار گسترده کرده است. مدل‌های پرکاربرد حوزه NLP همگی در کتابخانه spaCy وجود داشته و قابل استفاده می‌باشند. توجه توسعه‌دهندگان این کتابخانه به بالا بودن سرعت پردازش و استفاده از آخرین مدل‌های موجود شبکه‌های عصبی[۱۹] برای پیاده‌سازی مدل‌هایی همچون شناسایی موجودیت‌های نام‌دار[۲۰]، طبقه‌بندی متن[۲۱]، برچسب‌گذاری جزء کلام[۲۲] و دیگر مدل‌های محبوب حوزه NLP ، یکی از مهمترین دلایل عملکرد خوب این کتابخانه برای پردازش زبان طبیعی می‌باشد. این کتابخانه در بخش پیش پردازش متن (توییت) این پروژه مورد استفاده قرار گرفت. عملیات‌های lemmatization و حذف کلمات توقف[۲۳]، توسط این کتابخانه انجام شد که در بخش‌های بعد به تفصیل به آن‌ها خواهیم پرداخت.

۳-۳-۳ کتابخانه vaderSentiment

VADER پروژه‌ای متن‌باز و از پیش تعلیم دیده شده و ابزاری برای تحلیل احساسات بوده که به‌طور مشخص برای تحلیل احساسات متون شبکه‌های اجتماعی بهینه‌سازی شده است.[۵] این کتابخانه به علت قدرت بالای خود در تحلیل احساسات، در کتابخانه‌ی محبوب NLTK نیز در دسترس بوده و به راحتی قابل پیاده‌سازی است. یکی از نقاط قوت این کتابخانه، پیاده‌سازی تحلیل احساسات برای شکلک‌ها (ایموجی[۲۴]) می‌باشد که با توجه به استفاده‌ی گسترده‌ی آنها در متون شبکه‌های اجتماعی، برتری بزرگی برای این کتابخانه نسبت به دیگر گزینه‌های موجود است. شکل ۲-۳ ، نمونه‌ای از خروجی این کتابخانه برای متون مختلف است که در فصل‌های بعد به آن بیشتر خواهیم پرداخت.

[۱۹] Neural networks
[۲۰] Named-entity recognition
[۲۱] Text classification
[۲۲] Parts of speech
[۲۳] Stop words
[۲۴] Emoji

```python
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()

sample_texts = [
    "I'm feeling blessed 🙂" ,
    'The food was not bad' ,
    'The exam was very hard :-('
]

for text in sample_texts:
    print(sid.polarity_scores(text))
```

✓ 0.6s

```
{'neg': 0.0, 'neu': 0.27, 'pos': 0.73, 'compound': 0.7968}
{'neg': 0.0, 'neu': 0.584, 'pos': 0.416, 'compound': 0.431}
{'neg': 0.528, 'neu': 0.472, 'pos': 0.0, 'compound': -0.5379}
```

شکل ۲-۳ – نمونه‌ای از عملکرد کتابخانه vader

۳-۳-٤ کتابخانه pandas

این کتابخانه امکان ویرایش و طبقه‌بندی داده‌ها را به شکل سریع و بسیار انعطاف‌پذیر فراهم می‌کند. کتابخانه pandas یکی از اساسی‌ترین اجزاء این پروژه برای ذخیره داده‌های استخراج شده و اعمال عملیات‌های موردنظر بر روی ستون‌های مختلف آن بود. امکان اعمال توابع روی داده‌ها به آسانی توسط این کتابخانه فراهم است. با استفاده از این کتابخانه میتوان از نوع DataFrame استفاده کرد که هم از لحاظ بصری خواناتر است و هم سرعت محاسبات بالاتری نسبت به نوع list دارد. همچنین ایجاد، ویرایش و حذف سطر و ستون‌های داده نیز به راحتی توسط این کتابخانه قابل انجام می‌باشد. از دیگر مزیت‌های این کتابخانه، امکان اجرای کوئری‌های SQL بر روی داده‌ها است که کار را برای افرادی که از پیش با پایگاه‌های داده[۲۵] SQL کار کرده‌اند، بسیار ساده می‌کند.

[۲۵] Database

Here is an example to compare list and DataFrame

DataFrame is visually more Comprehensible

```
data = [['Ali', 6000000], ['Ahmad', 15000000], ['Zahra', 14000000]]

data
```

✓ 0.5s

```
[['Ali', 6000000], ['Ahmad', 15000000], ['Zahra', 14000000]]
```

```
import pandas as pd

df = pd.DataFrame(data, columns = ['Name', 'Salary'])

df
```

✓ 0.7s

	Name	Salary
0	Ali	6000000
1	Ahmad	15000000
2	Zahra	14000000

شکل ۳-۳ – مقایسه نوع list و نوع DataFrame

۳-۳-۵ کتابخانه matplotlib

کتابخانه matplotlib را می‌توان اساس مصورسازی دیتا[۲۶] در حوزه علوم داده در نظر گرفت. به‌طور کلی matplotlib پکیج جامعی برای ساخت مصورسازی‌های آماری، پویا و قابل تعامل می‌باشد. این کتابخانه بارها توسط دیگر افراد فعال در این حوزه fork شده‌است و پکیج‌های بسیاری نیز بر پایه آن تهیه شده‌اند که از جمله آنان میتوان به پکیج محبوب seaborn اشاره داشت. با استفاده از کتابخانه matplotlib می‌توان به راحتی ورودی‌ها را در گراف‌های[۲۷] دلخواه مصور کرد و درک بیشتری نسبت به نتایج بدست آورد. از جمله این گراف‌ها میتوان به نمودار میله‌ای، نمودار دایره‌ای و هیستوگرام[۲۸] اشاره داشت.

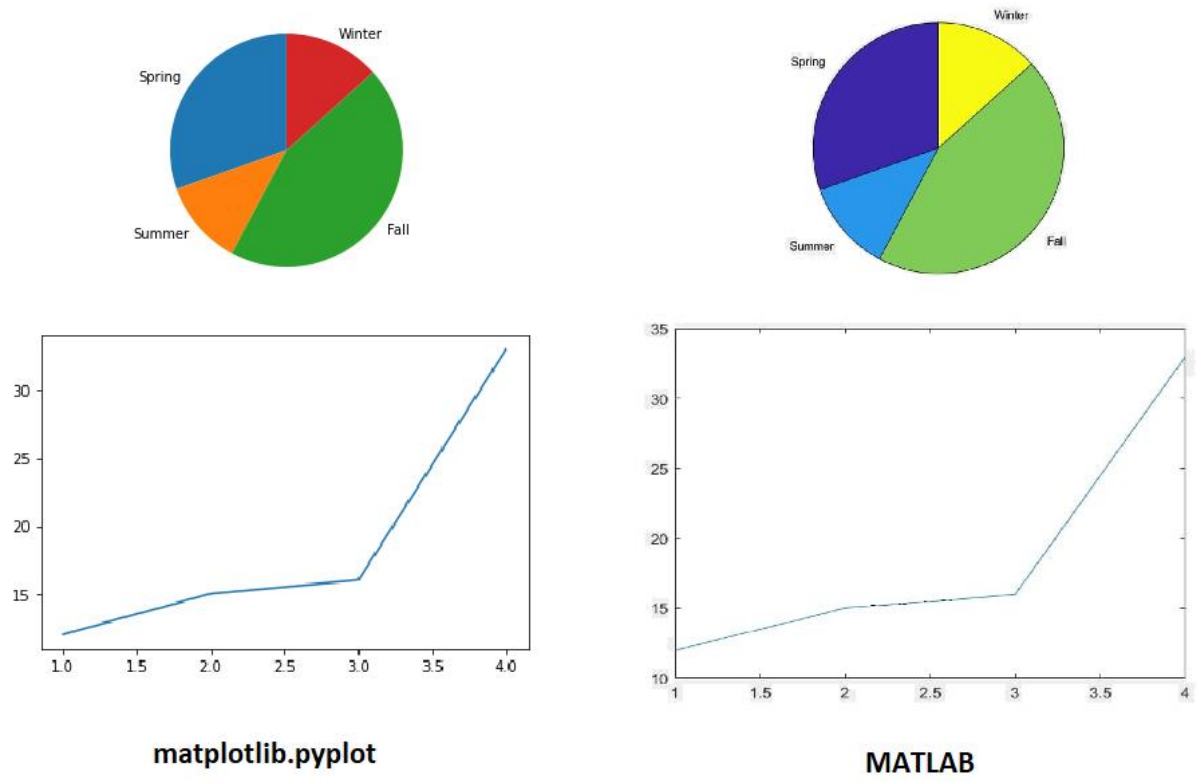
یکی از قابل تعامل‌ترین اشکال نمودار میان پکیج‌های موجود، نمودارهای ساخته شده در برنامه MATLAB می‌باشد که استانداردهای نرم‌افزارهای کامپیوتری را برای مصورسازی بسیار بالا برد. توسعه‌دهندگان کتابخانه matplotlib با پیاده‌سازی پکیج pyplot که خروجی بسیار مشابهی با MATLAB دارد، قدرت این کتابخانه را هرچه بیشتر بالا برده و امکان استفاده از گراف‌های با کیفیت را به کاربران میدهند که یکی از نقات قوت اساسی این کتابخانه است.

---

[۲۶] Data visualization

[۲۷] Graph

[۲۸] Histogram



شکل ۴-۳ – مقایسه pyplot و MATLAB

۳-۳-۶ کتابخانه wordcloud

این کتابخانه که بر مبنای کتابخانه‌های pillow و numpy پیاده‌سازی شده‌است، در بخش سوم پروژه برای مصورسازی نتایجِ پردازش شده، مورد استفاده قرار گرفت. اساس کار این کتابخانه، مصورسازی کلمات پرتکرار و تعیین اندازه فونت هرکلمه بر اساس تعداد تکرارهایش است. بدیهی است هرچقدر تعداد بیشتر باشد، اندازه قلم مورد استفاده برای مصورسازی آن کلمه نیز بزرگتر خواهد بود.

٤ فصل ۴ – الگوریتم برنامه و توضیح کدهای اجرایی

٤-١ مقدمه

همانطور که پیشتر نیز بیان شد، پیاده‌سازی این پروژه در سه بخش انجام شد. در این فصل به‌طور مفصل کدهای دو بخش ابتدایی برنامه توضیح داده می‌شوند و خروجی‌های هر قسمت را بررسی می‌کنیم. در بخش اول، نحوه اسکرپ کردن توییت‌ها و پیاده‌سازی فیلترهای مورد نظر و دلایل اعمال هرکدام از آن‌ها توضیح داده می‌شوند. در بخش دوم، روش پیش‌پردازش متون دریافتی شرح داده شده و سپس تحلیل احساسات را روی آن‌ها انجام می‌دهیم.

٤-٢ استخراج توییت‌ها

همانطور که بیان شد برای استخراج از توییتر، از کتابخانه snscrape استفاده میکنیم که در بخش قبل به معرفی آن پرداختیم. در ابتدا نیاز است رویکرد انتخاب شده برای انجام پروژه شرح داده شود. روشی که به‌طور معمول برای دریافت و پردازش داده استفاده می‌شود، ابتدا دریافت اطلاعات و سپس حذف و تصفیه اطلاعات غیرقابل استفاده می‌باشد. اما از آنجایی که با این‌کار تعداد داده‌ها کم می‌شود، تصمیم بر آن شد که از همان ابتدا فقط داده‌های هدف دریافت شوند تا کیفیت داده‌ها و در نتیجه کیفیت نتایج بالاتر برود. به همین دلیل نیاز بود تا برای استخراج توییت‌ها، تابعی تعریف شود.

```
def get_tweets(max_tweet : int, start_date : str, end_date : str):

    tweets_list = []
    user_IDs = []

    query = 'biden -trump since:{} until:{} lang:en -filter:retweets'.format(start_date, end_date)

    accepted_tweets = 0
    for tweet in sntwitter.TwitterSearchScraper(query).get_items():

        if accepted_tweets >= max_tweet :
            tweets_df = pd.DataFrame(tweets_list, columns=['Datetime', 'Tweet Id', 'Text', 'Username'])
            return tweets_df

        if tweet.user.id not in user_IDs and popularity(tweet) > 20 :
            tweets_list.append([tweet.date, tweet.id, tweet.content, tweet.user.username])
            accepted_tweets += 1
            user_IDs.append(tweet.user.id)
            scrape_report()
```

شکل ۱-۴ - تابع تعریف‌شده برای استخراج توییت‌ها

روند کلی کار بدین صورت است که ابتدا باید کوئری مناسب برای تابع TwitterSearchScraper ساخته شود که در ادامه روند ساخت آن شرح داده خواهد شد. این تابع تا جایی که تعداد توییت‌های استخراج شده‌ی مورد قبول به مقدار مورد نظر نرسیده باشد، به کار خود ادامه می‌دهد. تنها شرطی که باعث متوقف شدن این تابع می‌شود، رسیدن به تعداد توییتِ موردنظر خواهد بود. در آن صورت، تمام توییت‌هایی که استخراج شده‌اند و اکنون در لیستی به نام tweets_list قرار دارند، به شکل یک dataframe برگردانده می‌شوند.

قسمت اصلی این تابع TwitterSearchScraper می‌باشد که با اجرای کوئری داده‌شده به آن، توییت مورد نظر را دریافت می‌کند. در کوئری داده‌شده به تابع، کلمه‌ی کلیدی مدنظر ما، biden می‌باشد و به طور معمول می‌بایست فقط کلمه کلیدی که به دنبال آن هستیم را به عنوان کوئری وارد کنیم. اما همانطور که در شکل ۵ مشخص است، تابع اسکرپ از شروط و فیلترهای زیادی تشکیل شده‌است که در ادامه به آن‌ها می‌پردازیم و دلایل اعمال هرکدام را توضیح میدهیم.

۴-۲-۱ حذف توییت‌های دو وجهی

در طی تلاش‌های اولیه برای استخراج توییت و تحلیل احساست روی آن، بارها پیش می‌آمد که طرفداران و یا مخالفان آقای بایدن که قریب به اتفاق طرفدار دونالد ترامپ[۲۹] (رئیس جمهور پیشین و از حزب رقیب) بودند، این دو شخص را با هم مقایسه کرده و نتیجه‌گیری مدل تحلیل احساس را دچار اشتباه می‌کردند. به عنوان مثال در دو توییت زیر هر دو کلمه‌ی biden و trump به‌کار رفته‌است و نتیجه‌ی مدل تحلیل احساس نیز روی هردو منفی بوده‌است. اما همانطور که مشخص است، کاربر اول احساسات منفی خود را نسبت به بایدن ابراز داشته اما احساس منفیِ شخص دوم، نسبت به ترامپ بوده و به‌طور مشخص نظرش درباره بایدن، مثبت است. [۶]

[۲۹] Donald Trump

```python
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()

sample_texts = [

    """Trump wanting a military parade to honor the military is bad,
    but Biden arranges a boat parade around the continent and no one bats an eye""",

    """Donald Trump has inexplicably escaped any responsibility for his actions for the last five decades.
    But he's finally met his match in President Biden who is ordering the national
    archives to turn over documents Trump is trying to hide."""

]

for text in sample_texts:
    print(sid.polarity_scores(text))
```

✓ 0.8s

```
{'neg': 0.167, 'neu': 0.763, 'pos': 0.07, 'compound': -0.4497}
{'neg': 0.051, 'neu': 0.949, 'pos': 0.0, 'compound': -0.2617}
```

شکل ۲-۴ – نتیجه گیری گمراه‌کننده در توییت‌هایی که به مقایسه پرداخته‌اند

خوشبختانه از آنجایی که کتابخانه snscrape به‌طور غیرمستقیم با API توییتر ارتباط برقرار می‌کند و امکان اعمال محدودیت روی کوئری‌های API فراهم است، میتوان با توجه به مستندات فراهم شده در سایت توییتر، فیلترهای مورد نیاز را اعمال کرده و فقط توییت‌های واجد شرایط را دریافت کنیم. برای اعمال محدودیتی که توییت‌های حاوی کلمه trump استخراج نشوند، می‌بایست trump- را در ادامه کوئری اصلی اضافه کنیم.

۲-۲-٤ مشخص کردن بازه زمانی

برای انجام پروژه نیاز بود تا دیتاسِت‌های مختلفی در بازه‌های زمانی مشخص تشکیل شوند تا با اعمال مدل تحلیل احساس روی هر دیتاست[۳۰]، تغییرات نظرات کاربران را در طول سال بررسی کنیم. برای اینکار میبایست چندین تابع اسکرپ را در بازه‌های زمانی متفاوت اجرا کنیم. برای اعمال محدودیت روی بازه‌ی زمانی‌ای که توییت‌ها استخراج شوند، باید در کوئری، متغییرهای :since و :until را با تاریخ‌های مدنظر در هر قسمت، مقداردهی کنیم. در اینجا، مقادیر مدنظر به‌صورت آرگومان[۳۱] ورودی به تابع استخراج داده می‌شوند. با اینکار تابع اسکریپر صرفا در بازه‌ی زمانی مشخص شده، اقدام به جستجو برای توییت‌های مناسب می‌کند.

لازم به ذکر است در صورت استفاده از کتابخانه‌های متکی به API ، امکان دسترسی به توییت‌های قدیمی را نداشتیم. در بخش‌های آینده، نحوه تاریخ‌بندی برای تابع اسکرپ شرح داده خواهد شد.

---

[۳۰] Dataset

[۳۱] Argument

۳-۲-٤ دریافت توییت‌های انگلیسی

از آنجایی که مدل تحلیل احساسِ مورد استفاده در برنامه، روی متن‌های انگلیسی آموزش دیده‌است، نیاز بود تا توییت‌های دریافتی نیز صرفا به زبان انگلیسی باشند. یکی از مزیت‌های اعمال فیلتر روی کوئری، از بین بردن مشکلِ وجود توییت‌های غیرانگلیسی است. در صورت پیش‌بردن راهبرد اول برای انجام پروژه (دریافت تمام داده‌ها و سپس فیلتر کردنِ دیتاهای دریافت‌شده) نیاز بود تا مدل‌های تشخیص زبان را به برنامه اضافه کنیم که باعث سنگین‌تر شدن کار میشد. اما با اعمال فیلتر lang:en روی کوئری اجرایی، تشخیص زبان را به تابع اسکرپ و (به‌طور غیرمستقیم) API توییتر واگذار کردیم.

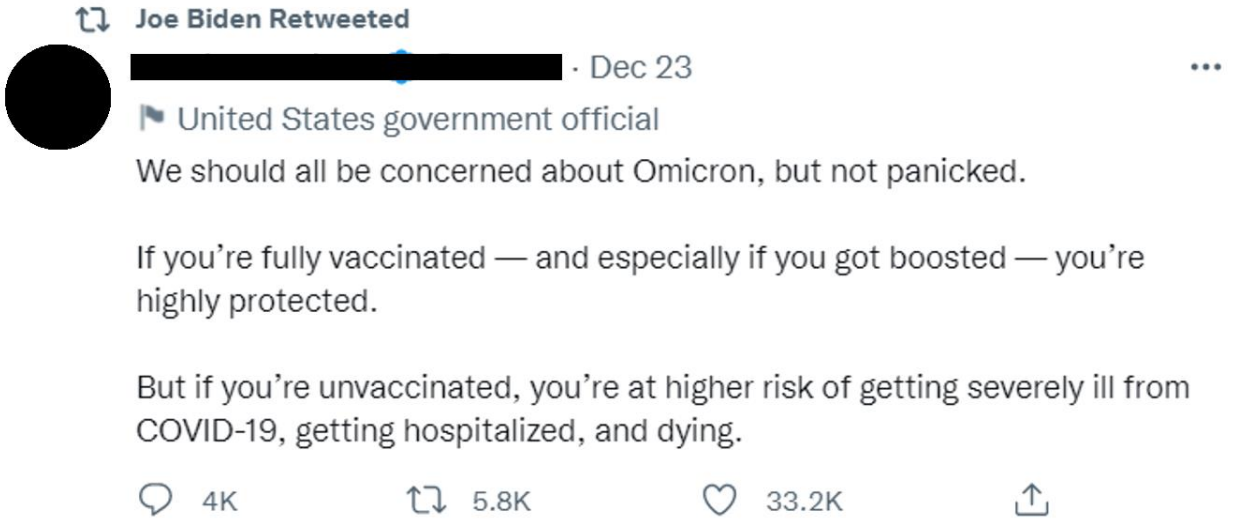
دریافت توییت‌های صرفا انگلیسی، مضاف بر بالابردن دقت تحلیل احساس، مزیت دیگری نیز به دنبال خود دارد؛ با دریافت توییت‌های انگلیسی می‌توان این ادعا را داشت که توییت‌هایی که بررسی شده‌اند، نظرات کاربرانی بوده‌اند که در آمریکا زندگی می‌کنند که بدون شک زندگی آن‌ها بیشتر از افراد دیگر در دنیا، تحت تاثیر تصمیمات بایدن است. در شکل زیر مشاهده می‌شود که از یک نمونه‌ی ۳۰تایی که استخراج شده، ۲۰ توییت متعلق به کاربرانی بوده‌است که محل زندگی را در مناطق مختلفی از ایالات متحده آمریکا انتخاب کرده‌اند.

	Datetime	Tweet Id	Text	Username	Location
2	2021-10-1	1.45E+18	I am wond	CryptoBull2020	MOONLAND
3	2021-10-1	1.45E+18	@alenkork	mccflint	
4	2021-10-1	1.45E+18	@CoriBusl	BlueBerriesWild	Surfside Beach, SC
5	2021-10-1	1.45E+18	â€œThe	clivehbest	
6	2021-10-1	1.45E+18	around the	david_darmofal	Columbia, SC
7	2021-10-1	1.45E+18	COMING	GeneralBrnovich	Arizona, USA
8	2021-10-1	1.45E+18	Add the Ba	RichardGrenell	LA, Rancho Mirage
9	2021-10-1	1.45E+18	@POTUS /	lloydtroe	Riverside, CA
10	2021-10-1	1.45E+18	Gee I won	kandibar2011	#Wisconsin , USA
11	2021-10-1	1.45E+18	@POTUS (	AmericaFightHim	United States
12	2021-10-1	1.45E+18	@Tylspn @	DavidPrecht	Brooklyn
13	2021-10-1	1.45E+18	Pres. Bider	nowthisnews	The Internet & NYC
14	2021-10-1	1.45E+18	It's easy	peterdaou	
15	2021-10-1	1.45E+18	@CoriBusl	TinkResists	
16	2021-10-1	1.45E+18	@CoriBusl	VoteandMask	Ohio, USA Sane REALITY
17	2021-10-1	1.45E+18	@POTUS \	ByDruidic	Fort Lauderdale, FL
18	2021-10-1	1.45E+18	Biden may	Tony19542	United States
19	2021-10-1	1.45E+18	@CoriBusl	SGempka	
20	2021-10-1	1.45E+18	@JackPos	ApprtionMission	South Dakota, USA
21	2021-10-1	1.45E+18	@CoriBusl	clementadesina	Midrand, South Africa
22	2021-10-1	1.45E+18	@RVnGrar	MCCLAiN142	Noneyabusiness
23	2021-10-1	1.45E+18	The Biden	TennesseeBoy8	
24	2021-10-1	1.45E+18	@senrob	jhofromcanton	Ohio, USA
25	2021-10-1	1.45E+18	@CoriBusl	SherriBursey	Maryland, USA
26	2021-10-1	1.45E+18	@CoriBusl	DangeloMorrison	Atlanta, GA
27	2021-10-1	1.45E+18	Times Squ	FoxNews	U.S.A.
28	2021-10-1	1.45E+18	Biden	mzjacobson	Stanford University
29	2021-10-1	1.45E+18	@Druzzer5	rellimmoney	Orange, California
30	2021-10-1	1.45E+18	@CoriBusl	echndc	San Francisco, CA
31	2021-10-1	1.45E+18	Haregew	Janeth49035458	

شکل ۳-۴ - عمده‌ی توییت‌های انگلیسی از آمریکا هستند

٤-٢-٤ حذف ریتوییت‌ها

ریتوییت[۳۲]، به دوباره ارسال کردن یک توییتِ دلخواه گفته می‌شود. دلیل اصلی‌ای که ریتوییت‌ها در تحلیل درنظر گرفته نشدند، محتوای یکسان آن‌ها با توییت اصلی بود. ریتوییت تمام محتوای خود از جمله تعداد لایک، کامنت‌های ثبت‌شده، متن اصلی و... را از توییت اصلی به ارث می‌برد و علاوه بر اینکه محتوای جدیدی به دیتاست اضافه نمی‌کند، مانع اضافه شدن توییت‌های دیگر نیز می‌شود. برای فیلتر کردن ریتوییت‌ها کافی است که filter:retweets- به کوئری اضافه شود.



شکل ۴-۴ - نمونه یک ریتوییت

٤-٢-٥ فقط یک صدا برای هر نفر

هر بار که توییتی استخراج می‌شود، آیدی کاربری که آن را فرستاده ذخیره می‌شود تا توییت‌های وی دیگر دریافت نشود. این مسئله از این جهت اهمیت پیدا می‌کند که همیشه افرادی هستند که با اختلاف بیشتر از بقیه توییت میزنند. این افراد معمولا در یکی از ۳ دسته زیر قرار می‌گیرند:

- اکانت‌های مرتبط با خبرگزاری‌ها: در حالتی که این فیلتر اعمال نمی‌شد، بخش زیادی از توییت‌های استخراج شده را اکانت‌های خبری مانند foxnews ، CNN و... در بر می‌گرفتند و جای زیادی برای دریافت نظر بقیه مردم باقی نمی‌ماند.
- ربات‌ها[۳۳]: معمولا چندین اکانت با هویت‌های متفاوت هستند که یک متن مشخص را ارسال میکنند. هرچند که شناسایی ربات‌ها در شبکه‌های اجتماعی آسان نیست و خودِ اینکار، پروژه‌ای جداگانه میطلبد

[۳۲] Retweet

[۳۳] Robot

اما با اعمال این فیلتر، در هر بازه زمانی صرفا یکبار توییت‌های اکانت‌های ربات دریافت شده و می‌توان گفت تا حدی با آن‌ها مقابله کرده‌ایم.[۷]

- اقلیت پر سروصدا[۳۴]: این افراد معمولا مخالف بایدن بودند و احساساتشان نسبت به وی به شدت منفی بود. این افراد معمولا با اختلاف توییت‌های بیشتری نسبت به دیگر کاربران ارسال می‌کردند. هرچند که به‌نظر نمیرسد این افراد نماینده بخش زیادی از جامعه باشند اما توییت‌های این افراد در نتایج تاثیرگذار هستند.[۸]

٤-٢-٦ میزان تعامل بقیه کاربران با توییت

بسیاری از توییت‌ها بودند که تعداد لایک و کامنت‌های بسیار ناچیزی داشتند و درکل به‌نظر می‌رسید که در فضای توییتر، بسیار کمتر از حد معمول دیده شدند. یکی بهترین راه‌ها برای حساب کردن میزان تعامل بقیه افراد با یک توییت، بررسی تعداد engagement های آن است.[۹] اما متاسفانه راهی برای دستیابی به این متغییر برای توییت‌های بقیه افراد وجود ندارد؛ پس می‌بایست راهکار دیگری در پیش گرفته‌شود.

شکل ۵-۴ - اطلاعاتی که هر توییت در خود دارد

[۳۴] Vocal minority

هنگام دریافت یک توییت، حدود ۵۰ متغییر همراه آن دریافت می‌شود. از میان آن‌ها، ۴ متغییری که بر مبنای تعاملات بقیه کاربران تعیین می‌شوند عبارتند از تعداد لایک، کامنت، ریتوییت و نقل‌قول. برای تعیین میزان تعامل با توییت، این ۴ متغییر را به وسیله تابع popularity با هم جمع کردیم. حداقل میزان تعامل را نیز عدد ۲۰ درنظر گرفتیم که با اینکه اصلا مقدار بالایی نیست، ولی درنهایت توییت‌هایی داریم که نماینده بهتری برای گروه‌های مختلف جامعه هستند.

```
def popularity(tweet):
    return (tweet.likeCount + tweet.retweetCount +
            tweet.replyCount + tweet.quoteCount)
```

شکل ۶-۴ – تابع popularity برای تعیین میزان تعاملات بقیه کاربران با توییت

۴-۳ گزارشی از وضعیت اسکریپ در طول اجرا

از آنجایی که فرایند استخراج توییت‌ها بسیار زمانبر بوده و معمولا چند صدهزار تا چندین میلیون توییت در این فرایند دریافت می‌شود، نیاز بود تا گزارشی از وضعیت تابع استخراج داشته باشیم. با اینکار، اگر خطایی در حین اجرا پیش می‌آمد یا اجرا متوقف می‌شد و یا به هر دلیل سرعت پایین می‌آمد، از آن مطلع می‌شدیم. برای اینکار، تابع گزارش را درون حلقه‌ای که توییتِ مورد قبول را به لیست موردنظر اضافه می‌کند، فراخوانی کردیم.

```
scraped_tweets = 0
base_time = time.time()
rate = 1
def scrape_report():
    global scraped_tweets, base_time, rate, total_tweets

    scraped_tweets += 1

    if scraped_tweets % 10 == 0:
        current_time = time.time()
        elapsed_time = current_time - base_time

        if elapsed_time > 0:
            rate = round( 10 / elapsed_time , 2)
            base_time = current_time

    remaining_time = round((((total_tweets - scraped_tweets) / rate) / 60) , 2)

    print('Total tweets = {} \tEstimated remaining time = {} minutes \tRate = {} tweets/sec'
    .format(scraped_tweets, remaining_time , rate) , end="\r")
```

شکل ۷-۴ – تابع گزارش استخراج

از آنجایی که فرایند استخراج با چندین نخ به‌طور همزمان انجام می‌شود، نیاز بود تا متغییرهایی که در این تابع مورد استفاده قرار گرفتند، در همه توابع قابل دسترسی و تغییر باشند. به همین دلیل آن‌ها را به شکل سراسری تعریف کردیم و در تابع نیز از همین دید[۳۵] استفاده کردیم. در این تابع تعداد توییت‌هایی که تا آن لحظه استخراج شده‌اند، نرخ[۳۶] اسکریپ بر دقیقه و تخمینی از زمان باقیمانده برای اتمام فرایند استخراج نمایش داده می‌شود.

متغییر scaraped_tweets تعداد توییت‌های قابل پذیرش را نشان می‌دهد و در هر بار اجرای تابع گزارش، یک واحد افزایش می‌یابد. نرخ استخراج، به ازای هر ۱۰ توییتی که استخراج می‌شود آپدیت می‌شود. برای بدست آوردن نرخ استخراج، ابتدا زمان سپری‌شده از اجرای پیشینِ تابع تا لحظه کنونی را اندازه می‌گیریم. سپس ۱۰ را تقسیم بر زمان سپری شده کرده تا دریابیم که به‌طور متوسط در هرثانیه، چند توییت درحال استخراج است. برای اندازه گیری زمان باقیمانده نیز تعداد توییت‌های باقیمانده برای اتمام کار را بر نرخ استخراج تقسیم کرده و نتیجه را بر ۶۰ تقسیم می‌کنیم تا زمان بدست آمده بر حسب دقیقه باشد.

بعضا پیش می‌آمد که دو یا چند نخ همزمان rate را آپدیت کنند. این مسئله سبب می‌شد که مقدار elapsed_time صفر شود و هنگام آپدیت متغییر rate ، خطای تقسیم بر صفر پیش بیاید. به همین دلیل شرط بزرگتر از صفر بودنِ متغیر elapsed_time را اعمال کردیم. دلیلی که نرخ استخراج پس از هر ۱۰ توییت آپدیت می‌شود این است که آپدیت کردن با فواصل خیلی کم، نتایج این تابع را هربار به مقدار زیادی تغییر میداد و با زیاد کردن این فاصله به مقدار ۱۰، دقت کار را بالا بردیم.

راهکار دیگری نیز برای بدست آوردن نرخ استخراج و زمان باقیمانده وجود داشت که در آن حالت زمان سپری شده را برابر با تفاضل زمان اولین اجرای تابع scrape_report تا لحظه اجرای فعلی قرار می‌دادیم. در آن روش مقدار متغییر start_time را در تابع get_tweets به عنوان آرگومان ورودی به تابع گزارش میدادیم. در آن صورت نرخ استخراج برابر با حاصل تقسیم تعداد توییت‌های استخراج شده بر زمان سپری شده بود. همچنین زمان باقیمانده نیز از رابطه $\frac{(total\_tweets-scraped\_tweets)*elapesed\_time}{scraped\_tweets}$ حاصل می‌شد.

یکی از چالش‌های اسکریپ کردن از توییتر این بود که سرعت استخراج، در ساعاتی از روز که ترافیک اینترنت بالا بود، به‌شدت کم میشد و نرخ استخراج از حداکثر ۲۰ tweets/second به مقادیر بسیار پایینی در حد ۰.۵ tweets/second تقلیل می‌یافت. اشکال روش یاد شده، این بود که هماهنگ شدن با این کاهش شدید با آن ممکن نبود و زمان‌های اشتباه را نمایش می‌داد.

[۳۵] Scope

[۳۶] Rate

```
scraped_tweets = 0
def scrape_report(start_time):
    global scraped_tweets, total_tweets

    scraped_tweets += 1
    elapsed_time = time.time() - start_time
    remaining_time = round( (((total_tweets / scraped_tweets) - 1) * elapsed_time) / 60 , 2)
    rate = round(scraped_tweets / elapsed_time, 2)

    print('Total tweets = {} \tEstimated remaining time = {} minutes \tRate = {} tweets/sec'
    .format(scraped_tweets, remaining_time , rate) , end="\r")
```

شکل ۸-۴- محاسبه‌ی اشتباهِ زمان سپری شده

٤-٤ آرگومان‌های تابع اسکریپر

به‌طور کلی پروژه‌ی انجام شده، بر تحلیل احساسات متن در طی مدت زمان مشخص (سال ۲۰۲۱ میلادی) تمرکز دارد. پس میبایست بازه‌های زمانی که می‌خواهیم تابع اسکرِیپر روی آن‌ها کار کند را تعیین کنیم. برای انجام این پروژه از تاریخ ۲۹ دسامبر ۲۰۲۰ تا ۱۴ دسامبر ۲۰۲۱ در بازه‌های زمانی ۱۰ روزه زمان‌ها را تنظیم می‌کنیم. تعداد توییت‌های دریافتی در هر بازه زمانی برابر با ۱۰۰۰۰ عدد و تعداد کل توییت‌های دریافتی برابر با ۳۵۰۰۰۰ می‌باشد.

```
NO_tweets_in_datasets = 10000
total_datasets = 35
total_tweets = NO_tweets_in_datasets * total_datasets

base_date = date(2021,12,14)
delta_date = timedelta(days = 10)

scraper_args = []

for i in range(total_datasets):

    argument = (NO_tweets_in_datasets, (base_date - delta_date)
    .strftime("%Y-%m-%d"), base_date.strftime("%Y-%m-%d"))

    scraper_args.append(argument)
    base_date -= delta_date
```

شکل ۹-۴ - ایجاد آرگومان‌های تابع اسکریپر

تابع timedelta با دریافت تعداد روزهای موردنظر، قابلیت ایجاد مقداری از نوع datetime.date را دارد که بخاطر یکسان بودن با نوع متغییر base_date می‌توان عملیات تفریق را روی آن‌ها انجام داد. تمام عملیات ایجاد تاریخ‌های مورد نیاز، توسط کتابخانه datetime انجام شد.

یکی از توابعی که در این کتابخانه پیاده‌سازی شده‌است، strftime می‌باشد که قابلیت چاپ کردن تاریخ با فرمت مورد نیاز را فراهم می‌کند. از آنجایی که در کوئری ورودی به تابع استخراج باید فرمت[۳۷] تاریخ‌های شروع و پایان هردو به شکل Year-Month-Day باشد، آرگومان ورودی strftime را به همین شکل وارد می‌کنیم.

در حلقه‌ای که برای تولید آرگومان‌های تابع اسکریپر استفاده شده، متغییر argument که از نوع tuple می‌باشد، هربار با مقداری که برای ورودی‌های این تابع مورد استفاده قرار میگیرد، مقداردهی می‌شود. اندیس[۳۸] اول argument تعداد توییت‌هایی است که باید تابع اسکریپر استخراج کند. اندیس دوم، تاریخ شروع و اندیس سوم هم تاریخ پایان است که در بخش‌های پیشین درباره آن‌ها توضیح داده شد.

```
[(10000, '2021-12-04', '2021-12-14'),
(10000, '2021-11-24', '2021-12-04'),
(10000, '2021-11-14', '2021-11-24'),
(10000, '2021-11-04', '2021-11-14'),
(10000, '2021-10-25', '2021-11-04'),
(10000, '2021-10-15', '2021-10-25'),
...
(10000, '2021-02-07', '2021-02-17'),
(10000, '2021-01-28', '2021-02-07'),
(10000, '2021-01-18', '2021-01-28'),
(10000, '2021-01-08', '2021-01-18'),
(10000, '2020-12-29', '2021-01-08')]
```

شکل ۱۰-۴- لیست آرگومان‌های تابع اسکریپ

[۳۷] Format

[۳۸] Index

٤-٥ شروع استخراج

از آنجایی که نیاز بود تا ۳۵ دیتاست داشته باشیم، تصمیم بر آن شد که ۳۵ تابع استخراج را همزمان به وسیله نخ[۳۹] اجرا کرده تا کار، روند مشخص‌تری را طی کند. اما چالشی که وجود داشت این بود که در زبان برنامه نویسی پایتون، نخ‌ها قابلیت برگرداندن متغییر را نداشته و صرفا توابعی در آن‌ها اجرا می‌شود که درنهایت مقدار بازگشتی نداشته باشند. این درحالی بود که تابع استخراج، درنهایت یک dataframe را باید بازمی‌گرداند.

برای حل این چالش، کلاسی به نام ThreadWithReturnValue تعریف شد که یک thread معمولی و رایج در پایتون را به عنوان آرگومان ورودی می‌گیرد و پس از اجرای آن، مقدار بازگشتیِ تابعی که به عنوان target دریافت کرده‌است را برمی‌گرداند. لازم به ذکر است که در این کلاس، تمام آرگومان‌های کلاس thread پایتون پیاده‌سازی شده‌اند و می‌توان گفت که صرفا کمی شخصی‌سازی برای اجرای توابعِ با مقدار بازگشتی، در آن انجام شده‌است.

```
class ThreadWithReturnValue(Thread):
    def __init__(self, group=None, target=None, name=None,
                 args=(), kwargs={}, Verbose=None):
        Thread.__init__(self, group, target, name, args, kwargs)
        self._return = None
    def run(self):
        if self._target is not None:
            self._return = self._target(*self._args,
                                        **self._kwargs)
    def join(self, *args):
        Thread.join(self, *args)
        return self._return
```

شکل ۱۱-۴- کلاس نخ با مقدار بازگشتی

اجرای این کلاس با روش رایج اجرای نخ در پایتون تفاوت زیادی ندارد. در مرحله اول کافی است لیستی از thread ها بسازیم. برای اینکار باید target و args را مشخص کنیم. target در واقع تابعی است که نخ باید آن را اجرا کند و args نیز آرگومان‌های ورودیِ target خواهند بود. لازم به ذکر است که نوع داده آرگومانِ args، باید از نوع tuple باشد و دلیل استفاده از tuple در بخش پیشین نیز همین مسئله بود. برای ساختن نخ‌ها، کافی است به

[۳۹] Thread

اندازه طول آرایه scraper_args حلقه‌ای اجرا شود و در هر اجرای آن، به ترتیب یکی از tuple های آن را به عنوان args و تابع get_tweets را به عنوان target به نخ بدهیم و آن نخ را در نهایت به یک لیست[۴۰] اضافه کنیم.

در مراحل بعدی نیز تقریبا مثل اجرای نخ معمولی در پایتون عمل می‌کنیم. در یک حلقه‌ی for تابع start را برای تمام نخ‌هایی که در مرحله پیشین ساخته بودیم اجرا میکنیم. سپس بار دیگر باید یک حلقه اجرا کنیم. تفاوت کلاس ThreadWithReturnValue با نخ معمولی در این قسمت مشخص می‌شود. در نخ معمولی باید تابع join بدون مقدار بازگشتی اجرا شود اما در اینجا متغییری به‌نام tweet_dataframe داریم که مقدار بازگشتیِ نخ، در آن قرار می‌گیرد.

```
df_list = []
threads = []

for i in range(len(scraper_args)):
    twrv = ThreadWithReturnValue(target=get_tweets, args = scraper_args[i])
    threads.append(twrv)

for t in threads:
    t.start()

for t in threads:
    tweet_dataframe = t.join()
    df_list.append(tweet_dataframe)
```

شکل ۱۲-۴- اجرای نخ‌ها و استخراج موازی توییت‌ها در نتیجه آن

اجرای این قسمت از کد، طولانی‌تر قسمت پروژه بود و در مجموع بیش از ۱ روز کامل به طول انجامید. هرچند که با استفاده از کتابخانه snscrape محدودیت‌های API برداشته شد، اما همچنان به دلیل وجود فیلترهای زیاد، بسیاری از توییت‌ها مورد قبولِ تابع اسکریپر (get_tweets) واقع نمی‌شدند و این مسئله باعث طولانی شدن این فرایند گردید.

[۴۰] List

٤-٦ ذخیره کردن اطلاعات روی سیستم

در نهایت اطلاعاتی که دریافت شدند روی سیستم ذخیره شده و از آن‌ها بک‌آپ[۴۱] گرفته شد. برای ذخیره کردن dataframe فرمت‌های زیادی وجود دارد که در شکل مشخص شده‌اند. دلایل استفاده از CSV در این پروژه، رایج بودن، راحتی کار و سرعت بالا بود. از دیگر دلایل انتخاب CSV، مصرف حافظه کمتر نسبت به اکسل[۴۲] بود که این مسئله به‌دلیل ساختار ساده‌ی این فرمت می‌باشد. همچنین فایل‌های CSV قابلیت ویرایش در ویرایشگرهای متنی را دارا می‌باشند که این مسئله در اکسل وجود ندارد.

pandas.DataFrame.to_clipboard	pandas.DataFrame.to_numpy
pandas.DataFrame.to_csv	pandas.DataFrame.to_parquet
pandas.DataFrame.to_dict	pandas.DataFrame.to_period
pandas.DataFrame.to_excel	pandas.DataFrame.to_pickle
pandas.DataFrame.to_feather	pandas.DataFrame.to_records
pandas.DataFrame.to_gbq	pandas.DataFrame.to_sql
pandas.DataFrame.to_hdf	pandas.DataFrame.to_stata
pandas.DataFrame.to_html	pandas.DataFrame.to_string
pandas.DataFrame.to_json	pandas.DataFrame.to_timestamp
pandas.DataFrame.to_latex	pandas.DataFrame.to_xarray
pandas.DataFrame.to_markdown	pandas.DataFrame.to_xml

شکل ۱۳-۴- فرمت‌های قابل خروجی گرفتن در کتابخانه pandas

از آنجایی که نقطه متمایزکننده‌ی این ۳۵ dataframe، بازه‌ی زمانیِ محتوای آن‌ها بود، تصمیم بر آن شد که نام فایل‌ها را تاریخ اولین توییتِ دریافتی در هر dataframe قرار دهیم. برای اینکار ابتدا مسیری که قرار است فایل‌ها را در سیستم ذخیره کنیم مشخص میشود. سپس تاریخ و زمان اولین توییت در آن dataframe را انتخاب کرده و قسمتی که مربوط به تاریخ می‌شود را جدا می‌کنیم و به عنوان نام فایل قرار می‌دهیم. سپس پسوند csv. به انتهای نام فایل افزوده می‌شود. در نهایت نیز با تابع to_csv که در کتابخانه pandas پیاده‌سازی شده‌است، عملیات خروجی گرفتن از dataframe را به پایان می‌رسانیم.

[۴۱] Backup

[۴۲] Excel

```
for i in range(len(df_list)):
    path=r"C:\Users\pouria\Desktop\project\datasets\\"
    filename = path + (df_list[i]['Datetime'].iloc[[0]]).to_string()[4:14] + '.csv'
    df_list[i].to_csv(filename, sep=',', index=False)
```

شکل ۱۴-۴- ذخیره کردن dataframe ها

برای خواندن فایل‌های csv ذخیره شده نیز می‌توان با استفاده از قطعه کد زیر آن‌ها را وارد حافظه برنامه کرد و ادامه کار را پی گرفت. برای این‌کار مسیر فایل‌های ذخیره شده را با استفاده از کتابخانه glob باید وارد کرد و با تعیین فرمت کلی فایل‌ها به‌صورت csv.* تابع را اجرا کرد. سپس با استفاده از تابع read_csv که در کتابخانه pandas پیاده‌سازی شده‌است فایل های csv را به شکل datadframe وارد برنامه کرد.

```
path = r'C:\Users\pouria\Desktop\project\datasets'
csv_files = glob.glob(os.path.join(path, "*.csv"))

dataFrames = []
for f in csv_files:

    df = pd.read_csv(f)
    dataFrames.append(df)
```

شکل ۱۵-۴- خواندن فایل‌های csv

٤-٧ پیش‌پردازش متون

اصولا در هر زمینه‌ای که با داده سروکار داشته‌باشیم در قریب به اتفاق موارد، آن داده‌ها مرتب نیستند و باید ابتدا پیش‌پردازشی روی آن‌ها انجام گیرد تا آن‌ها را به صورتی که مدنظر ما است تبدیل کنیم. به عنوان مثال این پیش‌پردازش در پروژه‌های سری زمانی[۴۳]، شامل مواردی مثل میانگین گیری از اعداد برای قرار دادن در فیلدهای NaN است. برای پروژه‌هایی که در حوزه NLP هستند نیز چنین فرایندی باید انجام شود. برای اینکار، تابعی به‌نام clean_text نوشته شد تا پیش‌پردازش متن را برای ما انجام دهد تا متن توییت‌ها، آماده‌ی تحلیل احساس شوند. این تابع چندین بخش دارد که در ادامه به آن‌ها پرداخته خواهد شد.

[۴۳] Time series

```
nlp = spacy.load('en_core_web_sm')
stopwords = nlp.Defaults.stop_words
stopwords.remove("not")

def clean_text(tweet : str):

    tweet = tweet.lower()
    tweet = re.sub("@[A-Za-z0-9_]+","", tweet)
    tweet = re.sub("#[A-Za-z0-9_]+","", tweet)
    tweet = re.sub(r"http\S+|www\S+|https\S+", '', tweet, flags=re.MULTILINE)

    tweet = nlp(tweet)
    tweet = ' '.join([token.lemma_ for token in tweet])
    tweet = ' '.join([word for word in tweet.split() if not word in stopwords])

    return tweet
```

شکل ۱۶-۴ - پیش‌پردازش متن توییت‌ها

۱-۷-٤ کوچک کردن حروف جملات

از آنجایی که توییت‌های دریافتی همگی به زبان انگلیسی هستند و الفبای این زبان، شامل حروف بزرگ و کوچک است، در نتیجه جملات از این نظر یک‌دست نیستند. دلیل استفاده از حروف بزرگ در توییت‌های دریافتی معمولا یکی از موارد زیر بود:

- حرف ابتداییِ اولین کلمه در جمله
- حرف ابتداییِ اسم‌های خاص (اشخاص، مکان‌ها و...)
- کلماتی که شخص روی آنها تاکید دارد و یا حالت عصبانیت را بیان می‌کنند
- کلماتی که شخص از روی کنایه بیان می‌کند

برای حل این مسئله، بهترین کار این است که تمام حروف جملات را به شکل کوچکشان تبدیل کنیم. برای اینکار از تابع lower استفاده می‌کنیم. با اینکار حروف بزرگ از رشته‌ی[۴۴] ورودی به تابع، حذف خواهد شد که موجب می‌شود تمام کلمات برای تابع تحلیل احساس، یک‌دست شوند و دقت کارِ این تابع بالاتر برود.

[۴۴] String

۲-۷-٤ حذف کردن مِنشِن‌ها

منشن[۴۵] کردن در توییتر عملی است که کاربری به‌طور مستقیم در توییتش، کاربر دیگری را خطاب قرار می‌دهد. اینکار معمولا در توییت‌ها و ریپلای‌های هر توییت انجام می‌شود. منشن کردن به‌صورت @username انجام می‌شود به طوری که ابتدا علامت @ و سپس نام کاربریِ شخصی که مورد خطاب است، قرار میگیرد. نکته‌ای که وجود دارد این است که منشن‌ها در تحلیل احساس هیچ نقشی نداشته و صرفا حروف اضافی‌ای هستند که حضورشان در کارِ پردازش، فقط بار اضافی دارد. به همین دلیل آن‌ها را حذف می‌کنیم.

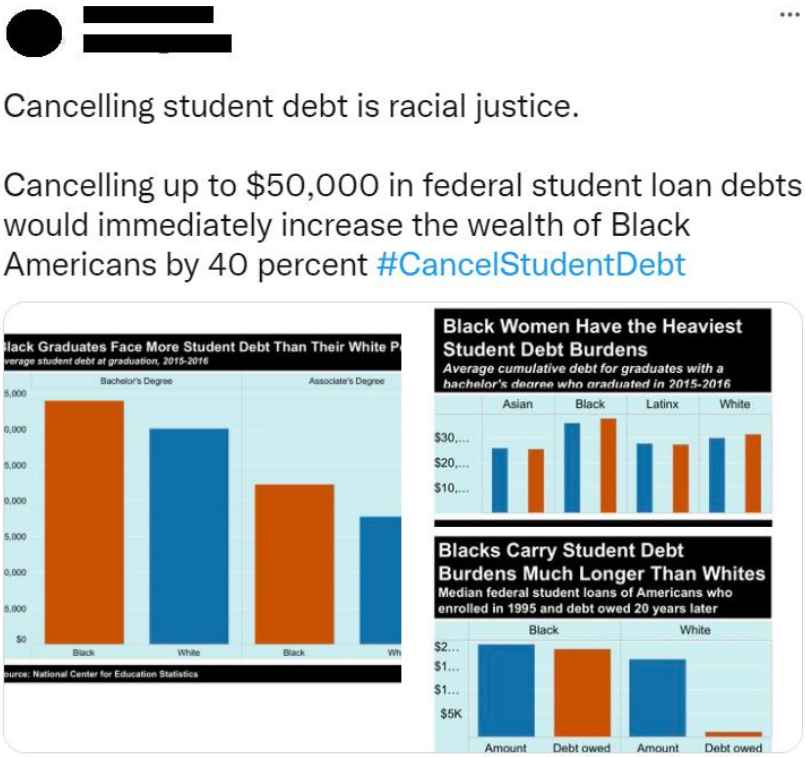
شکل ۱۷-۴ - نمونه منشن کردن در پلتفرم توییتر

[۴۵] Mention

از آنجایی که منشن همواره با @ شروع می‌شود، میتوان آن‌ها را توسط regular expression شناسایی کرد. برای این‌کار از کتابخانه‌ی re استفاده می‌کنیم. مزیت این‌کار، سرعت بالای آن است که در نتیجه‌ی آن، حافظه‌ی کمی مصرف می‌کند. از آنجایی که این عملیات باید روی ۳۵۰۰۰۰ متن انجام شود، مصرفِ پایین حافظه مزیت بزرگی است.

۳-۷-٤ حذف کردن هشتگ‌ها

هَشتَگ[۴۶] یا هش‌تگ با نشان #، یک نماد پیشوندی و یکی از تگ‌های فراداده است که در خدمات شبکه‌های اجتماعی و میکروبلاگینگ استفاده می‌شود. اگر بخواهیم هشتگ را به زبان ساده‌تر تعریف کنیم، برچسبی است که برای دسته‌بندی و به اشتراک گذاری پست‌ها و نظرات درباره موضوعی خاص در سطح جهانی و فراتر از حلقه و فهرست دوستان به‌کار می‌رود. هشتگ ابزاری برای دسته‌بندی پیام‌ها فراهم می‌کند، تا افراد بتوانند آن هشتگ را جستجو کنند و مجموعه‌ای از پیام‌هایی را که شامل آن هستند به دست آورند. عموماً کلیدی‌ترین واژه مربوط به آن موضوع را با نشانه هشتگ همراه کنند. این همراه کردن با استفاده از علامت # قبل از واژه مورد نظر انجام می‌شود. در نام‌گذاری هشتگ می‌توان از حروف، اعداد و علایم مجاز استفاده کنید.



شکل ۱۸-۴ - نمونه هشتگ در پلتفرم توییتر

[۴۶] Hashtag

هشتگ نیز مانند منشن، در فرایند تحلیل احساسات هیچ نقشی نداشته و صرفا کلمات اضافی به حساب می‌آیند. از آنجایی که ساختار هشتگ به‌صورت # و سپس رشته کاراکتر است، حذف کردن آن به‌وسیله regular expression انجام شد. با اینکار، کلمات بدون کاربرد (در تحلیل احساسات) حذف شده و جملات کوتاه‌تر و ساده‌تر می‌شوند.

٤-٧-٤ حذف کردن URLها

هنگامی که کاربری بخواهد در توییت خود به وبسایت دیگری اشاره کند، از URL وبسایت مدنظرش استفاده می‌کند. URLهای توییتر با یکی از عبارات http, https, www شروع می‌شوند و در ادامه رشته‌ای از حروف بدون فاصله را به‌دنبال دارند. از آنجایی که الگوی URL همیشه مشخص است، برای حذف آن از regular expression استفاده می‌کنیم.

شکل ۱۹-۴ - نمونه یک توییت دارای URL

هرچند که در توییتی که در شکل ۴-۱۹ مشخص شده، آدرسی به‌صورت www, http, https مشخص نیست اما هنگامی که آن توییت استخراج می‌شود و به شکل رشته نمایش داده می‌شود، آدرس دریافت شده دقیقا با همین فرمت خواهد بود. همچنین توییت‌هایی که quote کرده‌اند نیز به همین صورت دریافت شده که آدرس URL به وسیله regular expression قابل شناسایی و حذف خواهد بود.

۴-۷-۵ حذف نکردن کاراکترهای خاص

به‌طور معمول، در پیش‌پردازش متن، کاراکترهای خاص مانند ; ‘ - / : ( ) و همچنین تمام کاراکترهای عددی حذف شده و متن باقیمانده صرفا شامل کلمات هستند. اما در تحلیل احساس، کاراکترهای خاص و اعداد را حذف نمی‌کنیم زیرا با حذف آن‌ها، ایموجی و شکلک‌ها (اموتیکون[۴۷]) را نیز حذف کرده‌ایم. یکی از نقات قوت کتابخانه vadersentiment که برای تحلیل احساس در این پروژه از آن استفاده شده‌است، پیاده‌سازی تحلیل احساسات برای ایموجی و اموتیکون‌ها است.

ایموجی یا شکلک در واقع تصاویر مفهومی هستند که در پیام‌های مجازی و تارنماهای جهانی استفاده می‌شود. ایموجی را می‌توان بیان گرافیکی عواطف، و احساسات نامید. شکلک (اموتیکون) یک تصویرنگار است که با استفاده از حروف و علائم نقطه‌گذاری نمایش داده می‌شود تا حس و حالت صورت شخص را نمایش دهد. ایموجی و اموتیکون در شبکه‌های اجتماعی و پیام‌رسان‌ها به‌طور گسترده استفاده می‌شود. از آنجایی که با تحلیل آن‌ها می‌توان به سادگی احساسات یک جمله را دریافت، نگه‌داشتن آنها بسیار مفید است.[۱۰] بطور مثال در شکل ۲۰-۴، یک جمله‌ی کاملا خنثی با اضافه کردن ایموجی و اموتیکون دارای احساس شده و نتایج تغییر کرده‌است.[۱۱]

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()

print(sid.polarity_scores('hello there'))

print(sid.polarity_scores('hello there :-)'))

print(sid.polarity_scores('hello there 😕'))
```

✓ 0.6s

```
{'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
{'neg': 0.0, 'neu': 0.465, 'pos': 0.535, 'compound': 0.3182}
{'neg': 0.345, 'neu': 0.655, 'pos': 0.0, 'compound': -0.2748}
```

شکل ۲۰-۴ - تاثیر ایموجی و شکلک در احساسات جمله

[۴۷] Emoticon

نکته حائز اهمیت این است که unicode ایموجی‌ها، معمولا فرمتی به شکل U+۱Fxxx دارد که x باید ارقام در مبنای ۱۶ باشد (۱, ۲, ۳, …, D, E, F) که واضح است در حالتی که تمام کاراکترهای خاص و اعداد را حذف کنیم، ایموجی‌ها را نیز از دست خواهیم داد. اموتیکون‌ها نیز که بیشتر آن‌ها صرفا از کاراکترهای خاص تشکیل شده‌اند، با اعمال فیلترِ بیان شده، از بین خواهند رفت.

٦-٧-٤ lemmatization

در تمام زبان‌ها، از یک کلمه ترکیبات زیادی وجود دارد. به عنوان مثال در زبان انگلیسی لغات walks, walking, walked به کلمه‌ی walk برمی‌گردند. یکی از مهمترین بخش‌های پیش‌پردازش متن، عمل lemmatization می‌باشد که کار آن، تبدیل تمام اشکال کلمه به شکل اصلی است. دلایل استفاده از lemmatization عبارتند از:

- در تحلیل احساس، تمام اشکال یک کلمه معنای یکسان دارند. به عنوان مثال sad, sadden, sadly همگی مفهوم ناراحت بودن را به دنبال دارند و نیازی به وجود ترکیبات پیچیده از کلمه‌ی sad وجود ندارد. پس همه‌ی آن‌ها را به شکل اصلی برمی‌گردانیم. زیرا ترکیبات متفاوتِ آن، در دقت نتایج هیچگونه تاثیری نخواهند داشت.
- با این‌کار، عملیات تابع تحلیل احساس بسیار سبک‌تر خواهد بود. همچنین در صورتی که بخواهیم خودمان به وسیله روش‌های یادگیری ماشین[۴۸] یا یادگیری عمیق[۴۹] مدلی برای تحلیل احساس آموزش دهیم، حتما می‌بایست که بسته به نیاز، یکی از عملیات lemmatization و یا stemming را روی متون اعمال کنیم تا با ساده‌تر کردنِ آن‌ها، دقت مدل بالا برود.[۱۲]
- اشکال مخفف کلمات مانند shouldn’t, didn’t, haven’t با اعمال lemmatization به شکل کامل خود یعنی should not, did not, have not برمیگردند که نقش مهمی در نتایج تحلیل احساس ایفا می‌کنند. در این‌باره در بخش‌های آینده بیشتر صحبت خواهد شد.

از آنجایی که عملیات lemmatization یکی از رایج‌ترین پیش‌پردازش‌های متن در تمام زمینه‌ها می‌باشد، در کتابخانه‌های مشهور NLP نیز پیاده‌سازی شده‌است. در شکل، روی یک جمله‌ی ثابت با استفاده از NLTK و spacy عملیات lemmatization را انجام دادیم. مشاهده می‌شود که عملکرد spacy در انجام این‌کار بهتر بوده. زیرا کلمات is, don’t , concerning دقیقا به همان شکلی که میخواستیم تبدیل شده‌اند.[۱۳]

[۴۸] Machine learning

[۴۹] Deep learning

```
sentence = """There is much that we don't yet know concerning how electroreception functions."""
#############################################################################################


from nltk.stem import WordNetLemmatizer
from nltk import word_tokenize

words = word_tokenize(sentence)
wordnet_lemmatizer = WordNetLemmatizer()

print("NLTK lemmatization results: ")
print(" ".join([wordnet_lemmatizer.lemmatize(word) for word in words]))
#############################################################################################


import spacy
nlp = spacy.load('en_core_web_sm')
doc = nlp(sentence)

print("\nspacy lemmatization results: ")
print(" ".join([token.lemma_ for token in doc]))
```

✓ 6.7s

```
NLTK lemmatization results:
There is much that we do n't yet know concerning how electroreception function .

spacy lemmatization results:
there be much that we do not yet know concern how electroreception function .
```

شکل ۲۱-۴ – تفاوت عملکرد کتابخانه‌های NLTK و spacy برای lemmatization

۴-۷-۷ حذف کردن کلمات توقف

کلمات توقف کلماتی هستند که در ساختار جلمه نقش مهمی ایفا می‌کنند اما برای تحلیل احساس هیچ ارزشی ندارند. به عنوان در شکل زیر جمله‌ی اولیه درنهایت بسیار کوتاه‌تر شده اما اگر شکل ابتدایی و شکل کوتاه‌شده آن را به هر مدل تحلیل احساسی بدهیم، نتایج برابری دریافت خواهیم کرد. در نتیجه برای سبک‌تر شدن عملیات تحلیل احساس، تمام کلمات توقف را حذف میکنیم.[۱۴]

```
import spacy
nlp = spacy.load('en_core_web_sm')
stopwords = nlp.Defaults.stop_words

sentence = """ I just want to make you happy"""

print('sentence after removing stop words:')
print(' '.join([word for word in sentence.split() if not word in stopwords]))
```

✓ 6.4s

```
sentence after removing stop words:
I want happy
```

شکل ۲۲-۴ - حذف کلمات توقف

حذف کلمات توقف نیز به علت رایج بودن در بیشترحوزه‌های مرتبط با NLP، تقریبا درتمامی کتابخانه‌های پردازش زبان طبیعی، پیاده‌سازی شده است. اما دلیل انتخاب spacy برای اینکار، تعداد زیاد کلمات توقف این کتابخانه بود. برای مثال در NLTK تنها ۱۷۹ کلمه توقف وجود دارد و یا در scikit-learn این تعداد ۳۱۸ عدد می‌باشد. اما spacy با داشتن ۳۲۶ کلمه توقف، قطعا عملکرد بهتری از خود ارائه می‌دهد.

نکته بسیار مهمی که وجود دارد این است که کلمه not که در زبان انگلیسی برای منفی کردن استفاده می‌شود، در لیست کلمات توقف وجود دارد. اما در صورتی که این کلمه را در کنار بقیه کلمات توقف حذف کنیم، احساسات جمله تغییر می‌کند. به‌طور مثال در جمله زیر، تابع تحلیل احساس پیش‌بینی اشتباهی انجام داده که به علت حذف کلمه not از جمله بوده‌است. به همین علت، در ابتدای قسمت پیش‌پردازش که لیست کلمات توقف را فراخوانی کردیم، کلمه not را از آن لیست حذف می‌کنیم.[۱۵]

```
import spacy
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

nlp = spacy.load('en_core_web_sm')
stopwords = nlp.Defaults.stop_words
sid = SentimentIntensityAnalyzer()

sentance = """I do not wish them well or happy."""
sentance = ' '.join([word for word in sentance.split() if not word in stopwords])

sid.polarity_scores(sentance)
```

✓ 0.4s

```
{'neg': 0.0, 'neu': 0.135, 'pos': 0.865, 'compound': 0.7506}
```

شکل ۲۳-۴ - تحلیل احساسات اشتباه در نتیجه‌ی حذف شدن not از جمله

٤-٨ تحلیل احساسات

تحلیل احساسات مرحله آخر پردازش روی توییت‌ها است. در این مرحله، تمام متن‌ها به شکلی که میخواستیم تبدیل شده‌اند و آماده تحلیل احساس هستند. برای اینکار، از دو شیوه میتوانستیم استفاده کنیم:

- تعلیم دادن یک مدل از ابتدا: تعلیم دادن فرایندی است که تعداد داده‌های آماده بسیاری را می‌طلبد. دلیل انتخاب این شیوه معمولا عدم وجود مدل‌هایی با کارکرد مطلوب در حوزه موردنظر می‌باشد. به عنوان مثال اگر شخصی بخواهد تحلیل احساس را برای یک محصول خاص انجام دهد، میتواند شبکه عصبی را فقط برای تحلیل نظرات درباره آن محصول آموزش دهد.
- استفاده از مدل‌های آماده: وقتی که مدلی عملکرد مطلوبی در حوزه کاری ما دارد، نیازی به تعلیم مدل جدید نیست و می‌توان از همان مدل استفاده کرد. خوشبختانه کتابخانه vadersentiment روی شبکه‌های اجتماعی آموزش دیده و نیازهای پروژه را برطرف می‌کرد. پس از همین روش استفاده کردیم.

روش کار کتابخانه vadersentiment بدین صورت است که با دریافت متن، سه نمره مجزا بین ۰ تا ۱ تحت عناوین مثبت (pos)، منفی (neg) و خنثی (neu) به آن می‌دهد. باید توجه داشت که مجموع این سه نمره ۱ خواهد شود. در نهایت نمره نهایی را تحت عنوان مرکب (compound) با اعمال عملیات‌های مناسب روی سه نمره‌ی اولیه محاسبه می‌کند. طبق توضیحات سازندگان این کتابخانه، نمره‌های بین ۰.۰۵- تا ۰.۰۵ را می‌توان خنثی درنظر گرفت.

از آنجایی که در کتابخانه pandas قابلیت اعمال تابع روی ستون‌های dataframe وجود دارد، برای تحلیل احساس، یک تابع جداگانه نوشتیم تا در ادامه با روش مناسب از آن استفاده کنیم. از آنجایی که نمره compound ، با توجه به سه نمره‌ی دیگر محاسبه شده، کافی است همان نمره را ذخیره و مورد تحلیل قرار دهیم. باید توجه داشت که روی ورودی‌های این تابع، قبلا توسط تابع clean_text عملیات پیش‌پردازش انجام شده‌است.

```
sid = SentimentIntensityAnalyzer()

def get_sentiment(tweet : str):
    return sid.polarity_scores(tweet)['compound']
```

شکل ۲۴-۴ - تابع تحلیل احساس

۴-۹ اعمال تحلیل احساس روی توییت‌ها

در این مرحله تمام توابع آماده هستند و فقط کافی است تا آن‌ها را روی دیتاهایمان اعمال کنیم. همانطور که گفته شد، کتابخانه pandas قابلیت اعمال توابع را روی ستون‌های داده دارا می‌باشد. برای اینکار کافی است با اعمال تابع apply روی ستونی که مدنظر داریم و نوشتن نام تابع مورد نظر به عنوان آرگومان ورودی به آن، اینکار را انجام دهیم.

```
for i in range(len(dataFrames)):
    dataFrames[i]['Clean text'] = dataFrames[i]['Text'].apply(clean_text)
    dataFrames[i]['Sentiment'] = dataFrames[i]['Clean text'].apply(get_sentiment)
    dataFrames[i]['Clean text'] =  dataFrames[i]['Clean text'].apply(lambda x: re.sub('[^a-z]', ' ', str(x)))
    dataFrames[i].fillna('', inplace=True)
```

شکل ۲۵-۴ - اعمال توابع روی داده‌ها

از آنجایی که ۳۵ دیتاست هرکدام شامل ۱۰۰۰۰ توییت داریم، یک حلقه می‌نویسیم تا عملیات روی تمام آن‌ها انجام شود. این کار در چهار بخش انجام گرفت:

- ستونی به نام Clean text ایجاد کردیم که محتوای آن، شامل داده‌های پیش‌پردازش شده‌ی توییت‌ها در ستون Text بود. این‌کار با اعمال تابع clean_text انجام شد و مواردی که در بخش‌های پیشین درمورد آن‌ها توضیح داده شد را روی متن توییت‌ها اعمال شدند.
- ستونی به نام Sentiment که محتوای آن، نمره compound تابع تحلیل احساس به متن ورودی است. در واقع در این ستون، روی محتوای ستون Clean text تحلیل احساس انجام می‌دهیم و نمره آن را قرار می‌دهیم.
- حالا که تابع تحلیل احساس با توجه کلمات، ایموجی‌ها و شکلک‌ها نتیجه‌گیری خود را انجام داده، بهتر است که تمام کاراکترهای خاص را از ستون Clean text حذف کنیم. با اینکار علائم نشانه‌گذاری، اعداد، ایموجی‌ها و... را حذف کرده و فقط کلمات و حروف را نگه می‌داریم که در بخش‌های بعدی صرفا روی کلمات بتوانیم تحلیل انجام دهیم.
- در این قسمت، دیتاست جدید آماده تحلیل است. حالا فقط کافی است مواردی که NA و یا NaN هستند را با نوع رشته خالی ("") ، پر کنیم زیرا وجود این موارد در هنگام تحلیل، باعث بروز خطا در اجرای برنامه می‌شود.

۴-۱۰ ذخیره کردن داده‌های تحلیل شده

از آنجایی که اعمال توابع که در بخش پیشین به آن پرداختیم مدت زمان زیادی (بین ۳۰ تا ۶۰ دقیقه بسته به سرعت پردازش سیستم) به طول می‌انجامد، می‌بایست که آن‌ها را روی سیستم ذخیره کنیم تا در صورت وجود خطا در برنامه و لزوم شروع مجدد، این داده‌ها را از دست ندهیم. روند ذخیره کردن با بخش‌های پیشین تفاوتی نداشته و شامل تعیین مسیر و نام و استفاده از تابع to_csv می‌باشد. برای خواندن دیتاها نیز از کتابخانه glob و تابع read_csv استفاده کردیم که در شکل مشخص است و در بخش‌های پیشین به تفصیل درباره آن‌ها صحبت کردیم.

store processed data in the system

```
for i in range(len(dataFrames)):
    path=r"C:\Users\pouria\Desktop\project\clean-sentiment\\"
    filename = path + (dataFrames[i]['Datetime'].iloc[[0]]).to_string()[5:15] + '.csv'
    dataFrames[i].to_csv(filename, sep=',', index=False)
```

read stored data from system

```
path = r'C:\Users\pouria\Desktop\project\clean-sentiment'
csv_files = glob.glob(os.path.join(path, "*.csv"))

dataFrames = []
for f in csv_files:

    df = pd.read_csv(f)
    dataFrames.append(df)
```

شکل ۴-۲۶ - ذخیره کردن و سپس خواندن دیتاها در سیستم

۵ فصل ۵ – مصورسازی و تحلیل یافته‌ها

در قسمت قبل روی تمام داده‌ها ابتدا پیش‌پردازش کرده و سپس تحلیل احساس انجام دادیم و ذخیره کردیم. حال که دیتاها را آماده کرده‌ایم، می‌توانیم آن‌ها را تحلیل کنیم. از آنجایی که مصورسازی در درک نتایج موثر است، نتایج را در هر قسمت با شیوه مناسب مصور کرده و تحلیل می‌کنیم.

۵-۱ مشخص کردن پرتکرارترین کلمات

از آنجایی که نتیجه‌گیری‌های این پروژه بر مبنای مصورسازی بود، نیاز بود تا واژه‌های پرتکرار را به صورت مصور نمایش دهیم. برای این کار از کتابخانه wordcloud استفاده کردیم. از آنجایی که واژه biden مبنای جستجو برای استخراج توییت‌ها بود، واضح است که در تمام توییت‌ها یافت می‌شود و به همین دلیل نیازی به وجود این واژه وجود ندارد. همچنین کلمه joe را نیز که نام آقای biden است نیز از واژه‌ها حذف می‌کنیم. واژه دیگری که وجودش ضرورتی ندارد، amp است. این کلمه در نتیجه استفاده از علامت & در توییت دریافت می‌شود که به عملکرد توییتر مربوط است و برای تحلیل نیز نیازی به آن وجود ندارد. برای اعمال این محدودیت‌ها تابعی نوشتیم که این ۳ کلمه را از جمله حذف کند که در شکل مشخص است.

```
def remove_unwanted_words(text : str):
    unwanted_words = ['biden', 'joe', 'amp']
    text = ' '.join(word for word in text.split() if not word in unwanted_words)
    return text
```

شکل ۵-۱ - تابعی برای حذف کردن ۳ واژه بدون کاربرد در تحلیل

از آنجایی که مصورسازی کلمات پرتکرار را چندین بار انجام دادیم، نیاز بود تا تابعی برای این کار بنویسیم. در این تابع با دریافت متن به عنوان آرگومان ورودی، تابع remove_unwanted_words را روی آن اجرا می‌کنیم. نحوه ساخت اَبرِ کلمات[۵۰] پرتکرار در کتابخانه wordcloud پیاده سازی شده‌است و با دریافت متن در تابع generate ، این کار انجام می‌شود. تعداد کلمات نیز بر مبنای ابعادی که به عنوان آرگومان ورودی به شیء ساخته شده از کلاس می‌دهیم، تعیین می‌شود. نمایش تصویر ساخته شده نیز توسط کتابخانه matplotlib انجام می‌شود که در بخش‌های پیشین توضیح دادیم که اساس مصورسازی در پایتون است.

[۵۰] Word cloud

```
def tweets_wordcloud(tweets_text : str):

    tweets_text = remove_unwanted_words(tweets_text)

    wordcloud = WordCloud(width = 800, height = 800,
                          background_color ='white',
                          min_font_size = 10).generate(tweets_text)

    plt.figure(figsize = (8, 8), facecolor = None)
    plt.imshow(wordcloud)
    plt.axis("off")
    plt.tight_layout(pad = 0)

    plt.show()
```

شکل ۲-۵ - تابع ساخت ابر کلمات پرتکرار

برای امکان تحلیل بهتر، ساخت ابر کلمات پرتکرار را برای هر ۳ ماه سال ۲۰۲۱ به‌طور جداگانه انجام می‌دهیم. از آنجایی که بسیاری از کلمات در همه ۳ ماه‌ها بکار خواهند رفت، تابعی برای دریافت کلمات پرتکرار به‌طور جداگانه نیز درست می‌کنیم تا در نهایت با اعمال عملیات مناسب، کلماتی که انحصارا فقط در یکی از ۳ ماه‌ها پرتکرار بوده‌اند را دریافت کنیم. لیست mow که مخفف most accured words می‌باشد، در هر مرحله با لغات پرتکرار ۳ ماهی که در حال دریافت آن هستیم، کامل‌تر می‌شود. این لیست در نهایت ۴۰۰ عضو در خود دارد که در ادامه لغات مناسب را از آن دریافت خواهیم کرد.

```
def most_occured_words(text : str):
    split_it = text.split()
    Counters_found = Counter(split_it)
    most_occur = Counters_found.most_common(100)

    most_accured_list = []
    for word_tuple in most_occur:
        most_accured_list.append(word_tuple[0])

    return most_accured_list

mow = []
```

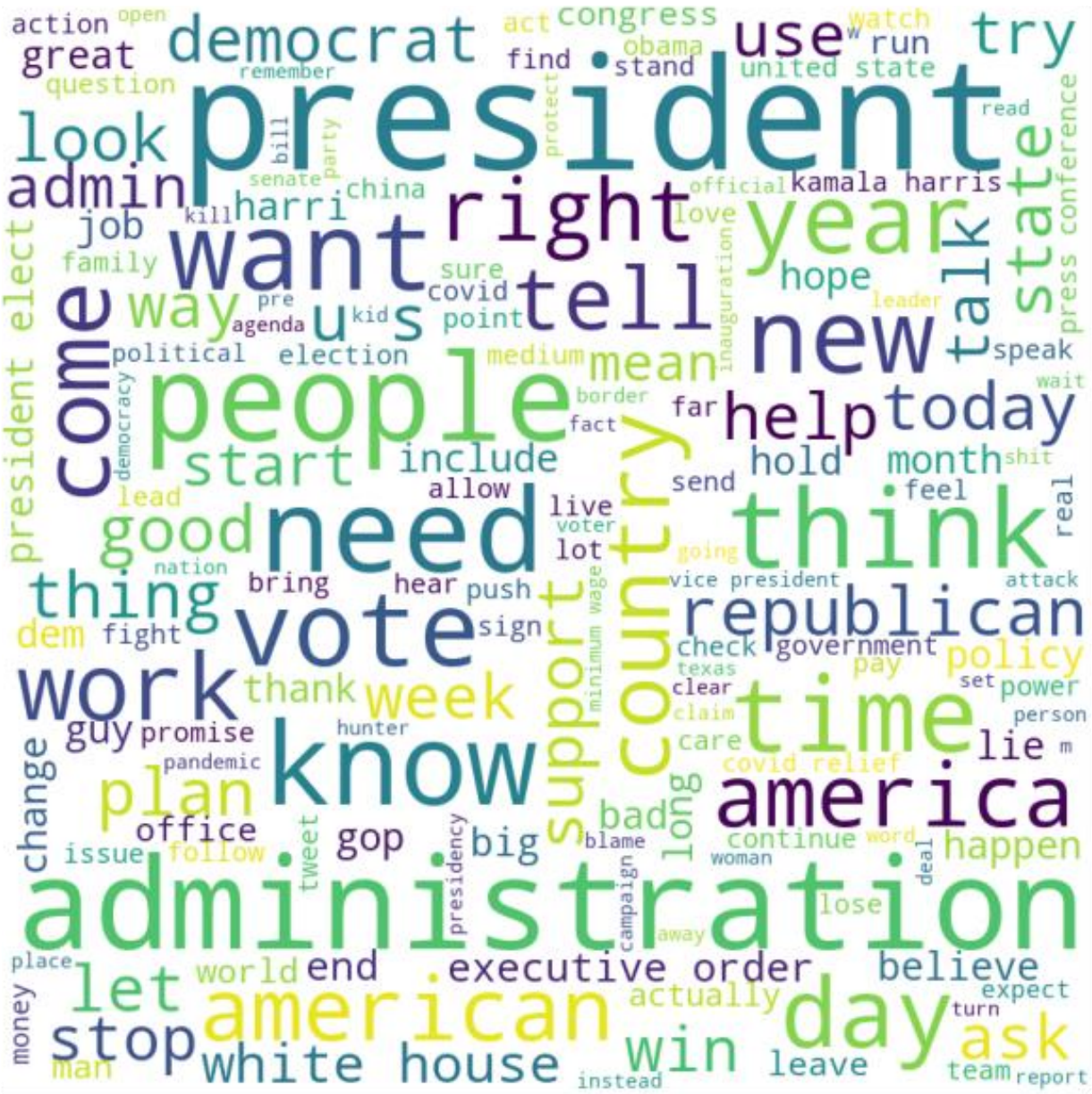
شکل ۳-۵ - تابعی برای دریافت ۱۰۰ کلمه پرتکرار در متن

۵-۱-۱ ابر کلمات در سه ماه اول سال ۲۰۲۱

پس از اینکه توابع مورد نیاز برای ساخت ابر کلمات پرتکرار را آماده کردیم، نوبت به پیاده سازی آن‌ها است. قطعه کد مربوطه برای سه ماه اول سال به‌صورت زیر است. از آنجایی که این فرایند برای سه ماه‌های بعدیِ سال نیز تقریبا مشابه است، برای حفظ کیفیت گزارش از گذاشتن آنها خودداری کردیم.

```
text = ''
for i in range(len(dataFrames)):

    df_month = (dataFrames[i]['Datetime'].iloc[[0]]).to_string()[10:12]

    if df_month == '01' or df_month == '02' or df_month == '03':
        for tweet in dataFrames[i]['Clean text']:
            text += " ".join(str(tweet).split())

mow += most_occured_words(text)
tweets_wordcloud(text)
```

شکل ۴-۵ – کد ساخت ابرکلمات برای سه ماه اول سال ۲۰۲۱

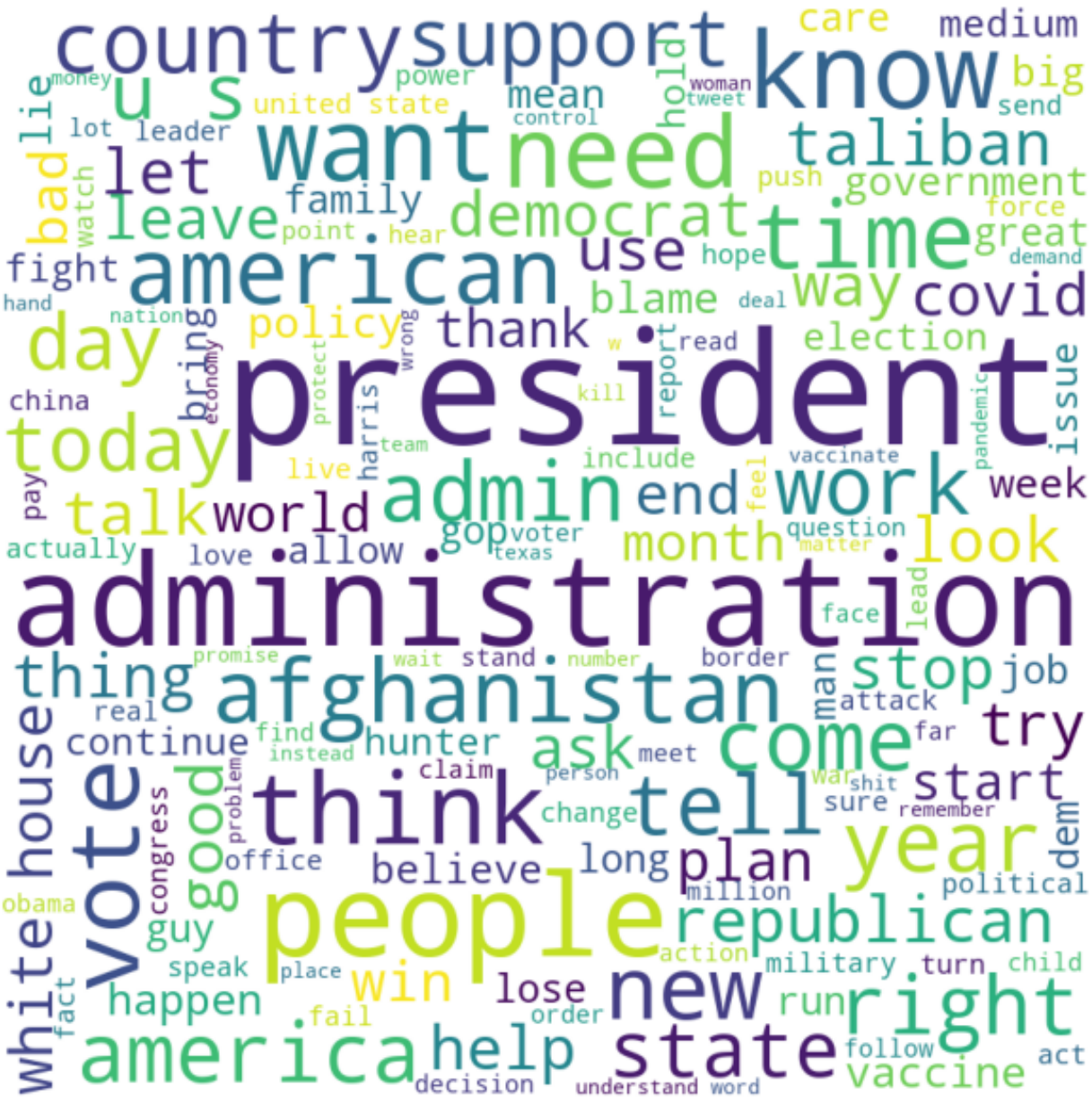


شکل ۵-۵ - ابر کلمات سه ماه اول سال ۲۰۲۱

۲-۱-۵ ابر کلمات در سه ماه دوم سال ۲۰۲۱

شکل ۶-۵ - ابر کلمات سه ماه دوم سال ۲۰۲۱

۳-۱-۵ ابر کلمات در سه ماه سوم سال ۲۰۲۱



شکل ۷-۵ - ابر کلمات سه ماه سوم سال ۲۰۲۱

٤-١-٥ ابر کلمات در سه ماه چهارم سال ۲۰۲۱

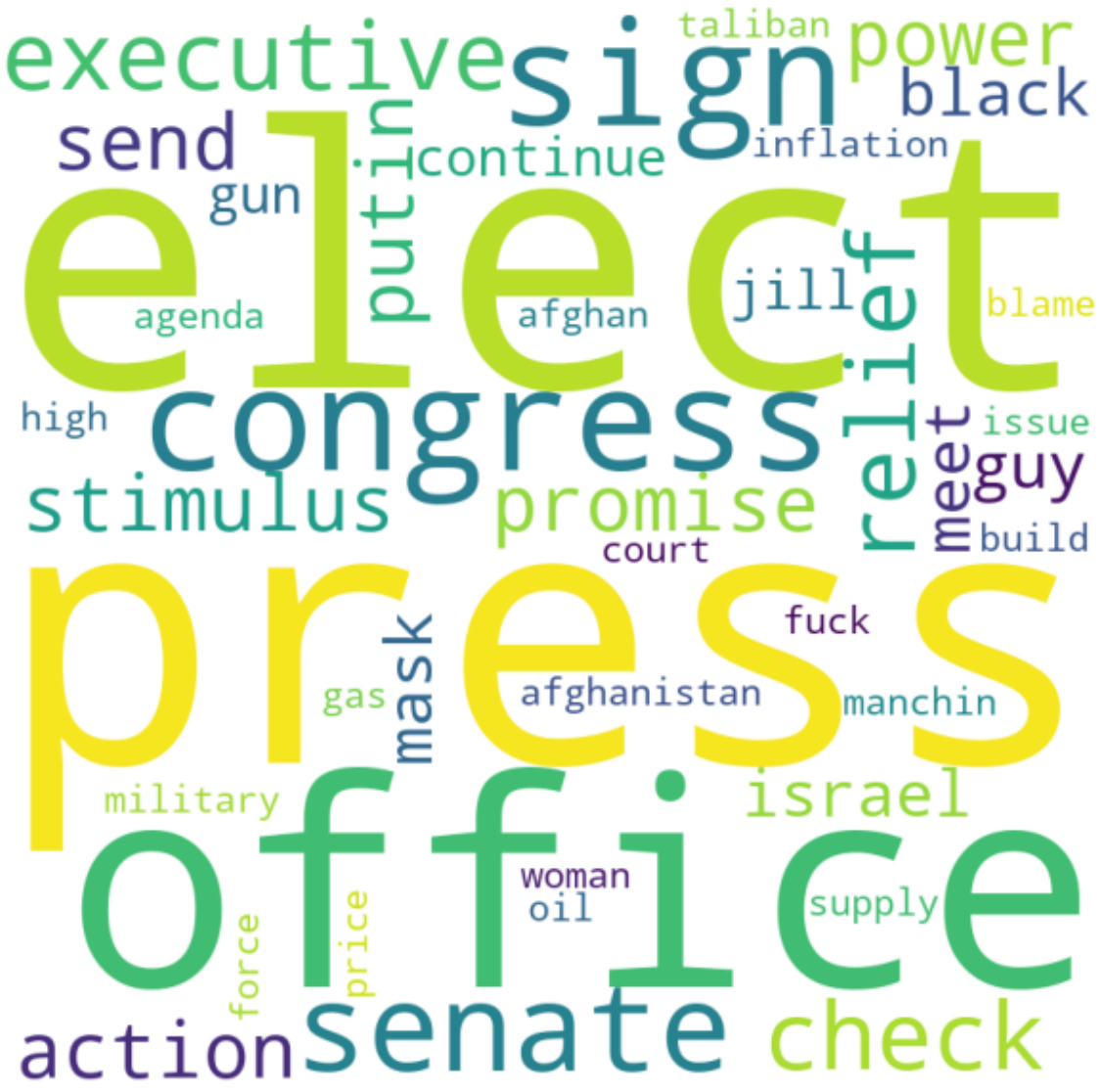
شکل ۸-۵ - ابر کلمات سه ماه چهارم سال ۲۰۲۱

۵-۱-۵ ابر کلماتی که فقط در یکی از سه ماه‌ها بودند

پیش از آنکه شروع به ساخت ابر کلمات کردیم، لیستی به نام mow ساختیم و در طی این فرایند، آن را کامل کردیم. حال نیاز است تا کلماتی که انحصارا فقط در یکی از سه ماه‌ها پرتکرار بودند را نگه داریم. برای اینکار از قطعه کد زیر و تابع counter از کتابخانه collections استفاده کردیم.

```
unique_list = [key for key in Counter(mow).keys() if Counter(mow)[key]==1]
tweets_wordcloud(' '.join([word for word in unique_list]))
```

شکل ۹-۵ - قطعه کد دریافت کلماتی که فقط در یکی از سه ماه‌های سال پرتکرار بوده‌اند



شکل ۱۰-۵ – ابر کلماتی که فقط در یکی از ۳ ماه‌های سال ۲۰۲۱ پرتکرار بوده‌اند

با بررسی این شکل، می‌توان دریافت که مسئله افغانستان صرفا در یک برهه زمانی خاص برای کاربران اهمیت داشته‌است. همچنین تعاملات ایالات متحده با روسیه نیز مطرح گشته است. درگیری‌های محله شیخ جراح در بیت المقدس هم در زمان وقوع، مورد توجه کاربرانی که درباره biden توییت کرده‌اند واقع شده‌است.

۵-۱-۶ ابر کلمات توییت‌های مثبت و منفی

در این قسمت به بررسی توییت‌هایی که احساساتشان مثبت و منفی بوده به‌طور جداگانه و با رویکرد مقایسه‌ای می‌پردازیم. برای اینکار باید در تمام دیتاست‌ها متن توییت‌های مثبت (نمره تابع تحلیل احساس به آن‌ها بالاتر از ۰.۰۵ بوده) و متن توییت‌های منفی (نمره تابع تحلیل احساس به آن‌ها پایین‌تر از ۰.۰۵- بوده) را جداگانه استخراج کنیم. برای اینکار از قطعه کد زیر استفاده می‌کنیم. از آنجایی که بسیاری از کلمات به‌طور مشترک در آن‌ها به‌کار رفته است، لازم بود تا برای امکانِ تحلیل بهتر نتایج، کلمات مشترک را حذف کنیم.

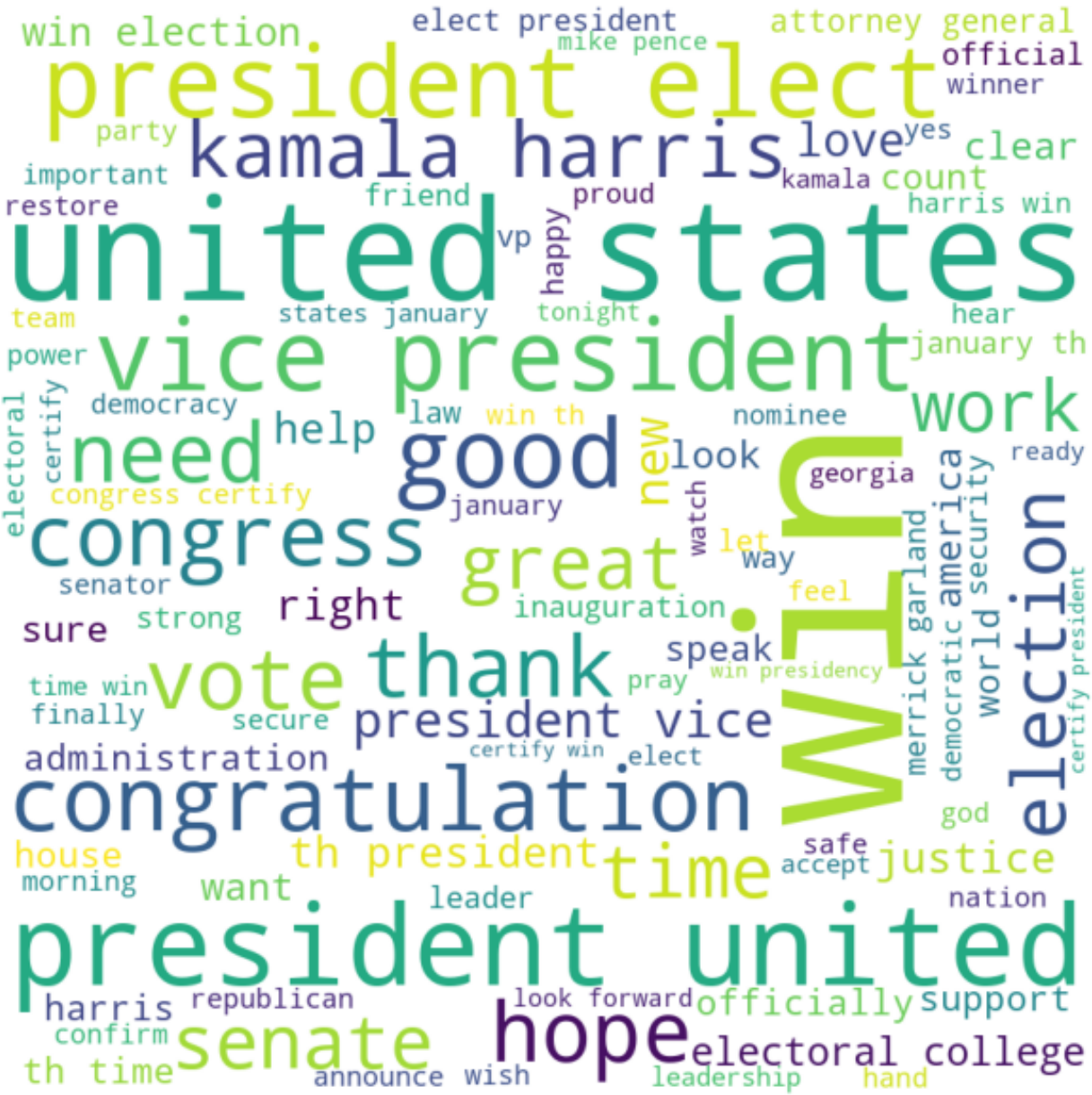
```
pos_text = ''
neg_text = ''

for i in range(len(dataFrames)):

    for j in range(len(dataFrames[i])):
        if dataFrames[i].iloc[j]['Sentiment'] > 0.05:
            pos_text += dataFrames[i].iloc[j]['Clean text']

        elif dataFrames[i].iloc[j]['Sentiment'] < -0.05:
            neg_text += dataFrames[i].iloc[j]['Clean text']

    break
```

✓ 2.3s

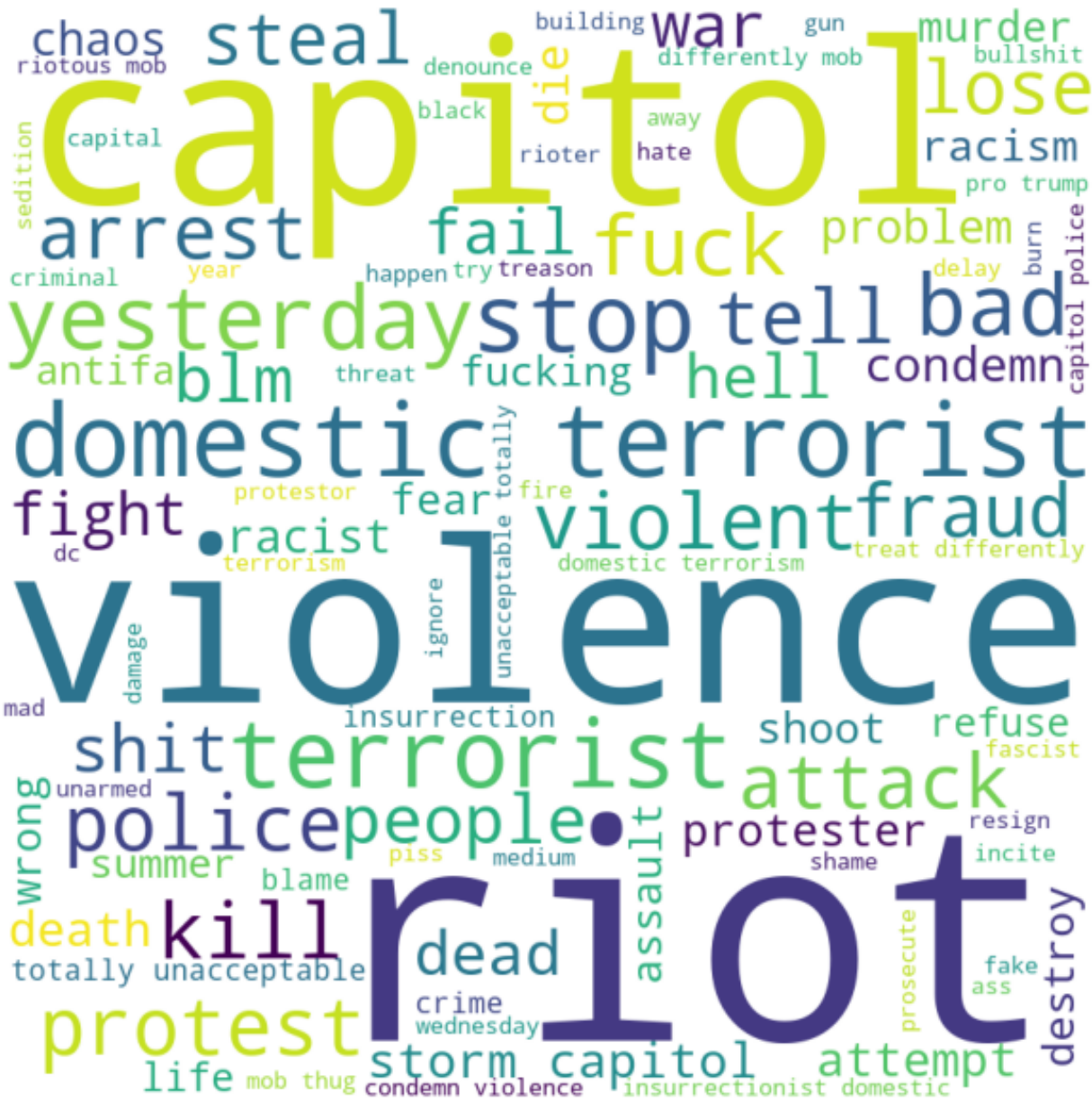
```
posetive_words_list = pos_text.split()
negative_words_list = neg_text.split()

for word in posetive_words_list[:]:
        if word in negative_words_list:
            posetive_words_list.remove(word)
            negative_words_list.remove(word)

pos_text = ' '.join([pos_word for pos_word in posetive_words_list])
neg_text = ' '.join([neg_word for neg_word in negative_words_list])
```

✓ 18.1s

شکل ۱۱-۵ - دریافت کلمات مثبت و منفی به‌طور جداگانه



شکل ۱۲-۵ - ابر کلمات مثبت



شکل ۱۳-۵ - ابر کلمات منفی

با مقایسه ابر کلمات مثبت و منفی میتوان به نکات زیر پی برد:

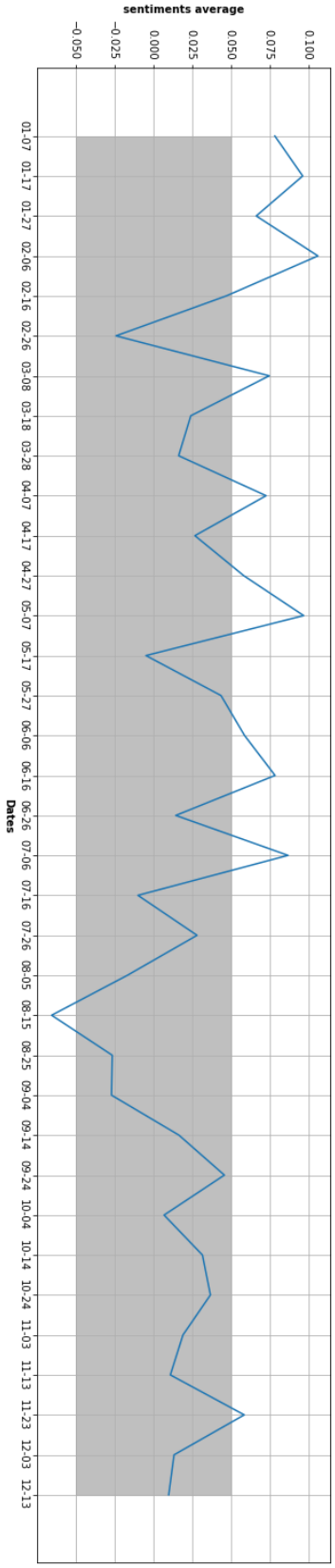
- در توییت‌های مثبت، نام اشخاص بیشتر به‌کار رفته است.
- توییت‌های مثبت به موضوعات داخلی آمریکا بیشتر پرداخته‌اند.
- توییت‌های منفی شامل تعداد زیادی الفاظ نامناسب هستند.
- توییت‌های منفی به مسائل بحث برانگیزی مثل BLM ، پلیس آمریکا و شورش‌های پایتخت پرداخته است.

۵-۲ نمودار میانگین احساسات در طول سال ۲۰۲۱

در مراحل پیشین به وسیله تابع تحلیل احساس، عددی برای احساسات کلی هر توییت مشخص کردیم. در این مرحله نتایج بدست آمده را تحلیل خواهیم کرد. برای درک بهتر، با استفاده از matplotlib نتایج مصور شده و مورد بررسی قرار می‌گیرند. در نموداری که رسم خواهد شد، محور x ، تاریخ ارسال توییت‌ها است که به فرمت MM-DD روی محور قرار می‌گیرند. از آنجایی که سالِ ارسال همه توییت‌ها ۲۰۲۱ بود، در نتایج نیازی به چاپ کردن سال نبود. در محور y ، میانگین احساسات هر دیتاست که در بازه‌های ۱۰ روزه در طول سال دریافت شده بودند را قرار دادیم. این‌کار با اعمال تابع mean روی ستون Sentiment انجام شد. همانطور که پیشتر بیان شد، نمرات بین ۰.۰۵- تا ۰.۰۵ را می‌توان خنثی درنظر گرفت بنابراین ناحیه بین این دو مقدار در نمودار را با رنگ خاکستری مشخص میکنیم. در انتها نیز برای دیدِ بهتر، از تابع grid برای خط‌کشی نمودار استفاده کردیم.

```
sentiments = []
dates = []
for i in range(len(dataFrames)):
    sentiments.append(round(dataFrames[i]['Sentiment'].mean() , 5))
    dates.append(dataFrames[i]['Datetime'].iloc[0][5:10])

plt.figure(figsize=(25, 5))
plt.plot(dates, sentiments)
plt.fill_between(dates, -0.05 , 0.05, alpha = 0.5 , color = 'grey')
plt.grid()
```

شکل ۱۴-۵ - رسم نمودار میانگین احساسات



شکل ۱۵-۵ - نمودار تغییرات میانگین احساسات کاربران در طول سال ۲۰۲۱

همانطور که در نمودار مشاهده می‌شود، در بیشترِ مقاطع سال احساسات کاربران خنثی یا مثبت بوده‌است. تنها بازه‌ای از سال که احساسات منفی بوده در تاریخ ۱۵-۰۸ (۶ تا ۱۵ آگوست) می‌باشد. نکته مهم اینجاست که احساسات کاربران در سه بازه‌ی قبلی نیز در حال منفی شدن بود. این نتایج با اتفاقات سال ۲۰۲۱ کاملا تطابق دارد زیرا پیش‌روی گسترده طالبان در افغانستان و در نهایت کنترل کامل شهر کابل در تاریخ ۱۵ آگوست اتفاق افتاد. در واقع در این تاریخ که "سقوط کابل[۵۱]" نام گرفت، حتی طرفداران آقای بایدن نیز به انتقاد از سیاست‌های او برای خروج از افغانستان پرداختند.

مثبت‌ترین بازه‌ها در اوایل سال بوده که بنظر می‌رسد به دلیل شروع دوره ریاست جمهوری بایدن (وی در تاریخ ۲۰ ژانویه سوگند خورد) بوده‌اند. اما احساسات مثبت، ناگهان بین ۷ تا ۲۶ فوریه شیب نزولی گرفته‌اند که بنظر می‌رسد به دلیل ایجاد لایحه بهبود مهاجرت[۵۲] توسط وی بود. هدف این لایحه، اعطای شهروندی به مهاجران غیرقانونی در ایالات متحده آمریکا بود. گفتنی است که "بحران مهاجران[۵۳]" یکی از بحث‌های داغ سال ۲۰۲۱ بود که همچنان نیز در جریان است.

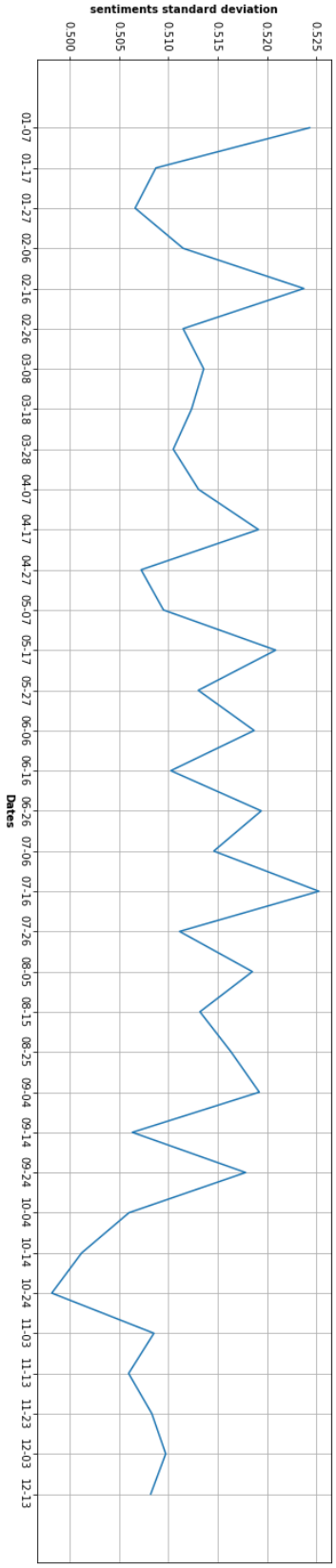
۳-۵ نمودار انحراف معیار احساسات در طول سال ۲۰۲۱

انحراف معیار نشان دهنده‌ی این است که به‌طور متوسط، داده‌ها چقدر با میانگین اختلاف دارند. برای تولید این نمودار، قطعه کدی که استفاده شد تقریبا مشابه قطعه کد قسمت قبل است تنها با این تفاوت که به‌جای تابع mean از تابع std استفاده کردیم. درضمن نیاز به هایلایت کردن نواحی نمودار نیز وجود نداشت. همانطور که مشاهده می‌شود، تغییرات انحراف معیار بین ۰.۴۹۸ و ۰.۵۲۵ می‌باشد. با توجه به بازه‌ی تغییرات داده‌ها (احساسات) که از ۱- تا ۱ بود، تغییرات انحراف معیار بسیار شدید است که این مسئله میتواند مهر تاییدی بر ادعای دودستگی مردم آمریکا در مسائل سیاسی باشد.

[۵۱] Fall of Kabul

[۵۲] Immigration reform bill

[۵۳] Immigration crisis



شکل ۱۶-۵ - نمودار تغییرات انحراف معیار احساسات کاربران در طول سال ۲۰۲۱

٦ فصل ۶ – نتیجه‌گیری و پیشنهادها

٦-١ نتیجه‌گیری

همانطور که در فصل پیشین مشاهده شد، نتایج بدست آمده با اتفاقات دنیای واقعی همبستگی خوبی دارند و می‌توان گفت که نتایج تحلیل احساس، قابل اتکا هستند. لازم به ذکر است تحلیل تمام نتایج در فصل پیشین انجام شد و نتیجه‌گیری که می‌شود از آن‌ها داشت این است که شبکه‌های اجتماعی می‌توانند نمود نسبتا دقیقی از جامعه تلقی شوند. همچنین به‌نظر می‌رسد در عصر حاضر، الگوریتم‌های هوش مصنوعی قدرت بالایی پیدا کرده‌اند و با توجه به نیازها، می‌توان از آن‌ها بهره برد.

٦-٢ پیشنهادها

نتیجه‌گیری‌های این پروژه، بر مبنای تحلیل احساسات بود و در نهایت نیز تلاش برای بررسی وجود همبستگی بین نتایج و مشاهدات دنیای واقعی صورت گرفت. پیشنهاد می‌شود برای اصلاح و یا پیشبرد این پروژه، داده‌ها از دیگر شبکه‌های اجتماعی نیز دریافت شده و بررسی گردند. همچنین می‌توان با همین رویکرد، الگوریتم‌های رایج دیگرِ NLP مانند شناسایی موجودیت‌های نام‌دار (NER) را روی داده‌ها پیاده سازی نمود.

فهرست مراجع

[١] Agarwal, Apoorv, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca J. Passonneau. "Sentiment analysis of twitter data." In *Proceedings of the workshop on language in social media (LSM 2011)*, pp. ٣٠-٣٨. ٢٠١١.

[٢] Dongo, Irvin, Yudith Cadinale, Ana Aguilera, Fabiola Martínez, Yuni Quintero, and Sergio Barrios. "Web scraping versus Twitter API: A comparison for a credibility analysis." In *Proceedings of the 22nd International Conference on Information Integration and Web-based Applications & Services*, pp. ٢٦٣-٢٧٣. ٢٠٢٠.

[٣] Roesslein, Joshua. "tweepy Documentation." *Online] http://tweepy. readthedocs. io/en/v3* ٥ (٢٠٠٩).

[٤] Makice, Kevin. *Twitter API: Up and running: Learn how to build applications with the Twitter API*. " O'Reilly Media, Inc.", ٢٠٠٩.

[٥] C. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text", *ICWSM*, vol. 8, no. 1, pp. 216-225, May 2014.

[٦] Twitter sentiment analysis using NLP techniques. Kaggle from https://www.kaggle.com/mishki/twitter-sentiment-analysis-using-nlp-techniques

[٧] Alothali, Eiman, Nazar Zaki, Elfadil A. Mohamed, and Hany Alashwal. "Detecting social bots on Twitter: A literature review." In *2018 International conference on innovations in information technology (IIT)*, pp. ١٧٥-١٨٠. IEEE, ٢٠١٨.

[٨] E. Mustafaraj, S. Finn, C. Whitlock and P. T. Metaxas, "Vocal Minority Versus Silent Majority: Discovering the Opionions of the Long Tail," *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, 2011, pp. 103-110, doi: 10.1109/PASSAT/SocialCom.2011.188.

[٩] Beng, Tan Wai, and Lim Tong Ming. "A Critical Review on Engagement Rate and Pattern on Social Media Sites." (٢٠٢٠).

[١٠] Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems 1st Edition

[١١] Shiha, Mohammed, and Serkan Ayvaz. "The effects of emoji in sentiment analysis." *Int. J. Comput. Electr. Eng.(IJCEE.)* ٩, no. ١ (٢٠١٧): ٣٦٠-٣٦٩.

[١٢] Balakrishnan, Vimala, and Ethel Lloyd-Yemoh. "Stemming and lemmatization: a comparison of retrieval performances." (٢٠١٤): ١٧٤-١٧٩.

[١٣] Srinivasa-Desikan, Bhargav. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, ٢٠١٨.

[١٤] Nothman, Joel, Hanmin Qin, and Roman Yurchak. "Stop word lists in free open-source software packages." In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pp. ٧-١٢. ٢٠١٨.

[١٥] Ladani, Dhara J., and Nikita P. Desai. "Stopword identification and removal techniques on TC and IR applications: A survey." In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pp. ٤٦٦-٤٧٢. IEEE, ٢٠٢٠.