Analiza skupa podataka *Russian Troll Tweets*

Seminarski rad

Matematički fakultet,
Univerzitet u Beogradu

Milan Ilić
Stefan Pantić

September 20, 2018

Sadržaj

1 Uvod **1**

2 Pretprocesiranje i analiza podataka **1**
2.1 Opis skupa podatka . . . 1
2.1.1 Datoteka *users.csv* . . . 1
2.1.2 Datoteka *tweets.csv* . . . 4
2.2 Pretprocesiranje . . . 6
2.3 Vizualizacija . . . 7
2.3.1 Raspodela broja objava po korisniku . . . 7
2.3.2 Vremenska raspodela objava . . . 8
2.3.3 Frekvencije korišćenih reči . . . 9

3 *Sentiment* analiza **10**
3.1 Uvod . . . 10
3.2 *Sentiment* teksta tvitova . . . 10
3.3 Simulacija glasanja na izborima . . . 13

4 Klasifikacija **14**
4.1 Uvod . . . 14
4.2 Priprema podataka za klasifikaciju . . . 15
4.2.1 Izvlacenje pet korisnika . . . 15
4.2.2 Izdvajanje najkorišćenijih reči . . . 16
4.2.3 Pravljenje *DataFrame*-a . . . 16
4.3 Klasifikacija nad spremnim skupom . . . 17

5 Klasterovanje **18**

6 Pravila pridruživanja **19**

7 Zaključak **21**

1 Uvod

Skup podataka *Russian Troll Tweets* sastoji se od dve tabele. Prva tabela, pod nazivom *users.csv*, sadrži podatke o 454 naloga sa društene mreže Twitter koji su identifikovani kao maliciozni. Druga tabela, *tweets.csv*, sadrži 203482 objava ovih korisnika nastalih u periodu oko predsedničkih izbora u Sjedinjenim Američkim Državama.
Podaci su preuzeti sa adrese: https://www.kaggle.com/vikasg/russian-troll-tweets.

2 Pretprocesiranje i analiza podataka

U ovoj sekciji ćemo se pozabaviti pretprocesiranjem i analizom navdenog skupa podataka korišćenjem programskog jezika *Python* i biblioteka:

- pandas
- numpy
- matplotlib
- wordcloud
- scikit-learn
- textblob
- i drugih

2.1 Opis skupa podatka

Za početak ćemo opisati sadržaj obe datoteke koristeći biblioteku za obradu skupova podataka *pandas*.

2.1.1 Datoteka *users.csv*

Počećemo sa opisom datoteke *users.csv*. Datoteka sadrzi 14 atributa, čiji se opisi nalaze u tabeli 1.

Naziv polja	Tip podatka	Opis
id	Numeric	Jedinstveni identifikator korisnika
location	String	Lokacija
name	String	Ime korisnika
followers_count	Numeric	Broj pratilaca
statuses_count	Numeric	Broj objava
time_zone	String	Vremenska zona
verified	Boolean	Status verifikacije
lang	String	Jezik
screen_name	String	Korisničko ime
description	String	Opis
created_at	String	Datum nastanka naloga
favourites_count	Numeric	Broj omiljenih objava
friends_count	Numeric	Broj prijatelja
listed_count	Numeric	Broj pojavljivanja na listama

Tabela 1: Opis datoteke *users.csv*

Naredni fragment koda učitava skup podataka *users.csv* i pravi instancu klase *pandas.DataFrame*, koju ćemo koristiti za obradu podataka.

```
users = pandas.read_csv('users.csv')
```

Zatim štampamo prvih pet redova da bismo stekli utisak o obliku u kom se nalaze podaci.

```
print(users.head())
```

```
           id                         location             name  \
0   18710816.0  near Utah Ave & Lighthouse an    Robby Delaware
1  100345056.0  still ⬆Block↰Corner⬇street  #Ezekiel2517🇸🇦...
2  247165706.0                   Chicago, IL  B E C K S T E R🇸🇦
3  249538861.0                           NaN     Chris Osborne
4  449689677.0                           NaN    Рамзан Кадыров

   followers_count  statuses_count                   time_zone verified lang  \
0            304.0         11484.0   Pacific Time (US & Canada)    False   en
1           1053.0         31858.0                          NaN    False   en
2            650.0          6742.0  Mountain Time (US & Canada)    False   en
3             44.0           843.0                          NaN    False   en
4          94773.0         10877.0                       Moscow    False   ru

     screen_name                                      description  \
0  RobbyDelaware  I support the free movement of people, ideas a...
1    SCOTTGOHARD  CELEBRITY TRAINER 🇸🇦#424W147th🇸🇦 #CrossfitCoach ...
2    Beckster319  Rebecca Lynn Hirschfeld Actress.Model.Writer.A...
3  skatewake1994                                                NaN
4  KadirovRussia  Пародийный аккаунт. Озвучиваю то, что политика...

                      created_at  favourites_count  friends_count  \
0  Wed Jan 07 04:38:02 +0000 2009              17.0          670.0
1  Tue Dec 29 23:15:22 +0000 2009            2774.0         1055.0
2  Fri Feb 04 06:38:45 +0000 2011            7273.0          896.0
3  Wed Feb 09 07:38:44 +0000 2011             227.0          154.0
4  Thu Dec 29 11:31:09 +0000 2011               0.0            7.0

   listed_count
0          13.0
1          35.0
2          30.0
3           1.0
4         691.0
```

Možemo primetiti da se neki simboli ne prikazuju kako treba, verovatno zbog kodiranja karaktera. Prilikom pretprocesiranja moraćemo da ih uklonimo. Takođe, moze se primetiti da su neki tvitovi napisani na ruskom, a neki na engleskom (što se može videti iz kolone **lang**). Ova informacija će nam biti od koristi prilikom pretprocesiranja.

Poslednje sto ćemo uraditi vezano za opis ovog skupa podataka jeste prebrojavanje ne-**NaN** vrednosti u kolonama i opis numeričkih podataka.

```
print(users.count())
print(users.describe())
```

```
id                  393
location            285
name                384
followers_count     384
statuses_count      384
time_zone           369
verified            384
lang                384
screen_name         454
description         339
created_at          384
favourites_count    384
friends_count       384
listed_count        384
dtype: int64
```

	followers_count	statuses_count	favourites_count	friends_count
count	384.000000	384.000000	384.000000	384.000000
mean	4730.726562	4649.156250	1747.578125	2334.039062
std	10415.236197	8522.484132	2583.843375	3524.839198
min	0.000000	1.000000	0.000000	0.000000
25%	674.000000	1135.000000	230.750000	402.000000
50%	1468.000000	1732.500000	1277.500000	1027.500000
75%	2820.750000	3012.750000	2196.750000	2667.500000
max	98412.000000	61735.000000	27181.000000	25600.000000

	listed_count
count	384.000000
mean	38.177083
std	74.507538
min	0.000000
25%	4.000000
50%	13.000000
75%	32.000000
max	691.000000

Na osnovu rezultata izvršavanja metode *count* možemo videti da se u koloni **id** nalaze **NaN**-ovi jer u koloni **screen_name** postoji više vrednosti, međutim, pošto na kolonu **id** referiše tabela *tweets.csv*, ona će biti korišćena kao primarni ključ nakon brisanja redova koji u ovom polju sadrže **NaN**. Metoda *describe* dala nam je neke osnovne statistike skupa podataka koje ćemo koristiti prilikom same analize.

2.1.2 Datoteka *tweets.csv*

Sada ćemo na isti način kao *users.csv* opisati i datoteku *tweets.csv*. Tabela sadrzi 16 atributa čiji se opisi mogu naći u tabeli 2.

Naziv polja	Tip podatka	Opis
user_id	Numeric	Jedinstveni identifikator korisnika
user_key	String	Korisničko ime
created_at	Numeric	Vreme nastanka tweeta
created_str	DateTime	Timestamp nastanka tweeta
retweet_count	String	Broj retweetova
retweeted	String	Indikator da li je tweet retweet-ovan
favourite_count	String	Broj korisnika koji su tweet dodali u listu omiljenih
text	String	Sadržaj tweet-a
tweet_id	Numeric	Jedinstvineni identifikator tweet-a
source	String	Izvor podatka
hashtags	String	Lista heštegova
expanded_urls	String	Lista linkova iz tweet-a
posted	String	Da li je tweet objavljen
mentions	String	Lista spominjanja drugih kosnika u tweet-u
retweeted_status_id	String	Jedninstveni indikator retweet-a
in_reply_to_status_id	String	Jedinstveni identifikator odgovora na tweet

Tabela 2: Opis datoteke *tweets.csv*

Ponovo ćemo napraviti instancu klase *pandas.DataFrame* i izvršićemo iste naredbe.

```
1 tweets = pandas.read_csv('tweets.csv')
2 print(tweets.head())
3 print(tweets.count())
4 print(tweets.describe())
```

```
        user_id         user_key    created_at          created_str  \
0  1.868981e+09    ryanmaxwell_1  1.458672e+12  2016-03-22 18:31:42
1  2.571870e+09  detroitdailynew  1.476133e+12  2016-10-10 20:57:00
2  1.710805e+09       cookncooks  1.487767e+12  2017-02-22 12:43:43
3  2.584153e+09     queenofthewo  1.482765e+12  2016-12-26 15:06:41
4  1.768260e+09     mrclydepratt  1.501987e+12  2017-08-06 02:36:24

   retweet_count retweeted  favorite_count  \
0            NaN       NaN             NaN
1            0.0     False             0.0
2            NaN       NaN             NaN
3            NaN       NaN             NaN
4            NaN       NaN             NaN

                                             text      tweet_id  \
0  #IslamKills Are you trying to say that there w...  7.123460e+17
1  Clinton: Trump should've apologized more, atta...  7.855849e+17
2  RT @ltapoll: Who was/is the best president of ...  8.343832e+17
3  RT @jww372: I don't have to guess your religio...  8.134006e+17
4  RT @Shareblue: Pence and his lawyers decided w...  8.940243e+17

                                          source                hashtags  \
0                                            NaN          ["IslamKills"]
1  <a href="http://twitterfeed.com" rel="nofollow...                    []
2                                            NaN                    []
3                                            NaN  ["ChristmasAftermath"]
4                                            NaN                    []

                expanded_urls  posted mentions  retweeted_status_id  \
0                          []  POSTED       []                  NaN
1  ["http://detne.ws/2e172jF"]  POSTED       []                  NaN
2                          []  POSTED       []                  NaN
3                          []  POSTED       []                  NaN
4                          []  POSTED       []                  NaN

   in_reply_to_status_id
0                    NaN
1                    NaN
2                    NaN
3                    NaN
4                    NaN
```

Prva stvar koja se može primetiti jeste da neke kolone uglavnom sadrže **NaN** vrednosti pa ćemo ih zbog toga verovatno izbaciti prilikom pretprocesiranja. Kolona **posted** na prvi pogled sadrži samo vrednosti **POSTED**, pa će i ona verovatno biti uklonjena. Od najvećeg značaja biće nam kolona **text**, kao i kolona **user_id** koja nam predstavlja strani ključ na tabelu *users.csv*.

```
user_id                 195417
user_key                203482
created_at              203461
created_str             203461
retweet_count            58083
retweeted                58083
favorite_count           58083
text                    203461
tweet_id                201168
source                   58084
hashtags                203482
expanded_urls           203482
posted                  203482
mentions                203482
retweeted_status_id      39651
in_reply_to_status_id      559
dtype: int64
            user_id    created_at  retweet_count  favorite_count  \
count  1.954170e+05  2.034610e+05   58083.000000    58083.000000
mean   1.403567e+16  1.473183e+12      39.641909       35.495085
std    1.017365e+17  1.698586e+10     290.904628      270.201692
min    1.871082e+07  1.405361e+12       0.000000        0.000000
25%    1.671235e+09  1.471270e+12       0.000000        0.000000
50%    1.856829e+09  1.476888e+12       0.000000        0.000000
75%    2.590038e+09  1.483194e+12       0.000000        0.000000
max    7.892661e+17  1.506417e+12   20494.000000    26655.000000

           tweet_id  retweeted_status_id  in_reply_to_status_id
count  2.011680e+05         3.965100e+04           5.590000e+02
mean   7.735202e+17         7.808826e+17           7.719047e+17
std    7.106134e+16         2.034659e+16           1.993880e+16
min    4.887460e+17         7.675590e+16           6.108386e+17
25%    7.654796e+17         7.768833e+17           7.627307e+17
50%    7.887575e+17         7.838451e+17           7.735847e+17
75%    8.153402e+17         7.892722e+17           7.814269e+17
max    9.126040e+17         8.927026e+17           8.009970e+17
```

Na osnovu rezultata naredbe *count* može se videti da neke kolone sadrže relativno mali broj ne-**NaN** vrednosti pa ćemo ih zbog toga odbaciti prilikom čisćenja podataka. Statistiku koju nam je dala naredba *describe* ćemo detaljno analizirati kasnije.

2.2 Pretprocesiranje

Glavno pretprocesiranje podataka će biti prikazano u ovoj sekciji, dok će pretprocesiranje za probleme u nastavku ovog rada biti objašnjeno u sklopu tih sekcija. Nakon učitavanja datoteka, jedino što ćemo ukloniti jesu svi redovi datoteke *tweets.csv* koje nemaju definisane atribute **user_id** i **tweet_id** i redovi datoteke *users.csv* koje nemaju definisan atribut **id**. Ovime smo obezbedili da je kolona **id** datoteke *users.csv* primarni ključ koji pokazuje na kolonu **user_id** datoteke *tweets.csv*. Pored toga, transformisaćemo kolone **id, user_id i tweet_id** u niske radi lakše manipulacije nad podacima.

```
1  def __init__(self, users, tweets):
2          """
3          Initialize fields with data from passed .csv files.
4
5          :param str users: Path to file with user data.
6          :param str tweets: Path to file with tweet data.
7          """
8          self._users = pd.read_csv(users, dtype={'id': 'str'}).dropna(subset=['id'])
9          self._tweets = pd.read_csv(tweets, dtype={'user_id': 'str', 'tweet_id': 'str'})
10         self._tweets = self._tweets.dropna(subset=['user_id', 'tweet_id'])
```

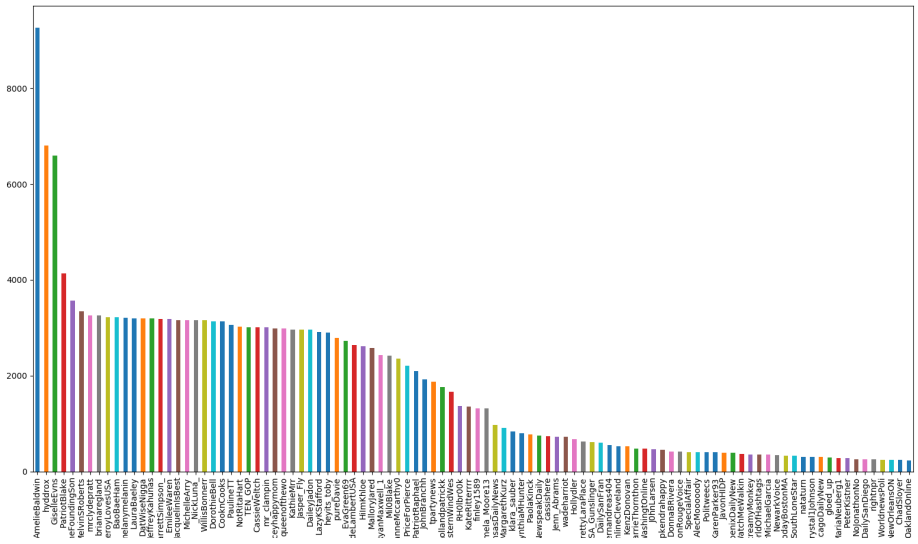
2.3 Vizualizacija

Vizualizacija spada u bitne aspekte procesa istraživanja podatka. U ovoj sekciji ćemo vizualno predstaviti tri bitne karakteristike naših podataka.

2.3.1 Raspodela broja objava po korisniku

Prvo ćemo predstaviti raspodelu broja objava za 100 korisnika koji imaju najviše objava. Odabrali smo ovaj broj iz razloga što svi ostali korisnici imaju sličan broj objava pa ne utiču previše na izgled nacrta. Naredni kod nam grupiše objave sa korisnicima koji su ih napisali, zatim vrši njihovo prebrojavanje i sortiranje po broju objava i na kraju crta *bar plot* za prvih 100 korisnika is skupa.

```
def tweets_by_user(self):
    """
    Get user IDs matched with their tweets.
    :return: DataFrame of user_id matched with tweet_id.
    """
    users = self._miner.users()[['id', 'screen_name']]
    tweets = self._miner.tweets().rename(columns={'user_id': 'id'})[['id', 'tweet_id']]
    return pd.merge(users, tweets, on='id')

def tweets_per_user(self, by='screen_name'):
    """
    Count number of tweets per user.
    :return: DataFrame with number of tweets by user.
    """
    return pd.value_counts(self.tweets_by_user()[by])

# Get tweets by user.
by_user = analyzer.tweets_per_user(by='screen_name')
by_user.head(100).plot.bar()
plt.show()
```



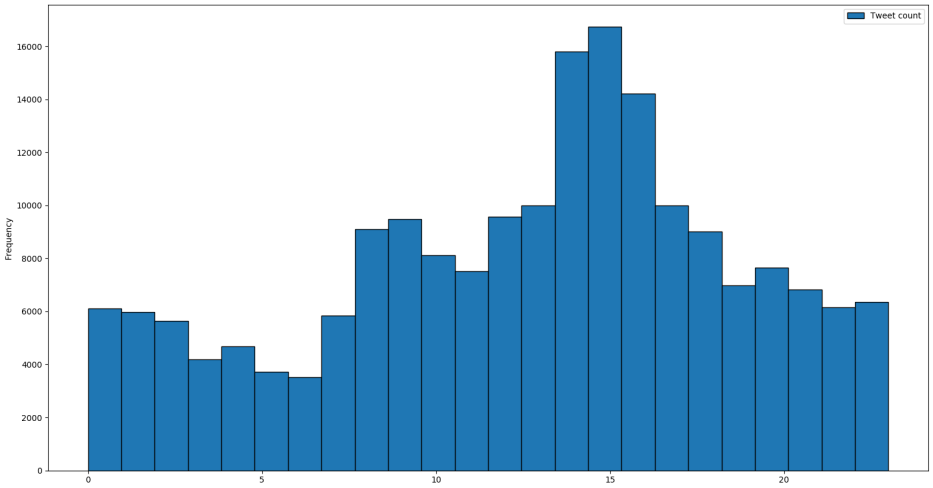
Slika 1: *Bar plot* broja objava po korisniku.

Možemo primetiti da korisnik **AmelieBaldwin** ima ubedljivo najviše objava sa 9262 objave. Prvi sledeći korisnik ima 6809 objava. Takođe možemo primetiti da veliki broj korisnika ima broj objava u intervalu 2900–3300. Nagli pad broja objava nalazi se u intervalu 900–1600, nakon kog sledi spora umerena konvergencija ka jedinici.

2.3.2 Vremenska raspodela objava

Sada ćemo prikazati raspodelu frekvencija objava po satima u toku dana, nakon čega ćemo prikazati i broj objava po danu u 2016. godini, kada su i održani predsednički izbori u SAD-u. Naredni kod nam pravi histogram sa 25 tačaka podele koristeći biblioteku *matplotlib*.

```
def time_of_tweets(self):
    """
    Get time when each tweet was posted.

    :return: DataFrame with timestamp of each tweet.
    """
    tweets = self._miner.tweets()['created_str'].to_frame()
    return tweets.applymap(lambda x: pd.Timestamp(x, tz=None))

# Get hour that each tweet was posted at.
times = analyzer.time_of_tweets().applymap(lambda x : x.hour)
times.plot.hist(bins=24, edgecolor='black', linewidth=1)
plt.show()
```



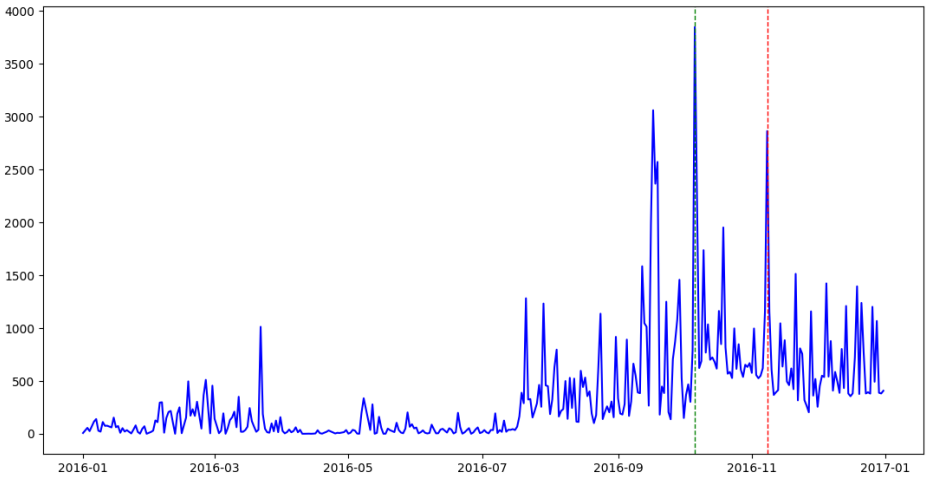
Slika 2: Histogram vremena nastanka objava po satima.

Sa histograma možemo zaključiti da je najveći broj objava nastao u popodnevnim časovima, tačnije između 14 i 17 časova, dok je najmanji broj objava bilo u ranim jutarnjim časovima, tačnije između 3 i 6 časova.

Prikažimo sada i frekvencije objava po danu.

```
def get_dates(self):
        """
        Get dates of all tweets posted in 2016.
        :return: Dates sorted in ascending order and its frequency
        """

        # Filter dates of tweets posted in 2016
        tmp = list(map(lambda x: str(x).split(' ')[0], self._miner.tweets()['created_str'].tolist()))
        dates = dict(Counter(list(filter(lambda y: str(y)[:4] == '2016', tmp))))

        # Convert dates into format for representing
        dates_only = list(dates.keys())
        dates_only = list(map(lambda x: datetime.datetime.strptime(x, '%Y-%m-%d'), dates_only))
```

```
        dates_only = mdates.date2num(dates_only)

        count_only = list(dates.values())

        # Sort dates and their count
        sorted_dates = [x for x,_ in sorted(zip(dates_only, count_only))]
        sorted_counts = [x for _,x in sorted(zip(dates_only, count_only))]

        # Return sorted dates and their count
        return sorted_dates, sorted_counts

plt.plot_date(dates, counts, 'b-')
plt.axvline(datetime.datetime(2016, 11, 8), color='red', ls='dashed', lw=1)
plt.axvline(datetime.datetime(2016, 10, 6), color='green', ls='dashed', lw=1)
plt.show()
```



Slika 3: Grafik frekvencije tvitova po danu u 2016. godini

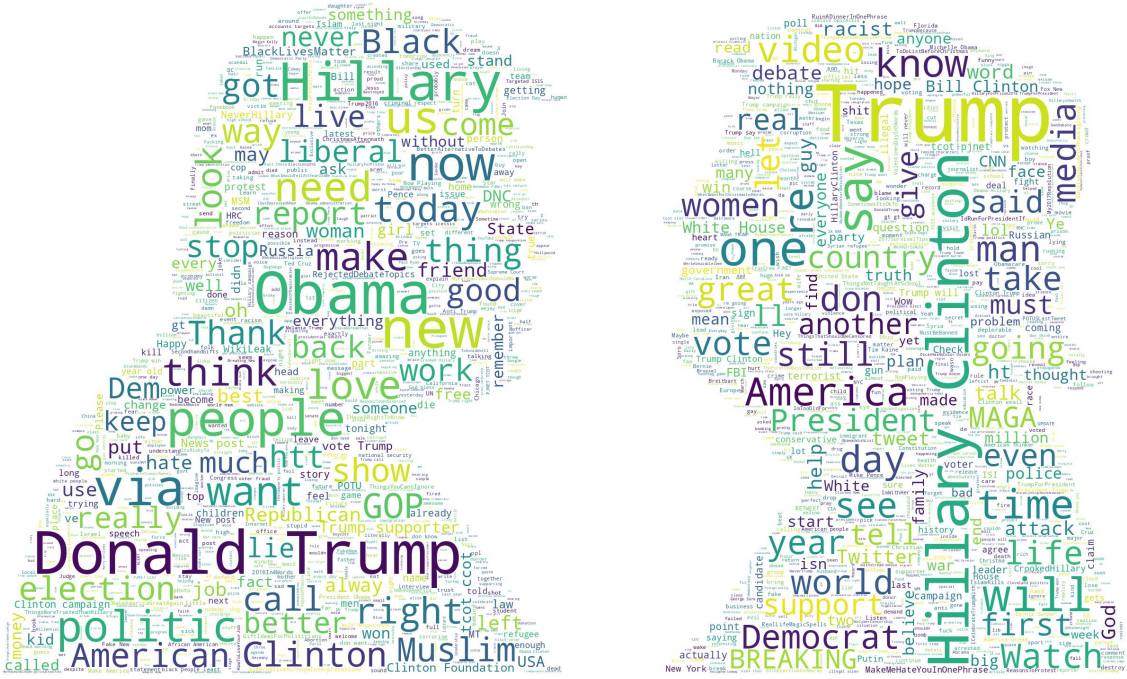
Na grafiku smo dodali i dve vertikalne linije, zelena predstavlja dan kada je došlo do incidenta sa elektronskom poštom kandidatkinje Hilari Klinton, a crvena predstavlja dan predsedničkih izbora. Upravo tih dana dolazi do naglog povećanja broja tvitova, što je i očekivano.

2.3.3 Frekvencije korišćenih reči

Poslednja statistika kojom ćemo se pozabaviti u ovoj sekciji biće vizualizacija najčešće korišćenih reči u korisničkim objavama. Zarad lepšeg prikaza istih, koristićemo biblioteku *wordcloud*, kojom ćemo prikazati reči u različitim veličinama, srazmerno njihovoj frekvenciji u objavama. Zbog velike učestalosti takozvanih *stop* reči, njih smo izbacili iz razmatranja. Kao i u prošlim sekcijama sledi kod, nakon koga i slika izlaza datog koda.

```
def word_cloud(self, outfile):
    """
    Generate word cloud with word occurrences from all tweets.

    :param outfile: Output file to write word cloud image.
    :return: None
    """
    # Get tweet text and concatenate into single string.
    tweets = self._miner.tweets()['text'].map(lambda x: TweetDataMiner.clean_text(str(x))).to_frame()
    text = ''
    for _, item in tweets.iterrows():
```

```
        text += ' ' + item['text']

    # Create set of words to skip.
    sw = set(STOPWORDS)
    sw.update(['RT', 'amp', 'https'])

    # Create background mask.
    mask = np.array(Image.open('../resource/trump_mask.jpg'))

    # Create word cloud.
    wc = WordCloud(background_color='white', max_words=2000, mask=mask, stopwords=sw)
    wc.generate(text)
    wc.to_file(outfile)
```



Slika 4: Vizualni prikaz frekvencija reči.

Vidimo da su najčešće reči uglavnom vezane za predsedničke izbore u Sjedinjenim Američkim Državama 2016. godine.

3 *Sentiment* analiza

3.1 Uvod

U ovoj sekciji ćemo izvršiti *sentiment* analizu nad našim skupom podataka. Bitno je prvo objasniti šta uopšte predstavlja *sentiment* analiza.

Sentiment analiza je računarski proces kategorizacije mišljenja izraženih u datom tekstu, posebno da bi se utvrdilo da li je odnos pisca prema određenoj temi pozitivan, negativan ili neutralan.

Za *sentiment* analizu nad našim skupom podataka koristićemo biblioteku *textblob* koja je izuzetno laka za korišćenje, s druge strane veoma moćna. Zbog mogućeg rada nad tekstualnim podacima različitog jezika, ukljucili smo tvitove napisane na bilo kom jeziku.

3.2 *Sentiment* teksta tvitova

Počnimo jednostavnim primerom korišćenja biblioteke. Napravićemo *.csv* datoteku koja za svaki tvit određuje njegovu sentimentalnost (pozitivnu, negativnu ili neutralnu).

Zaglavlje datoteke je oblika:

user_id, tweet_id, sentiment

Sledeća funkcija upravo radi gore opisan posao za nas.

```
def sentiment(self, infile=None, outfile=None):
        """
        Analyze sentiment for each tweet.

        :param str infile: Path to input file (optional).
        :param str outfile: Path to output file (optional).
        :returns pandas.DataFrame: DataFrame object with sentiment data for each user.
        """
        # If there is an infile, load into DataFrame.
        if infile and os.path.exists(infile):
            with open(infile, 'r') as sentfile:
                return pd.read_csv(sentfile, dtype={'user_id': 'str'})

        # Initialize result dict.
        df = OrderedDict([('user_id', []), ('tweet_id', []), ('sentiment', [])])

        # Iterate through selected rows.
        for index,row in self._tweets[['user_id', 'text', 'tweet_id']].dropna().iterrows():
            # Write user id and tweet id.
            df['user_id'].append(row['user_id'])
            df['tweet_id'].append(row['tweet_id'])

            # Get sentiment for tweet.
            blob = TextBlob(TweetDataMiner.clean_text(str(row['text'])))
            if 0 == blob.sentiment.polarity:
                df['sentiment'].append('neutral')
            elif blob.sentiment.polarity > 0:
                df['sentiment'].append('positive')
            else:
                df['sentiment'].append('negative')

        # Store data in DataFrame.
        df = pd.DataFrame(df)

        # Write data to csv file.
        if outfile:
            df.to_csv(outfile, sep=',', encoding='utf-8', index=False)

        # Return data frame.
        return df

miner.sentiment(infile='../resource/sentiments.csv')
```

Prikazaćemo po jedan tvit koji je obeležen pozitivno, negativno i neutralno radi provere tačnosti biblioteke.

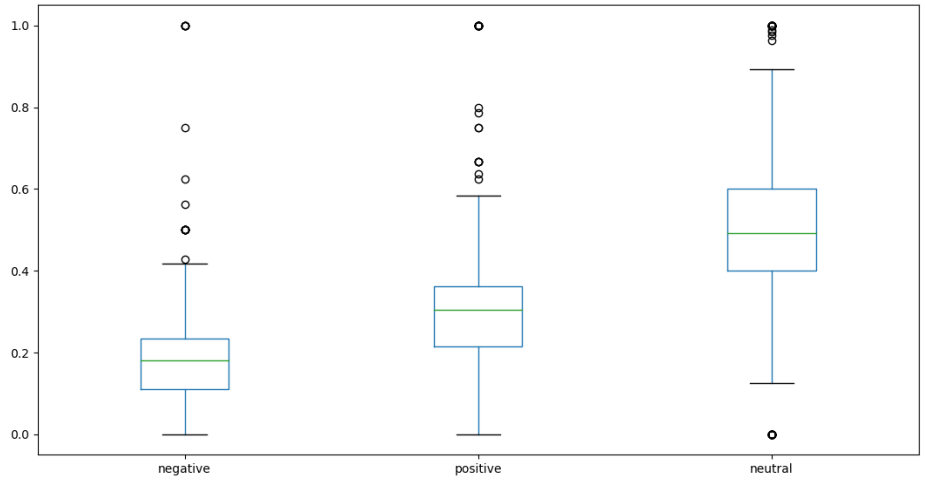
#IslamKills I don't care if you call me insensitive, or other names. "Syrian refugees" are not welcome here in U.S. - negative

Happy #2ndAmendmentDay! - positive

I can understand the guy))) https://t.co/CY4zOkNjNs - neutral

U ovoj sekciji ćemo takođe prikazati 3 *box-plot*-a za pozitivan, negativan i neutralan *sentiment* redom. Prvo je potrebno da za svakog korisnika odredimo procenat pozitivno, negativno i neutralno obeleženih tvitova i svaki ubaciti u odgovarajući niz koji će nam služiti za konstrukciju *box-plot*-a.

```
def sentiment_ratio_per_user(self, sentiments=None, infile=None, outfile=None):
        """
        Get sentiment ratio of each user.

        :param sentiments: Path to sentiments data set (optional).
        :param infile: Path to input file with data (optional).
        :param outfile: Path to output file (optional).
        :return: DataFrame with sentiment ratios for each user.
        """
        # If there is an infile, load data from it.
        if infile and os.path.exists(infile):
            with open(infile, 'r') as rfile:
                return pd.read_csv(rfile, dtype={'user_id' : 'str'})

        # Load sentiments if file is passed or initialize them otherwise
        sentiments = self._miner.sentiment(infile=sentiments) if sentiments else self._miner.sentiment()

        # Initialize result dict.
        df = OrderedDict([('user_id', []), ('negative', []), ('positive', []), ('neutral', [])])

        # Iterate through data and calculate tweet ratios.
        for user in sentiments.user_id.unique():
            df['user_id'].append(user)

            # Count sentiments.
            negative = sentiments[(sentiments.sentiment == 'negative') & (sentiments.user_id == user)].tweet_i
            positive = sentiments[(sentiments.sentiment == 'positive') & (sentiments.user_id == user)].tweet_i
            neutral = sentiments[(sentiments.sentiment == 'neutral') & (sentiments.user_id == user)].tweet_id.

            # Get total number of sentiments for user.
            total = negative + positive + neutral

            # Input ratios into result dict.
            df['negative'].append(negative/total)
            df['positive'].append(positive/total)
            df['neutral'].append(neutral/total)

        # Construct DataFrame from dict.
        df = pd.DataFrame(df)

        # Write data to csv file.
        if outfile:
            df.to_csv(outfile, sep=',', encoding='utf-8', index=False)

        # Return data frame.
        return df

sentiment_ratios = analyzer.sentiment_ratio_per_user(infile='../resource/sentiment_ratios.csv')
sentiment_ratios.plot.box()
plt.show()
```



Slika 5: *Box-plot*-ovi sentimentalne vrednosti

Vidimo da je nas *Sentiment* analizator najčešće klasifikovao tvitove kao neutrealne, dok se negativni javljaju najređe u datom skupu.

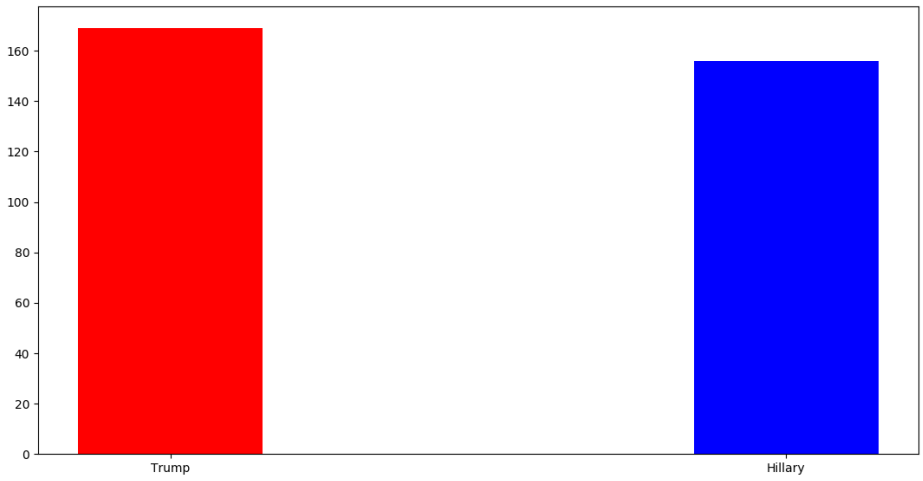
3.3 Simulacija glasanja na izborima

Naredni kod će na osnovu sentimentalne vrednosti objava korisnika vezanih za **Donalda Trampa** i **Hilari Klinton** simulirati glasanje na izborima.

Prvo ćemo odrediti procentualne vrednosti *sentiment*-a po korisniku uzimajući u obzir samo tvitove koji su vezani za kandidate na predsedničkim izborima.

```
def voting_simulation(self):
        """
        Simulates election based on tweet polarity.
        :return: Two integers representing voting results.
        """
        users = {}

        # Get tweets containing words 'Donald', 'Trump' and 'Hillary', 'Clinton'
        tweetsDonald = self._miner.filter_tweets(keywords=['Donald', 'Trump'])[['text', 'user_id']]
        tweetsHillary = self._miner.filter_tweets(keywords=['Hillary', 'Clinton'])[['text', 'user_id']]

        # Make a list containing polarity of above tweets
        for i, row in tweetsDonald.iterrows():
            if row['user_id'] not in users:
                users[row['user_id']] = [0, 0, 0, 0, 0, 0]
            response = TextBlob(TweetDataMiner.clean_text(row['text'])).sentiment.polarity
            if response < 0:
                users[row['user_id']][0] += 1
            elif response > 0:
                users[row['user_id']][1] += 1
            else:
                users[row['user_id']][2] += 1

        for i, row in tweetsHillary.iterrows():
            if row['user_id'] not in users:
                users[row['user_id']] = [0, 0, 0, 0, 0, 0]
            response = TextBlob(TweetDataMiner.clean_text(row['text'])).sentiment.polarity
            if response < 0:
                users[row['user_id']][3] += 1
            elif response > 0:
                users[row['user_id']][4] += 1
```

```
            else:
                users[row['user_id']][5] += 1

        proTrump = 0
        proHillary = 0

        # Calculate if a person is pro-Trump or pro-Hillary
        for k, v in users.items():
            posTrump = v[1]/(1 if v[0] + v[1] + v[2] == 0 else v[0] + v[1] + v[2])
            negTrump = v[0]/(1 if v[0] + v[1] + v[2] == 0 else v[0] + v[1] + v[2])
            posHillary = v[4]/(1 if v[3] + v[4] + v[5] == 0 else v[3] + v[4] + v[5])
            negHillary = v[3]/(1 if v[3] + v[4] + v[5] == 0 else v[3] + v[4] + v[5])
            if posTrump - negTrump > posHillary - negHillary: proTrump += 1
            elif posTrump - negTrump < posHillary - negHillary: proHillary += 1

        # Return number of pro-Trump and pro-Hillary accounts
        return proTrump, proHillary

proTrump, proHillary = analyzer.voting_simulation()
objects = ('Trump', 'Hillary')
y_pos = np.arange(len(objects))

plt.bar(y_pos, [proTrump, proHillary], align='center', color=['red', 'blue'], width=[0.3, 0.3])
plt.xticks(y_pos, objects)
plt.show()
```



Slika 6: Rezultati simulacije glasanje korisnika

Iz rezultata možemo zaključiti da su tvitovi iz datog skupa za nijansu naklonjeniji kandidatu Donaldu Trampu, koji bi u našoj simulaciji osvojio 169 glasova, dok bi Hilari Klinton osvojila 156.

4 Klasifikacija

4.1 Uvod

U ovom poglavlju ćemo primeniti neke klasifikacione metode nad našim podacima koristeći biblioteku *scikit-learn* koja sadrži razne korisne metode za istraživanje podataka.
Pitanje na koje ćemo pokušati da odovorimo jeste da li je moguće predvideti pisca tvita koristeći samo reči koje spominje u datom tvitu. Zbog lakšeg pretprocesiranja podataka za naš problem, uzeli smo u obzir samo 5 korisnika sa najvećim brojem napisanih tvitova, i od njih uzeli 3300

(ukupan broj tvitova koje razmatramo je 16500) nasumično izabranih tvitova, da bi naše klase imale jedank broj predstavnika. Takođe, iz datog skupa smo isključili tvitove napisane na ruskom jeziku i odlučili smo da uzmemo 2000 najkorišćenijih reči u tvitovima za klasifikaciju.
Izvršili smo dodatno pretprocesiranje izbacivši tvitove korisnika kojima kolona **lang** inije jednaka *en*.

4.2 Priprema podataka za klasifikaciju

Prvo ćemo prikazati pretprocesiranje podataka za ovaj problem.

4.2.1 Izvlacenje pet korisnika

Potrebno je naći 5 korisnika sa najvećim brojem tvitova i od njih izabrati 3300 nasumično izabranih.
To radimo sledećim dvema funkcijama:

```
def drop_user_less_3300(tweets, users):
    """
    Drops users who have less then 3300 tweets posted
    :return: tweets and users DataFrames
    """

    for i, row in users.iterrows():
        count_user_tweets[int(row['id'])] = 0

    for i, row in tweets.iterrows():
        count_user_tweets[int(row['user_id'])] += 1

    count = sorted(count_user_tweets.items(), key = operator.itemgetter(1))[::-1]
    users_more_3300 = list(filter(lambda c: c[1] > 3300, count))
    users_more_3300 = list(map(lambda c: int(c[0]), users_more_3300))

    for i, row in users.iterrows():
        if int(row['id']) not in users_more_3300:
            users.at[i, 'id'] = None

    for i, row in tweets.iterrows():
        if int(row['user_id']) not in users_more_3300:
            tweets.at[i, 'user_id'] = None

    tweets = tweets[pandas.notnull(tweets['user_id'])]
    users = users[pandas.notnull(users['id'])]

    return tweets, users
```

```
counters = [0, 0, 0, 0, 0]

def reduce_to_3300(tweets):
    """
    Reduces number of tweets of 5 with most to 3300
    :return: redused tweets DataFrame
    """

    to_delete = []
    for i, row in tweets.iterrows():
        if counters[int(row['user_id'])] == 3300:
            to_delete.append(i)
        else:
            counters[int(row['user_id'])] += 1

    return tweets.drop(tweets.index[to_delete])
```

4.2.2 Izdvajanje najkorišćenijih reči

Sledećom funkcijom se iz svih tvitova izdvaja podskup kardinalnosti 2000 najkorišćenijih reči u istim.

```
words = {}

def common_words(tweets):
    """
    Finds most common words in tweets
    :return: list of pairs of words and their count
    """

    for i, row in tweets.iterrows():
        tweets.at[i, 'text'] = clean_tweet(row['text'])
        wordList = re.sub("[^\w]", " ", row['text']).split()
        for word in wordList:
            word = word.lower()
            if len(word) >= 5:
                if word not in words.keys():
                    words[word] = 1
                else:
                    words[word] += 1

    count = sorted(words.items(), key=operator.itemgetter(1))[::-1]
    return list(map(lambda c: c[0], count))

my_words = common_words(tweets)
my_words = my_words[:2000]
```

4.2.3 Pravljenje *DataFrame*-a

Nakom skupljanja najkorišćenijih reči pravimo *DataFrame* koji će nam služiti za klasifikaciju. Zaglavlje datog *Dataframe*-a je sledećeg oblika:

user_id, reč1, reč2, reč3, ... , rečN

gde su reč1...rečN reči koje uzimamo u obzir.

Za svaki tvit se pravi vrsta u kojoj je upisana jedinica ako se ta reč nalazi u tom tvitu, u suprotnom nula.
Sledi nastavak malopređašnjeg koda:

```
for j, word in enumerate(my_words):
        tmp_words = []
        for i, row in tweets.iterrows():
            if word in re.sub("[^\w]", " ", row['text']).split():
                tmp_words.append(1)
            else:
                tmp_words.append(0)
        tweets[word] = tmp_words
```

Ostala je još jedna sitnica koju treba uraditi, a to je da preimenujemo klase, tj. korisnička imena zbog preglednije klasifikacije.

```
for i, row in tweets.iterrows():
        if row['user_id'] not in users:
            users.append(row['user_id'])
        tweets.at[i, 'user_id'] = users.index(row['user_id'])

        return tweets
```

4.3 Klasifikacija nad spremnim skupom

Sada smo spremni za klasifikaciju tvitova po korisniku. Primenićemo 2 algoritma klasifikacije nad našim skupom: metoda potpornih vektora (*LinearSVC*) i klasifikacija neuronskom mrežom (*MLPClassifier*), koje se dobro ponašaju sa velikim brojem atributa, u ovom slučaju 2000.

Naredni kod deli skup podataka na trening (70%) i test (30%) skup, a zatim pravi model za dve spomenute klasifikacije i pokušava da predvidi koji tvit je koji korisnik napisao. Nakon predviđanja ispisujemo izlaz funkcija *classification_report* i *confusion_matrix* da bismo videli koliko je naš model dobar/loš.

```
import pandas, math, operator
import numpy as np
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC, LinearSVC
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

def main():
    tweets = pandas.read_csv('forNM.csv')
    tweets = tweets.sample(frac=1).reset_index(drop=True)

    features = tweets.columns.drop(['Unnamed: 0','user_id']).tolist()

    x = tweets[features]
    y = tweets['user_id']

    x = np.array(x)
    y = np.array(y)

    X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3)

    clf = LinearSVC()
    clf.fit(X_train, y_train)
    res = clf.predict(X_test)

    print(classification_report(res, y_test))
    print(confusion_matrix(y_test, res))


    clf = MLPClassifier(solver='lbfgs', alpha=1e-5)
    clf.fit(X_train, y_train)
    res = clf.predict(X_test)

    print(classification_report(res, y_test))
    print(confusion_matrix(y_test, res))

if __name__ == '__main__':
    main()
```

Dobijamo sledeće rezultate:

```
LinearSVC:
              precision    recall  f1-score   support

         0.0       0.38      0.27      0.32      1151
         1.0       0.32      0.23      0.26      1108
         2.0       0.28      0.18      0.22      1078
         3.0       0.31      0.20      0.24      1152
         4.0       0.27      0.62      0.38      1121

avg / total       0.31      0.30      0.29      5610

[[313  87  86  66 599]
 [ 99 250 180 191 388]
 [101 191 199 156 431]
 [105 194 187 228 438]
 [207  66  67  85 696]]

MLPClassifier:
              precision    recall  f1-score   support

         0.0       0.35      0.25      0.29      1151
         1.0       0.30      0.19      0.23      1108
         2.0       0.25      0.18      0.21      1078
         3.0       0.29      0.20      0.24      1152
         4.0       0.27      0.61      0.37      1121

avg / total       0.29      0.29      0.27      5610

[[291  89 105  81 585]
 [ 98 207 180 224 399]
 [107 175 194 174 428]
 [119 169 208 231 425]
 [207  59  80  93 682]]
```

Slika 7: Rezultat funkcija *classification_report* i *confusion_matrix*

Dobili smo jako mali procenat pogođenih klasa u našem test skupu, tačnije, 30% metodom potpornih vektora i 29% neuronskom mrežom, naspram 20% koje bismo dobili nagađanjem klasa, što nam govori o tome da tvitovi korisnika imaju jako sličnu tematiku. To je i očekivano jer je dati skup podataka uglavnom vezan za predsedničke izbore u SAD-u.

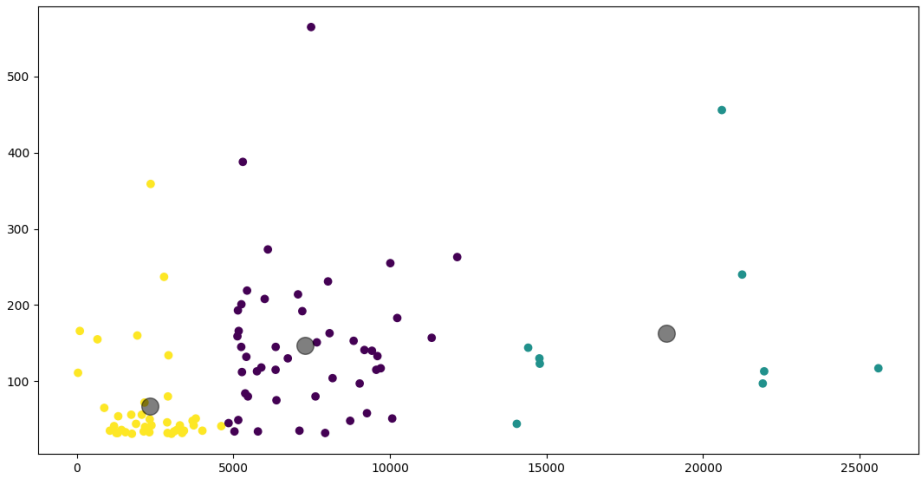
5 Klasterovanje

Klasterovanje predstavlja jednu od osnovnih tehnika istraživanja podataka. Zbog toga ćemo se u ovoj sekciji pozabaviti klasterovanjem nad naš skupom podataka. Koristićemo algoritam k-sredina koji možemo naći u biblioteci *scikit-learn* pod nazivom **KMeans**.

Predstavićemo klasterovanje u dvodimenzionalnom prostoru gde nam x osa predstavlja broj prijatelja (**friends_count**) korisnika koji imaju vise od 30 prijatelja, dok nam y osa predstavlja broj objava na listama (**listed_count**) korisnika koji takođe imaju preko 30 istih.
Za broj klastera smo izabrali broj 3 nakon pogleda na *scatter-plot* datih podataka, i sledećim algoritmom smo izvršili klasterovanje:

```
1  def cluster(self):
2          """
3          Finds clasters based on hashtags and mentions length
4          :return: np.array of two collums (friends_count and listed_count)
5          """
6          fl = np.array(self._miner.users()[['friends_count', 'listed_count']].dropna())
7          lst = []
8
```

```
        for i in fl:
            if i[0] > 30 and i[1] > 30:
                lst.append([i[0], i[1]])

        return np.array(lst)

kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

plt.scatter(X[:, 0], X[:, 1], c=y_kmeans)
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
plt.show()
```



Slika 8: Vizualni prikaz klastera

Sa slike vidimo da smo našli 3 klastera od kojih je svaki drugačije gustine i da postoje dosta *outlier*-a. Veliki broj tačaka se našao u žutom klasteru koji predstavlja male vrednosti atributa, dok se najređe susreću korisnici sa velikim vrednostima oba atributa.

6 Pravila pridruživanja

Poslednju tehniku istraživanja podataka koju ćemo iskoristiti u ovom radu jeste tehnika pravila pridruživanja, koju smo izvršili uz pomoć biblioteke *mlxtend*. Za ovaj problem izabrali smo da gledamo heštegove koji su korišćeni u tvitovima. Zbog ogromnog broja različitih heštegova razmatrali smo samo one tvitove koji sadrže više od 15 heštegova u sebi.

Za parametar minimalne podrške izabrali smo vrednost 0.2, dok smo za minimalnu pouzdanost izabrali 0.7. Sledi kod za dati problem, prvo smo filtrirali podatke, nakon toga izvršili *apriori* algoritam i pronašli česte skupove i na kraju uradili algoritam pravila pridruživanja nad čestim skupom.

```
def get_frequent(self):
        """
        Finds hashtags sets containing minimum 15 hashtags
        :return: List of lists of strings representing hashtags
        """
        tweets_hashtags = self._miner.tweets()['hashtags'].to_frame()
        hashtags = []
        for i, row in tweets_hashtags.iterrows():
```

```
            lst = row.tolist()
            if(lst[0] != '[]'):
                lst = lst[0].replace('\"', '\'')
                lst = ast.literal_eval(lst)
                if(len(lst) > 15):
                    hashtags.append(lst)

        return hashtags

freq = analyzer.get_frequent()

te = TransactionEncoder()
te_fit = te.fit(freq).transform(freq)
df = pd.DataFrame(te_fit, columns=te.columns_)

frequent_items = apriori(df, min_support=0.2, use_colnames=True)
print(frequent_items)

print(association_rules(frequent_items, metric='confidence', min_threshold=0.7))
```

	support	itemsets
0	0.203390	[LNYHBT]
1	0.245763	[NeverHillary]
2	0.203390	[TGDN]
3	0.347458	[Trump]
4	0.347458	[maga]
5	0.364407	[tcot]
6	0.203390	[LNYHBT, TGDN]
7	0.203390	[LNYHBT, maga]
8	0.203390	[LNYHBT, tcot]
9	0.203390	[TGDN, maga]
10	0.203390	[TGDN, tcot]
11	0.220339	[Trump, maga]
12	0.220339	[Trump, tcot]
13	0.245763	[maga, tcot]
14	0.203390	[LNYHBT, TGDN, maga]
15	0.203390	[LNYHBT, TGDN, tcot]
16	0.203390	[LNYHBT, maga, tcot]
17	0.203390	[TGDN, maga, tcot]
18	0.203390	[Trump, maga, tcot]
19	0.203390	[LNYHBT, TGDN, maga, tcot]

Slika 9: Česti skupovi heštagova

Sa slike možemo zaključiti da imamo 20 čestih skupova od kojih su 6 jednočlani (*LNYHBT*, *NeverHillary*, *TGDN*, *Trump*, *mega* i *tcot*), dok najveći skup sadrži 4 heštega. Najveću podršku imaju heštegovi *Trump* i *mega* čija je vrednost 0.347458.

	antecedants	consequents	antecedent support	consequent support	...	confidence	lift	leverage	conviction
0	(LNYHBT)	(TGDN)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
1	(TGDN)	(LNYHBT)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
2	(LNYHBT)	(maga)	0.203390	0.347458	...	1.000000	2.878049	0.132720	inf
3	(LNYHBT)	(tcot)	0.203390	0.364407	...	1.000000	2.744186	0.129273	inf
4	(TGDN)	(maga)	0.203390	0.347458	...	1.000000	2.878049	0.132720	inf
5	(TGDN)	(tcot)	0.203390	0.364407	...	1.000000	2.744186	0.129273	inf
6	(maga)	(tcot)	0.347458	0.364407	...	0.707317	1.941010	0.119147	2.171610
7	(LNYHBT, TGDN)	(maga)	0.203390	0.347458	...	1.000000	2.878049	0.132720	inf
8	(LNYHBT, maga)	(TGDN)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
9	(TGDN, maga)	(LNYHBT)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
10	(LNYHBT)	(TGDN, maga)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
11	(TGDN)	(LNYHBT, maga)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
12	(LNYHBT, TGDN)	(tcot)	0.203390	0.364407	...	1.000000	2.744186	0.129273	inf
13	(LNYHBT, tcot)	(TGDN)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
14	(TGDN, tcot)	(LNYHBT)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
15	(LNYHBT)	(TGDN, tcot)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
16	(TGDN)	(LNYHBT, tcot)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
17	(LNYHBT, tcot)	(maga)	0.203390	0.347458	...	1.000000	2.878049	0.132720	inf
18	(LNYHBT, maga)	(tcot)	0.203390	0.364407	...	1.000000	2.744186	0.129273	inf
19	(tcot, maga)	(LNYHBT)	0.245763	0.203390	...	0.827586	4.068966	0.153404	4.620339
20	(LNYHBT)	(tcot, maga)	0.203390	0.245763	...	1.000000	4.068966	0.153404	inf
21	(TGDN, tcot)	(maga)	0.203390	0.347458	...	1.000000	2.878049	0.132720	inf
22	(TGDN, maga)	(tcot)	0.203390	0.364407	...	1.000000	2.744186	0.129273	inf
23	(tcot, maga)	(TGDN)	0.245763	0.203390	...	0.827586	4.068966	0.153404	4.620339
24	(TGDN)	(tcot, maga)	0.203390	0.245763	...	1.000000	4.068966	0.153404	inf
25	(Trump, tcot)	(maga)	0.220339	0.347458	...	0.923077	2.656660	0.126831	8.483051
26	(tcot, maga)	(Trump)	0.245763	0.347458	...	0.827586	2.381833	0.117998	3.784746
27	(Trump, maga)	(tcot)	0.220339	0.364407	...	0.923077	2.533095	0.123097	8.262712
28	(LNYHBT, TGDN, tcot)	(maga)	0.203390	0.347458	...	1.000000	2.878049	0.132720	inf
29	(LNYHBT, TGDN, maga)	(tcot)	0.203390	0.364407	...	1.000000	2.744186	0.129273	inf
30	(LNYHBT, tcot, maga)	(TGDN)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
31	(TGDN, tcot, maga)	(LNYHBT)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
32	(LNYHBT, TGDN)	(tcot, maga)	0.203390	0.245763	...	1.000000	4.068966	0.153404	inf
33	(LNYHBT, tcot)	(TGDN, maga)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
34	(LNYHBT, maga)	(TGDN, tcot)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
35	(TGDN, tcot)	(LNYHBT, maga)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
36	(TGDN, maga)	(LNYHBT, tcot)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
37	(tcot, maga)	(LNYHBT, TGDN)	0.245763	0.203390	...	0.827586	4.068966	0.153404	4.620339
38	(LNYHBT)	(TGDN, tcot, maga)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf
39	(TGDN)	(LNYHBT, tcot, maga)	0.203390	0.203390	...	1.000000	4.916667	0.162022	inf

Slika 10: Pravila pridruživanja

Veliki broj pravila ima pouzdanost 1, što nam govori da kada se heštegovi sa leve strane nađu u nekom tvitu, uvek se nalaze i heštegovi s desne. Lift mera nam je uvek veća od 2 sem u jednom slučaju kada je za nijansu ispod što je takođe dobra osobina naših pravila.

7 Zaključak

Analizirajući dati skup podataka raznim metodama opisanim u ovom radu, došli smo do sledećih zaključaka:

- Vizualizacijom smo pronašli najkorišćenije reči u tvitovima korisnika i vremenske periode sa najvećim brojem objavljenih tvitova. Videli smo da korisnici uglavnom objavljuju tvitove vezane za kandidate na predsedničkim izborima.
- *Sentiment* analizom smo izvršili simulaciju glasanja kojom smo videli da su korisnici naklonjeniji kandidatu Donaldu Trampu.
- Klasifikacijom nismo uspeli sa velikom sigurnošću klasifikovati tvitove po korisnicima sa najvećim brojem istih
- Pronalaskom čestih skupova smo dodatno učvrstili naše uverenje da su tvitovi naklonjeni Donaldu Trampu jer se datim čestim skupovima nalaze i heštgovi *Trump* i *NeverHillary*.