TwitterTask

February 17, 2019

0.1 The Hebrew King of Twitter - Task

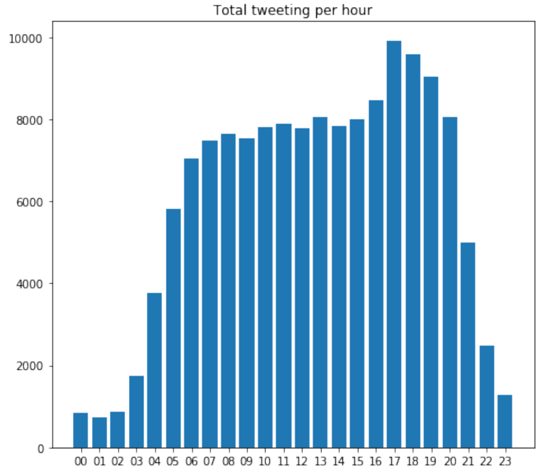
```
In [3]: SRC = "/home/itai/Desktop/KHealth/april_tweets"
        dirs = os.listdir(SRC)
        df = pd.read_csv(SRC+ "/" + dirs[0]).sample(frac = 0.1)
        for file in dirs[1:]:
            df = df.append(pd.read_csv(SRC + "/" + file).sample(frac=0.1,),ignore_index=True)
```

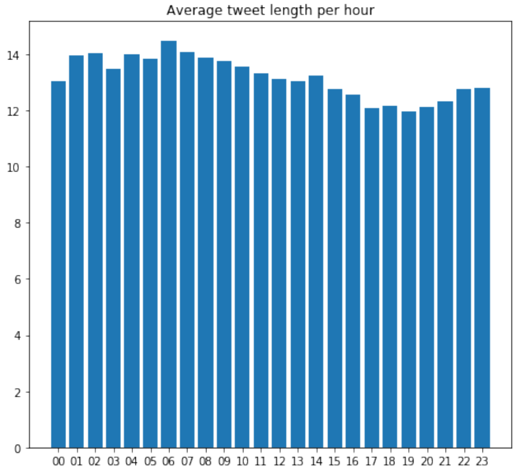
1 Cleaning the data:

```
In [6]: df["text"] = df['text'].str.replace(emoji_pattern,"")
        df["text"] = df['text'].str.replace('[^\w\s]','')
        df["text"] = df["text"].apply(lambda row: re.sub(r'[a-zA-Z0-9]','',row))
        df["text"] = df["text"].apply(lambda row: re.sub(r'[\n]',' ',row))
        df["text"] = df["text"].apply(lambda row: re.sub(r'[\s]{2,}',' ',row))
        df["text"]=df["text"].str.strip()
        df['text_len'] = df["text"].apply(lambda row: len(row.split(" ")))
        df['hour']=df["created_at"].apply(lambda row: row[11:13])
        df['day']=df["created_at"].apply(lambda row: row[-11:-9])

        df2= df[['hour']].groupby(df['hour']).count()
        df3 = df[['hour','text_len']].groupby(df['hour']).mean()
```

1.1 Basic histograms describing the data:

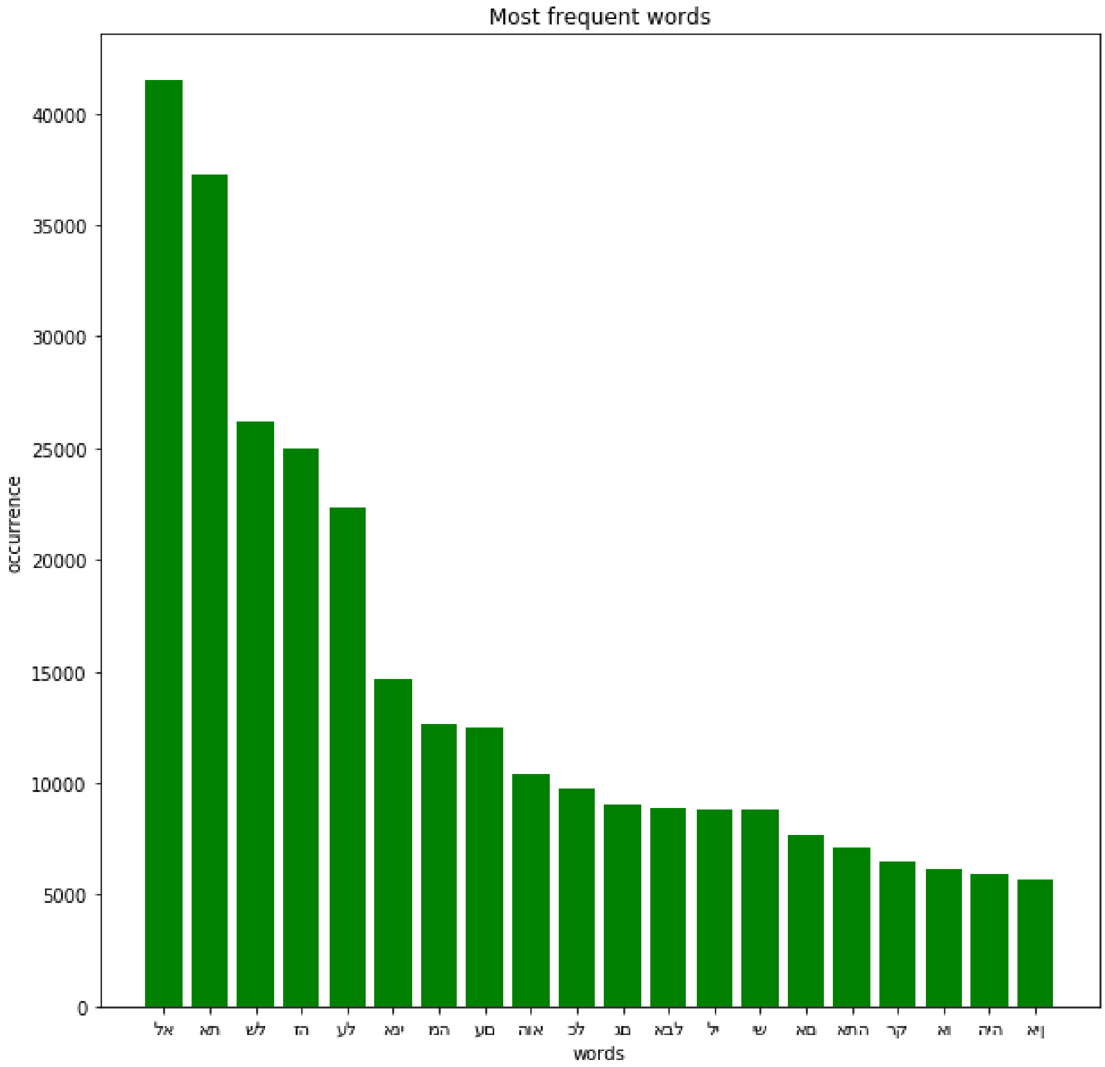
```
In [7]:
```





1.2 Most frequent words

```
In [9]:
```



```
In [10]:
```

```
Number of distinct words on our sample : 162867
```

1.3 Data after cleaning: removing all punctuations, Emojis, English letters:

1.4 removing rows with legnth less than 5:

2 List of ideas:

1. Clustering: Implematation of NLP "bi- grams" with vectorization and then applying the Kmeans algorithem in order to have data clusters by topic.

2. Supervised: "Spam classiffiyers": create labels to the train data as positive/negative emotions, list of topics of the tweets, and then use "bag of words/ ngrams" followed by SVM on a test data. Trying to predict the classes.

3. Deducing most frequent trends of tweets by NLP (ngrams), plotting versus a time line.

4. Deep Learning Rnn's ?

5. Hierarchical clustering - agglomerative clustering?

6. Some ensemble method combines 1,4,5 ?

2.1 Problems occured during the proccess:

4.* Must have a Robust Hebrew dictionary and stops words for having an effective language clustering.

5.** My PC CPU isn't sufficient for the Kmeans computation, need to work on cloud/spark for the database.

6. *** Further improve the data's cleaning proccess.

2.2 ngrams with Kmeans implemetation:

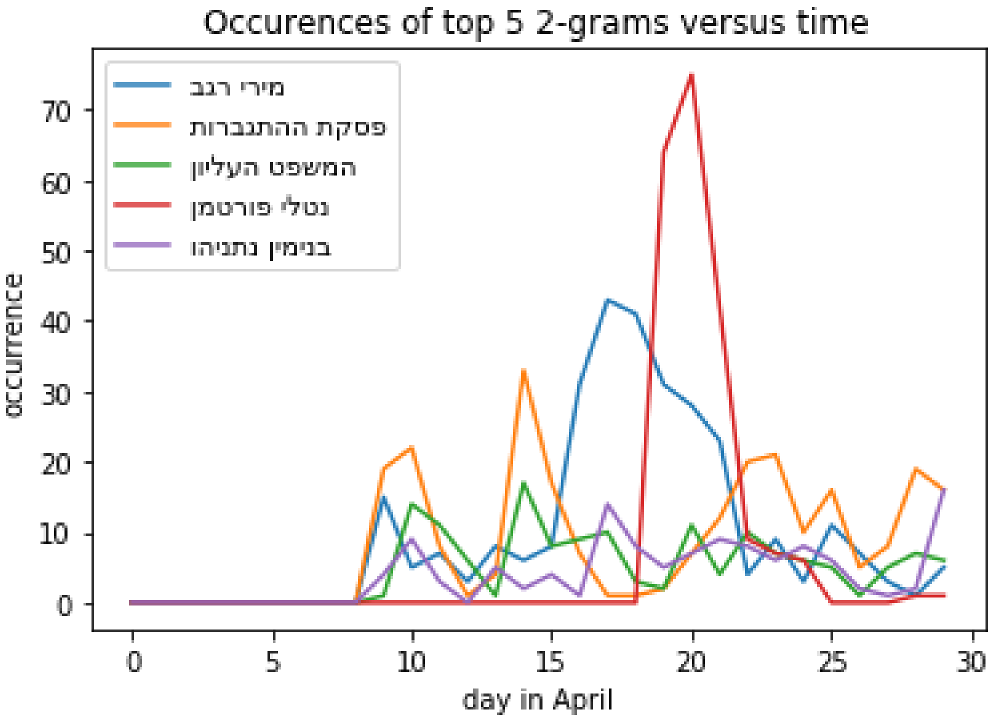
```
In [15]: from sklearn.feature_extraction import text
         from sklearn.feature_extraction.text import TfidfVectorizer
```

2.3 Excluding frequent hebrew words from the algorithm.

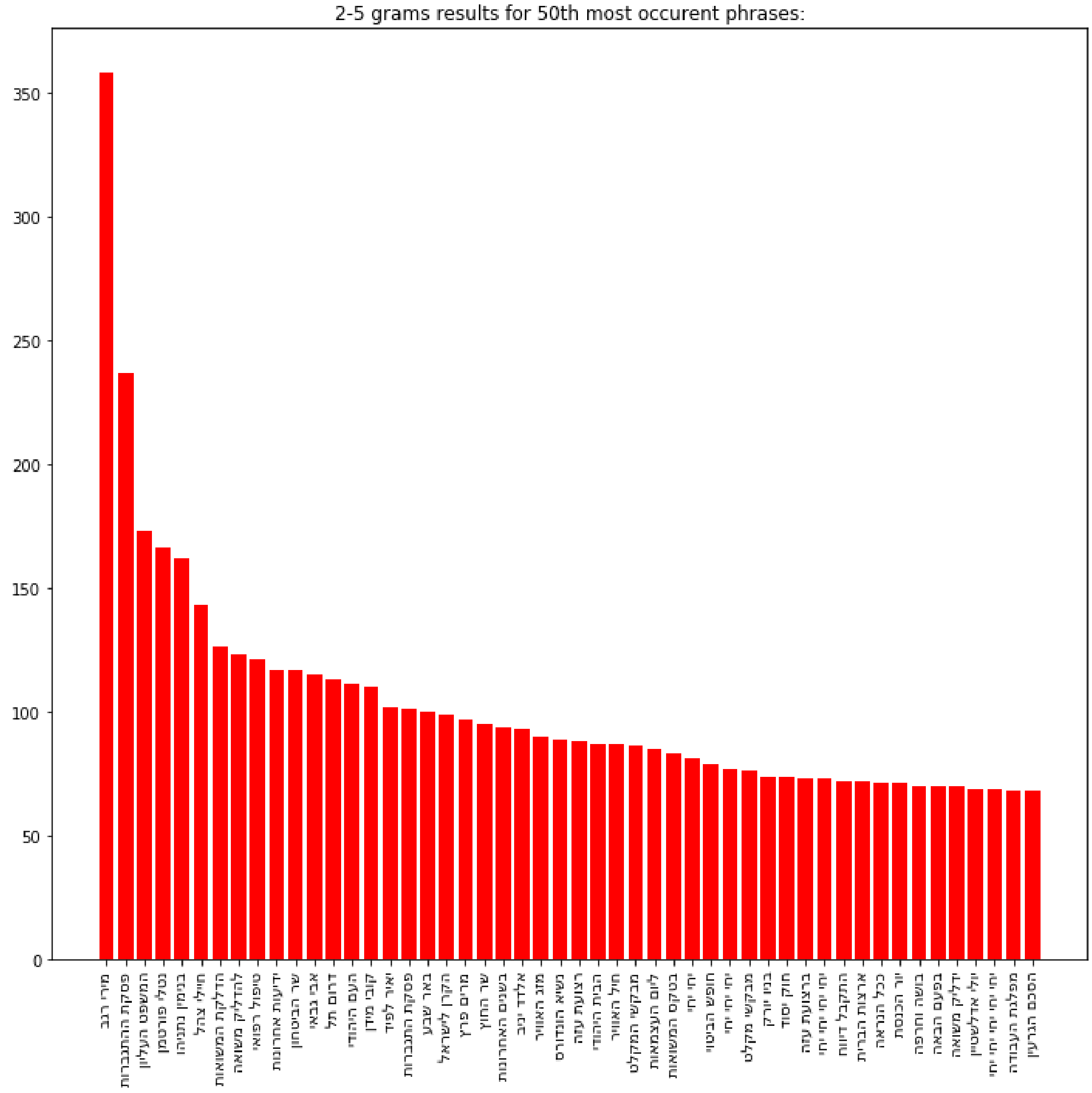
```
In [16]: heb_stopwords = []
         with open('/home/itai/Desktop/hebrewstopwords.txt') as file:
             for line in file:
                 line = line.strip()
                 heb_stopwords.append(line)
```

2.4 What are the 2 grams most frequent phrases?

```
In [19]:
```


Occurences of top 5 2-grams versus time
מירי רגב
פסקת ההתגברות
המשפט העליון
נטלי פורטמן
בנימין נתניהו
occurrence
0
10
20
30
40
50
60
70
0
5
10
15
20
25
30
day in April

In [20]:



2.5 Vectorization according to (1,2) grams: We pick 1000 features/words vocabulary for the clustering task:

```
In [42]: #Vectorizing with TF-IDF Vectorizer and creating X matrix
         tfv = TfidfVectorizer(ngram_range=(1,2), max_features=1000, stop_words =heb_stopwords
         #tfv.get_stop_words()
         X = tfv.fit_transform(df["text"]).todense()
         #print(X.shape)
```

2.6 Removing zeros rows from our vectorization matrix, in order to improve the Kmeans:

```
In [47]: X.shape
```

```
Out[47]: (99088, 1000)
```

```
In [48]: X_new = np.delete(X,zero_row_indices , 0)
```

```
In [49]: X_new.shape
```

```
Out[49]: (83619, 1000)
```

2.7 Applying Kmeans algorithem with 5 clusters:

```
In [50]: from sklearn.cluster import KMeans
```

```
In [51]: N_of_clusters = 5
         km = KMeans(n_clusters= N_of_clusters, init='k-means++', max_iter=100, n_init=1,verbos
```

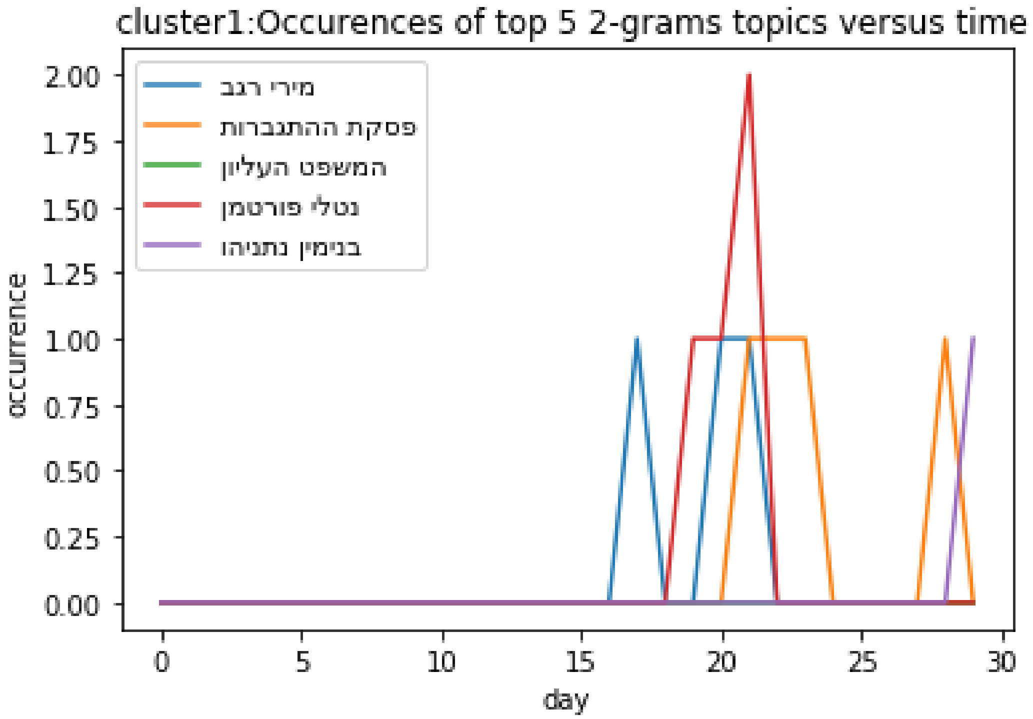
2.8 fitting Kmeans:

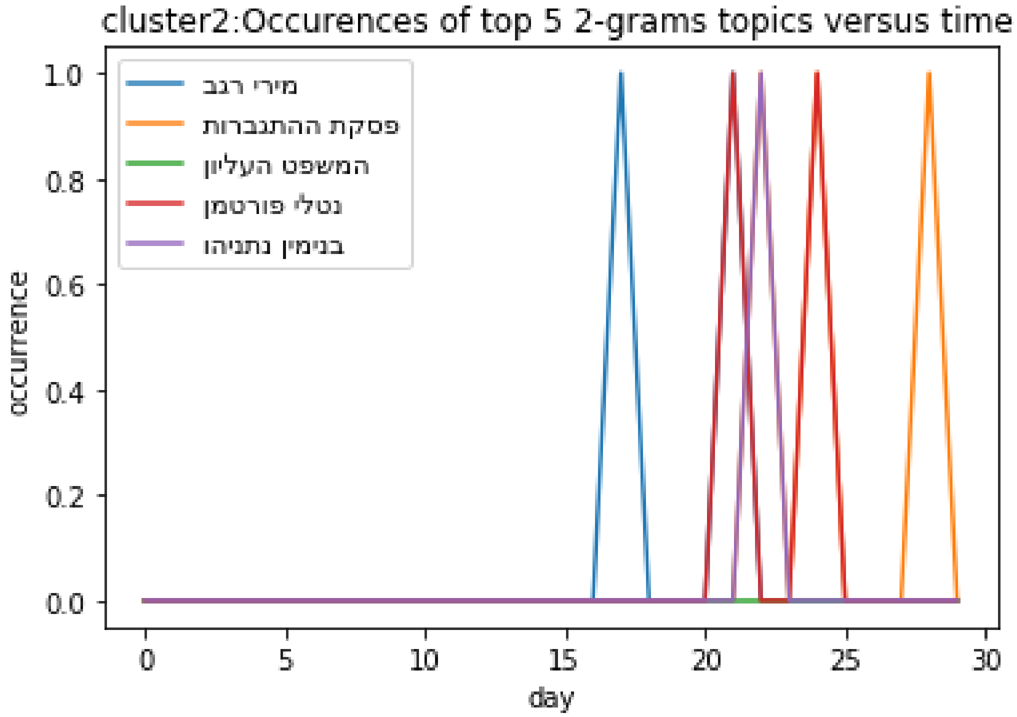
```
In [2]: km.fit(X_new)
```

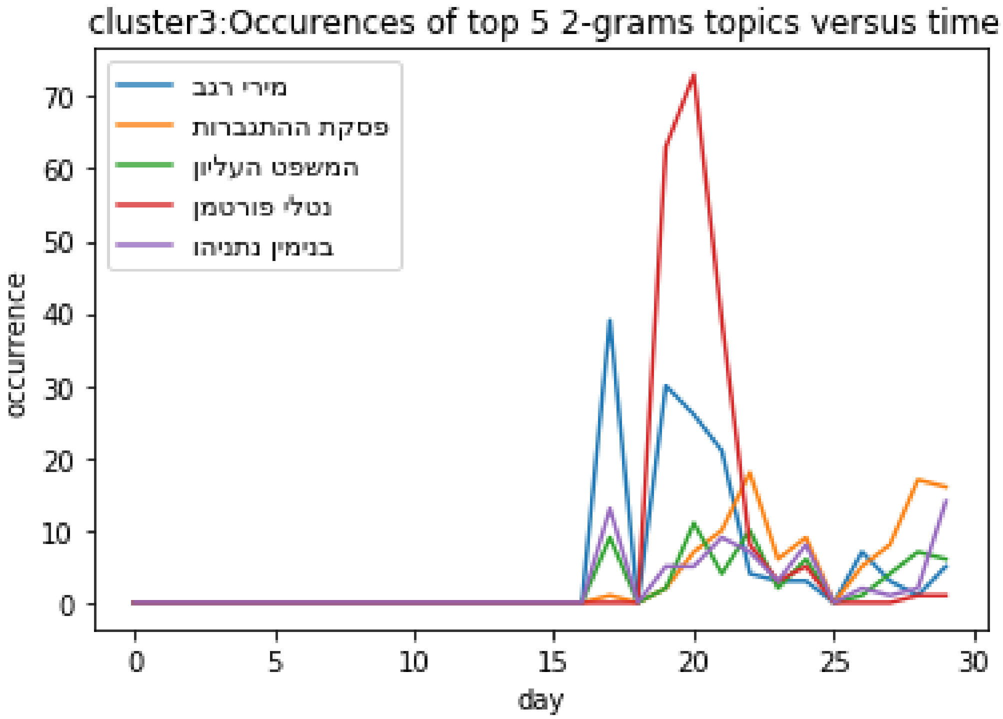
2.9 Examine the clustering we got:

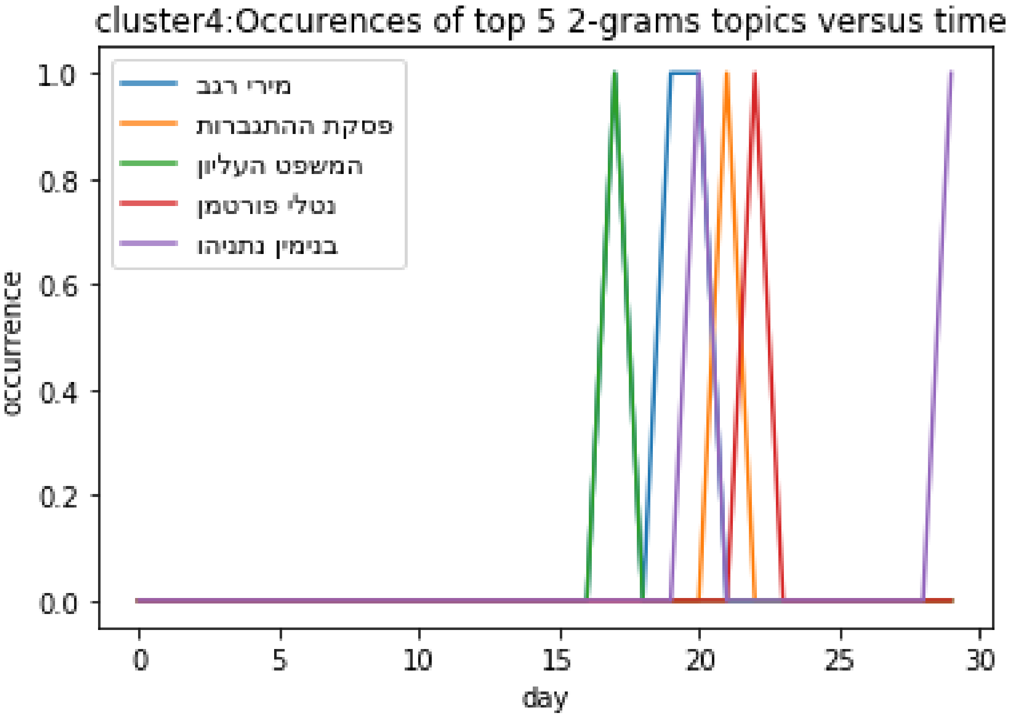
```
In [55]:
```

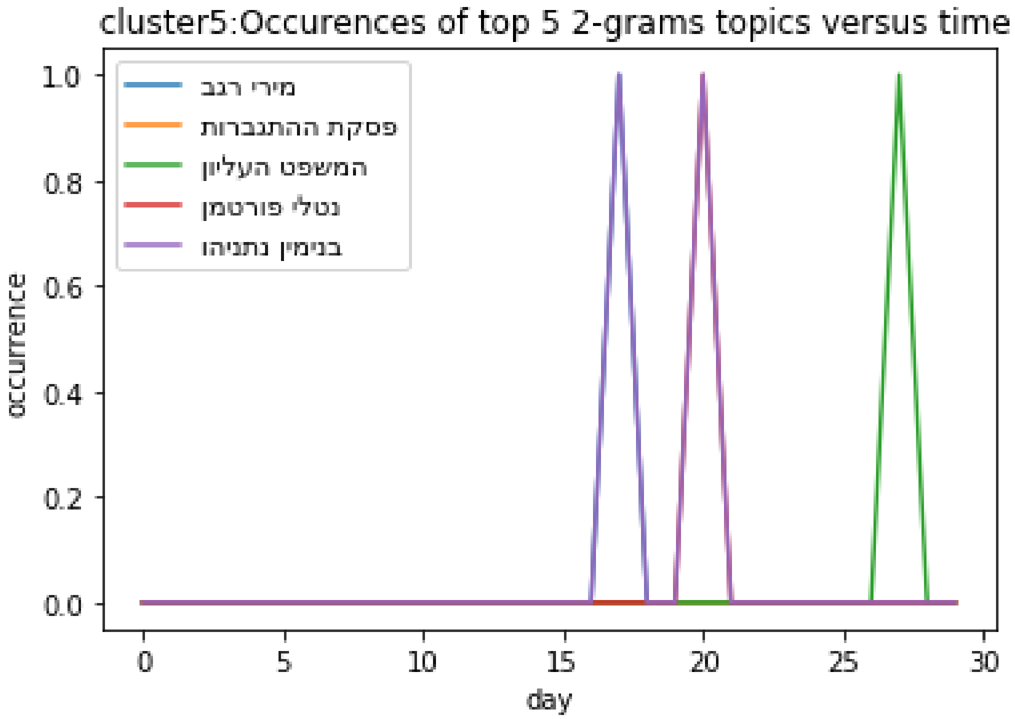
```
/home/itai/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:4: FutureWarning:
Passing list-likes to .loc or [] with any missing label will raise
KeyError in the future, you can use .reindex() as an alternative.

See the documentation here:
https://pandas.pydata.org/pandas-docs/stable/indexing.html#deprecate-loc-reindex-listlike
  after removing the cwd from sys.path.
```


cluster1:Occurences of top 5 2-grams topics versus time
מירי רגב
פסקת ההתגברות
המשפט העליון
נטלי פורטמן
בנימין נתניהו
occurrence
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
5
10
15
20
25
30
day


cluster2:Occurences of top 5 2-grams topics versus time
מירי רגב
פסקת ההתגברות
המשפט העליון
נטלי פורטמן
בנימין נתניהו
occurrence
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
25
30
day


cluster3:Occurences of top 5 2-grams topics versus time
מירי רגב
פסקת ההתגברות
המשפט העליון
נטלי פורטמן
בנימין נתניהו
occurrence
0
10
20
30
40
50
60
70
day
0
5
10
15
20
25
30


cluster4:Occurences of top 5 2-grams topics versus time
מירי רגב
פסקת ההתגברות
המשפט העליון
נטלי פורטמן
בנימין נתניהו
occurrence
0.0
0.2
0.4
0.6
0.8
1.0
day
0
5
10
15
20
25
30



2.10 Summary:

We were hoping for a better clustering. We deduce that we should have used a better hebrew vocabulary/stops words, as well as more computing power/cloud for larger data handling with more langueges features at hand.