simple-intro

April 26, 2016

1 Data mining New York Times articles

by Harshini Konduri, Pallavi Damle, Shantanu Saha, Harshvardhan Shukla, Koba Khitalishvili

1.1 Introduction

In our project we explore Natural Language Processing by mining the New York Times articles. The basic idea was to see if it is possible to classify the articles by topic using the article abstracts. By classifying the articles we are able to obtain the most important words for each topic. Potential use would be for marketing purposes to see what keywords are trending for a given topic. We used the TFxIDF statistic for each unique word as a feature for our classification model.

We are using open source tools like machine learning python library Scikit-learn and Beautiful soup for web scraping. We are proponents of reproducible research and you can find all our code at https://github.com/KobaKhit/Data-mining-using-NLP .

1.2 Results

We wrote a python script that downloads data from the New York times. We were able to obtain 36 thousand article abstracts and other information in less than 5 minutes. After preprocessing the raw text data and obtaining the TFxIDF statistics for each word across all articles we use the multinomial Naive Bayes model for classification.

You can see in appendix that our model has 100% accuracy on test data which basically means that the New York Times search engine is efficient. However, when classifying articles from different time periods by author the accuracy of the model is only 0.8. In every instance we provide the most important words by topic (author).

1.3 Conclusion

Going forward we want to focus on data preprocessing. Namely, edit the corpus of the articles and remove redundant words like prepositions. In this way we will be able to capture the actual topic of the article. According to Blei, efficient topic modeling increases the relevance and flexibility of article or any text data search results.

Additionaly, we would like to make use of other classifiers like RandomForest and Support vector machines.

1.4 References

- Blei, David. Probabilistic Topic Models. Communications of the ACM. April 2012. vol. 55. no. 4

1.5 Appendix

```
In [5]: from nytsnippetgetter import get_data
        import time
```

```
# start timer
start_time = time.time()

# get available number of pages for each topic. Each page is equivalent to 10 articles
topics=['economics','politics','espionage','global+warming', 'clinton', 'sanders', 'guns', 'canc
npages = [1500,1000,500,100,100,100,100,100,100]
# topics=['economics','politics']
get_data(topics, BEGINDATE = 20131213, LIMITS=True) # articles written since 2013-December-13
```

```
Total number of pages available (each page is 10 articles):
economics :  4584
politics :  19196
espionage :  1181
global+warming :  2035
clinton :  9229
sanders :  3553
guns :  3369
cancer :  7645
sex :  10003
```

In [6]:
```
# download article data
articles = get_data(TOPICS = topics, NPAGES = npages, BEGINDATE = 20131213)
# articles = get_data(TOPICS = topics, NPAGES = [150,100,50,10,10,10,10,10,10])
# articles = get_data(TOPICS = topics, BEGINDATE = 20131213, NPAGES = [15,10])
```

```
Topics:  ['economics', 'politics', 'espionage', 'global+warming', 'clinton', 'sanders', 'guns', 'cancer
NPages:  [1500, 1000, 500, 100, 100, 100, 100, 100, 100]

Total documents:  36000
Started download...
economics is done | 1500/3600
politics is done | 2500/3600
espionage is done | 3000/3600
global+warming is done | 3100/3600
clinton is done | 3200/3600
sanders is done | 3300/3600
guns is done | 3400/3600
cancer is done | 3500/3600
sex is done | 3600/3600

Done in  457.22085785865784 seconds
```

In [7]:
```
# articles is a list of objects with each object being one document
articles[1]
```

Out[7]:

```
{'abstract': 'Hating on anyone who suggests limits.',
 'author': 'PAUL KRUGMAN',
 'date_modified': '2016-04-22T16:55:06Z',
 'date_published': '2016-04-22T10:54:17Z',
 'keywords': [],
 'lead_paragraph': None,
 'nytclass': None,
 'section_name': {'content': 'opinion', 'display_name': 'Opinion'},
 'snippet': 'be doing anything especially new | on the contrary, he and his defenders claimed t
```

```
 'title': 'Sarandonizing Economics',
 'user_topic': 'economics',
 'weburl': 'http://krugman.blogs.nytimes.com/2016/04/22/sarandonizing-economics/'}
```

```
In [8]: # number of articles
        len(articles)
```

```
Out[8]: 36000
```

```
In [9]: # some articles are missing lead paragraph and abstract. just use snippets for now
        data = [ x['snippet'] for x in articles]

        # check if data has Nones
        nonesidx = [data.index(x) for x in data if x == None and len(x) > 0]
        if len(nonesidx) > 0:
            print("You have nones. Below are indices of articles.")
            print(nonesidx)
        else:
            print("No Nones. Good to go.")

        data[0:3]
```

```
No Nones. Good to go.
```

```
Out[9]: ['Psychology in economics',
         'be doing anything especially new | on the contrary, he and his defenders claimed that they wer
         'GASOLINE prices on the South Fork are consistently 20 to 50 cents more per gallon than in othe
```

```
In [10]: # article topics
         label = [ x['user_topic'] for x in articles]
         label[-5:-1]
```

```
Out[10]: ['sex', 'sex', 'sex', 'sex']
```

```
In [11]: # http://scikit-learn.org/stable/datasets/twenty_newsgroups.html

         from sklearn.feature_extraction.text import TfidfVectorizer
         from sklearn import preprocessing

         vectorizer = TfidfVectorizer() # Convert a collection of raw documents to a matrix of TF-IDF f
         le = preprocessing.LabelEncoder() # Encode labels with value between 0 and n_classes-1.

         vectors = vectorizer.fit_transform(data)
         target = le.fit_transform(label)

         print(vectors.shape, target.shape,'\n')
         print("Number of unique words (features) across all docs: ", vectors.shape[1], '\n')
         print("(docid, wordid) TFIDF", '\n')
         print(vectors[0,])
```

```
(36000, 1249) (36000,)

Number of unique words (features) across all docs:  1249

(docid, wordid) TFIDF
```

```
  (0, 376)        0.423466656957
  (0, 553)        0.331808863831
  (0, 889)        0.84295840249
```

```
In [12]: # average number of non-zero components by sample, i.e. average number of words with non-zero i
         vectors.nnz / float(vectors.shape[0])
```

```
Out[12]: 23.18888888888889
```

```
In [13]: from sklearn.cross_validation import train_test_split

         X_train, X_test, y_train, y_test = train_test_split(vectors, target, test_size=0.33, random_sta
         print(X_train.shape, X_test.shape)
```

```
(24120, 1249) (11880, 1249)
```

```
In [14]: from sklearn.naive_bayes import MultinomialNB
         from sklearn import metrics

         import pandas as pd

         clf = MultinomialNB(alpha=2)
         clf.fit(X_train, y_train)
         pred = clf.predict(X_test)
         print("F-score: ", metrics.f1_score(y_test, pred, average='weighted'))
         print("Accuracy: ", metrics.precision_score(y_test, pred, average='weighted'))
         print("Confusion matrix:\n", pd.crosstab(y_test, pred, rownames=['True'], colnames=['Predicted
```

```
F-score:  1.0
Accuracy:  1.0
Confusion matrix:
 Predicted    0    1     2     3    4    5     6    7    8
True
0          337    0     0     0    0    0     0    0    0
1            0  321     0     0    0    0     0    0    0
2            0    0  4910     0    0    0     0    0    0
3            0    0     0  1670    0    0     0    0    0
4            0    0     0     0  334    0     0    0    0
5            0    0     0     0    0  343     0    0    0
6            0    0     0     0    0    0  3327    0    0
7            0    0     0     0    0    0     0  313    0
8            0    0     0     0    0    0     0    0  325
```

```
In [15]: import numpy as np

         def show_top10(classifier, vectorizer, categories):
             feature_names = np.asarray(vectorizer.get_feature_names())
             for i, category in enumerate(categories):
                 top10 = np.argsort(classifier.coef_[i])[-10:]
                 print("%s: %s" % (category, " ".join(feature_names[top10])))

         # print ten most important words for each topic
         show_top10(clf, vectorizer, le.classes_)
```

```
cancer: with is and breast the of for as strong cancer
clinton: trump art facility correctional mr his the in strong clinton
```

```
economics: exposure and that of to psychology in the strong economics
espionage: his and she to was in had the strong espionage
global+warming: group is planet to climate the warming change strong global
guns: night fort of increase at roses for strong the guns
politics: may to province havana oriente arms by washington of the
sanders: weight woman to year strong show is larry the sanders
sex: that about its strangers of and in the strong sex
```

In [16]:
```
# end timer
print(time.time()-start_time, 'seconds')
```

```
495.6035737991333 seconds
```

In [17]:
```
# test the model on articles written before 2013-December-13
test = get_data(TOPICS = topics, ENDDATE = 20131213, NPAGES = [10,10,10,10,10,10,10,10,10,10])
test_data = [ x['snippet'] for x in test]
test_label = [ x['user_topic'] for x in test]

test_vectors = vectorizer.transform(test_data)
test_target = le.transform(test_label)

print(test_vectors.shape, test_target.shape,'\n')
print("Number of unique words (features) across all docs: ", vectors.shape[1], '\n')
print("(docid, wordid) TFIDF", '\n')
print(vectors[0,])
```

```
Topics:  ['economics', 'politics', 'espionage', 'global+warming', 'clinton', 'sanders', 'guns', 'cancer
NPages:  [10, 10, 10, 10, 10, 10, 10, 10, 10, 10]

Total documents:  1000
Started download...
economics is done | 10/100
politics is done | 20/100
espionage is done | 30/100
global+warming is done | 40/100
clinton is done | 50/100
sanders is done | 60/100
guns is done | 70/100
cancer is done | 80/100
sex is done | 90/100

Done in  13.58777117729187 seconds
(900, 1249) (900,)

Number of unique words (features) across all docs:  1249

(docid, wordid) TFIDF

  (0, 376)        0.423466656957
  (0, 553)        0.331808863831
  (0, 889)        0.84295840249
```

In [18]:
```
pred_test = clf.predict(test_vectors)
print("F-score: ", metrics.f1_score(test_target, pred_test, average='weighted'))
print("Accuracy: ", metrics.precision_score(test_target, pred_test, average='weighted'))
print("Confusion matrix:\n", pd.crosstab(test_target, pred_test, rownames=['True'], colnames=[
```

```
F-score:  1.0
Accuracy:  1.0
Confusion matrix:
 Predicted    0    1    2    3    4    5    6    7    8
True
0          100    0    0    0    0    0    0    0    0
1            0  100    0    0    0    0    0    0    0
2            0    0  100    0    0    0    0    0    0
3            0    0    0  100    0    0    0    0    0
4            0    0    0    0  100    0    0    0    0
5            0    0    0    0    0  100    0    0    0
6            0    0    0    0    0    0  100    0    0
7            0    0    0    0    0    0    0  100    0
8            0    0    0    0    0    0    0    0  100
```

1.5.1 Classify by author

```
In [19]: # article author
         label = [ x['author'] for x in articles]
         label[-5:-1]

Out[19]: ['TIMOTHY EGAN', 'KAREN ALEXANDER', 'DWIGHT GARNER', 'BEN KENIGSBERG']

In [20]: target = le.fit_transform(label)

         X_train, X_test, y_train, y_test = train_test_split(vectors, target, test_size=0.33, random_sta
         print(X_train.shape, X_test.shape)

(24120, 1249) (11880, 1249)

In [30]: clf.fit(X_train, y_train)
         pred = clf.predict(X_test)
         print("F-score: ", metrics.f1_score(y_test, pred, average='weighted'))
         print("Accuracy: ", metrics.precision_score(y_test, pred, average='weighted'))
         # print("Confusion matrix:\n", pd.crosstab(y_test, pred, rownames=['True'], colnames=['Predict

F-score:  0.815594555496
Accuracy:  0.791768788424

//anaconda/envs/tpot/lib/python3.5/site-packages/sklearn/metrics/classification.py:1074: UndefinedMetri
  'precision', 'predicted', average, warn_for)
//anaconda/envs/tpot/lib/python3.5/site-packages/sklearn/metrics/classification.py:1074: UndefinedMetri
  'precision', 'predicted', average, warn_for)

In [22]: # print ten most important words for each topic
         show_top10(clf, vectorizer, le.classes_)

: washington by exposure for and economics of in the psychology
ALEX BERENSON: these implicitly either them corporation calif rand goldman we make
AMY HARMON: months approach toxic less treatment promising breakthrough effective championed results
ANDREW JACOBS: her ms coup little platform assert campaign oust rousseff état
ANDREW R. CHOW: coachella angus reversed brought singer will guns roses ac dc
ANNIE CORREAL: spends son previously alpert edited month andavolu endings obit his
BEN KENIGSBERG: portrait hardly radio paints 90 91 wfmu fm film city
BENJAMIN WEISER: sporyshev based prisoners television charged inspired was the mr he
CEYLAN YEGINSU: journalists quickly istanbul media potential concerns prompting whether turkish closed
```

```
CHARLES ISHERWOOD: carlyle top thomas cirque concurrently lucas du season timing continuing
CHARLES M. BLOW: bash comment odd planned nomination cnn thursday convention if democratic
CHOE SANG-HUN: chul dong naturalized steal kim claims koreans had south military
CLINTON: namt vas incident collano employ mike himself tree lathrop sawing
CNBC: exposure facing of his support research discusses parker immunotherapy sean
DEAN R. LEIMER, and SELIG D. LESNOY: support offset impact however recommendations saving negative desig
DENISE GRADY: 52 nearly disease cure the 000 breast still strong cancer
DWIGHT GARNER: among senior poems poem flowering betjeman novels front writes bring
ERIK PIEPENBURG: novelist theater younger female hander someone male company relationship frisky
FARAH STOCKMAN: philadelphia away crossing appeared protested rally carrying april his so
GARDINER HARRIS: moonshot 2003 initiative barriers biden jr will to cancer her
GEORGE R. HARRIS: imitation went immediately fired 77 was the it whiz bang
GINA KOLATA: strong officially type patients downgraded decided panel thousands tumor cancer
GRAHAM BOWLEY: monday him thrown sexual effort cosby assault pennsylvania appeals blocked
INTERNATIONAL HERALD TRIBUNE: defense admitted admission spy germany minister had been west she
JAMES PONIEWOZIK: jeffrey garry audience hank preparing hear tambor the sanders larry
JAVIER C. HERNÁNDEZ: sentenced documents threats beijing technician china 150 selling combat aggressive
JEREMY EGNER: creating led precursor invite performances sorts tonight host show guest
JOHN SCHWARTZ: mobil eric investigation schneiderman statements is strong exxon global warming
JON PARELES: foot arena brings together festival immobile keep injured reunited founding
JON SANDERS FILMS: her loss come is inspirational terms struggling since widow teacher
KAREN ALEXANDER: into others virility step gym quest trap unwittingly turbocharged rats
KAREN W. ARENSON: lesnoy disclosed calculation paper erred turned flaw the in feldstein
KEN JAWOROWSKI: strong writing intriguing saves script moments place the sex partly
KJ DELL'ANTONIA: williams starts good mary catastrophes those phone happy elizabeth with
LISA SANDERS, M.D: solve readers unexplained gain year old why woman 59 weight
MARK MAZZETTI: head 83 lawyer intelligence role leesburg contra career in his
MARTIN S. FELDSTEIN: years during problems few increasing decide next financial deal the
MICHAEL SCHWIRTZ and MICHAEL WINERIP: with district reviewing attorney jurisdiction inmate violent encou
MICHAEL WINERIP, MICHAEL SCHWIRTZ and TOM ROBBINS: brutality northern bad investigations federal murdere
MIKKAEL A. SEKERES, M.D: say environmental impossible need cancers collective randomly except nose speci
NEIL GENZLINGER: isn technologically dementia easier impactful personal coherent thank struggle playing
NO POLITICS: jnew 19o3 _____ benefit 22 jan gratify committee mcgillicudy some
PATRICK J. EGAN and MEGAN MULLIN: evenly pleasant living period across themselves although seasons coun
PAUL KRUGMAN: to were an as they motivated reasoning antidote models doing
PETER KEEPNEWS: ran dark behind comics hbo 1998 stand show sanders larry
Pamela G. Hollie: sisters penthouse guccione own begun suspense winston twin will magazine
REUTERS: downplayed sarcastically calling win state presidential senator victory achievement trump
ROGER COHEN: politics is change planet come the body am therefore home
SAMANTH SUBRAMANIAN: survey carry instead secret das became britain enchanted sarat chandra
SIMONE GUBLER: administrators sporting daily unarmed events law carve environment guns into
STEWART AIN: awakening residents cents consistently gallon areas south gasoline fork prices
SYLVIANE GOLD: hartford has arrived laptop clunky antiquated gone matter delightful time
Special to The New York Times: coolidge action authorized munitions hughes establishing shipment issued
THE ASSOCIATED PRESS: pellet felony middleweight kelly shooting ohio accused pavlik indicted matt
TIMOTHY EGAN: regarding catholics pope apostolic desire stirring teachings exhortation church skeptics
WILLIAM B. GAIL: significant longstanding century hence predicted comprehend repeatable yourself scienti
Wireless to THE NEW YORK TIMES: wireless acclaimed strong by espionage play new london hackett walter
YAMICHE ALCINDOR: bill citizens situation unemployed 2000 mrs strong hillary five clinton
```

1.5.2 Classify by author using articles from different time periods

In [23]:
```
# Classify by author using articles from different time periods
articles = get_data(TOPICS = topics, NPAGES = [150,100,50,10,10,10,10,10,10])
```

```
print(articles[2])
# number of articles
print(len(articles))
```

```
Topics:  ['economics', 'politics', 'espionage', 'global+warming', 'clinton', 'sanders', 'guns', 'cancer
NPages:  [150, 100, 50, 10, 10, 10, 10, 10, 10]

Total documents:  3600
Started download...
economics is done | 150/360
politics is done | 250/360
espionage is done | 300/360
global+warming is done | 310/360
clinton is done | 320/360
sanders is done | 330/360
guns is done | 340/360
cancer is done | 350/360
sex is done | 360/360

Done in  44.43974494934082 seconds
{'section_name': {'display_name': 'N.Y. / Region', 'content': 'nyregion'}, 'date_published': '2008-03-30
3600
```

In [24]:
```
# some articles are missing lead paragraph and abstract. just use snippets for now
data = [ x['snippet'] for x in articles]

# check if data has Nones
nonesidx = [data.index(x) for x in data if x == None and len(x) > 0]
if len(nonesidx) > 0:
    print("You have nones. Below are indices of articles.")
    print(nonesidx)
else:
    print("No Nones. Good to go.")

data[6:9]
```

```
No Nones. Good to go.
```

Out[24]:
```
['decide during the next few years how best to deal with the increasing financial problems of 
 "this time for a theory of economic policy that is appropriate for the 1970's and 80's in much
 'standards on scaffolding, combined savings. asbestos exposure, cadmium and chromium exposure
```

In [25]:
```
# article authors
label = [ x['author'] for x in articles]
label[-5:-1]
```

Out[25]:
```
['TIMOTHY EGAN', 'KAREN ALEXANDER', 'DWIGHT GARNER', 'BEN KENIGSBERG']
```

In [26]:
```
vectors = vectorizer.fit_transform(data)
target = le.fit_transform(label)

X_train, X_test, y_train, y_test = train_test_split(vectors, target, test_size=0.33, random_sta
print(X_train.shape, X_test.shape)
```

```
(2412, 1249) (1188, 1249)
```

In [29]:
```
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
print("F-score: ", metrics.f1_score(y_test, pred, average='weighted'))
print("Accuracy: ", metrics.precision_score(y_test, pred, average='weighted'))
# print("Confusion matrix:\n", pd.crosstab(y_test, pred, rownames=['True'], colnames=['Predict
```

```
F-score:  0.815594555496
Accuracy:  0.791768788424

//anaconda/envs/tpot/lib/python3.5/site-packages/sklearn/metrics/classification.py:1074: UndefinedMetri
  'precision', 'predicted', average, warn_for)
//anaconda/envs/tpot/lib/python3.5/site-packages/sklearn/metrics/classification.py:1074: UndefinedMetri
  'precision', 'predicted', average, warn_for)
```

In [28]:
```
print("Homogeneity: %0.3f" % metrics.homogeneity_score(y_test, pred))
print("Completeness: %0.3f" % metrics.completeness_score(y_test, pred))
print("V-measure: %0.3f" % metrics.v_measure_score(y_test, pred))
print("Adjusted Rand-Index: %.3f"
      % metrics.adjusted_rand_score(y_test, pred))
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(X_test, pred, sample_size=1000))
```

```
Homogeneity: 0.774
Completeness: 0.987
V-measure: 0.868
Adjusted Rand-Index: 0.814
Silhouette Coefficient: 0.405
```