deletweet-hashtag-recommender

April 4, 2017

1 DELETWEET HASHTAG RECOMMENDER

```
In [11]: import csv
         import json
         import pandas
         from random import randint
```

1.1 USE RESERVOIR SAMPLING TO SUBSET THE DATA

In order to cluster the data in a reasonable amount of time, a subset of the original dataset was needed. The reservoir sampling algorithm implemented for this class was used to gather a truly random sample of 10,000 tweets.

This section is just a demonstration and does not represent the actual subset used for clustering.

```
In [12]: # import dataset and extract tweets to feed algorithm
         deletweet = pandas.read_csv('../../deletweet/data/deleted_tweets_cleaned.csv')
         tweets = deletweet['tweet']

In [13]: def reservoir(stream, k):
             '''
             Populate sample space with first k items from stream. For remaining items in stream,
             choose a random number j from 0 to item's index. If j is less than k, replace
             jth element in sample with ith element from stream
             '''
             sample = stream[0:k]

             for i in range(k, len(stream)):
                 j = randint(0, i + 1)
                 if j < k:
                     sample[j] = stream[i]

             return sample

In [14]: # take the sample
         samples = reservoir(tweets, 10000)

In [15]: # make sure we have 10,000 tweets
         len(samples)

Out[15]: 10000

In [16]: # take a look at first sample
         tweet = json.loads(samples[0])
         tweet['text']
```

```
Out[16]: "Enjoyed another opportunity to visit with Nebraskans at today's Open Coffee in Holdrege. Than
```

1.2 EXAMINE SUBSET USED FOR CLUSTERING

```
In [17]: tweets = {}
         hashtag_dict = {}

In [18]: # import subset used for clutering
         with open('../data/deletweet_subset_10000.json', 'r') as f:

             for line in f:
                 tweet = json.loads(line)
                 tweets[tweet['id']] = tweet

In [19]: # number of tweets in subset
         len(tweets)

Out[19]: 10000

In [20]: # extract hashtags from tweets in subset
         for key in tweets.keys():
             tweet = tweets[key]
             if (tweet['entities']['hashtags']):
                 hashtag_dict[key] = [tweet['entities']['hashtags'][i]['text'] \
                                      for i in range(len(tweet['entities']['hashtags']))]

In [21]: # how many tweets in the sample have hashtags
         len(hashtag_dict)

Out[21]: 4830

In [22]: tags = []
         for key in hashtag_dict.keys():
             for tag in hashtag_dict[key]:
                 tags.append(tag)

         # how many individual hashtags are in the sample
         len(tags)

Out[22]: 7821

In [23]: # how many unique hashtags are in the sample
         len(set(tags))

Out[23]: 4300
```

1.3 CLUSTER VIA JACCARD DISTANCE AND K-MEANS

Based on the average number of unique hashtags in any subset of this dataset - which is roughly 40% of the number of tweets in the subset - 1,000 was the number of clusters chosen. This seemed to strike a balance between the uniqueness of the clusters and the likelihood that a cluster would have at least one hashtag in it that could be recommended to other tweets in the cluster.

The clustering of 10,000 tweets into 1,000 clusters was done via K-Means clustering using Jaccard Distance as the distance metric. The implemenatation of the algorithm used is open source, although I updated the code to make it run under Python 3.

The clustering took over 50 hours to complete, so the code will not be included here.

1.4 IMPORT CLUSTERS

```
In [24]: clusters = pandas.read_csv('../data/clusters_1000_from_10000_reformat.csv')
```

```
In [25]: tweet_ids_int = []

         for i in range(len(clusters)):
             nums = []
             stripped = clusters['tweet_ids'][i].strip('{}').split(', ')

             for num in stripped:
                 nums.append(int(num))

             tweet_ids_int.append(nums)

         clusters['tweet_ids_int'] = tweet_ids_int

         # same as above:
         # tweet_ids_int = [[int(num) for num in clusters['tweet_ids'][i].strip('{}').split(', ')] \
         # for i in range(len(clusters))]
```

```
In [26]: # parse output of clustering algorithm into following format:
         #
         # {
         # cluster_01_id: {tweet_01_id: tweet_01_text}, [...], {tweet_n_id: tweet_n_text},
         # [...],
         # cluster_n_id: {tweet_01_id: tweet_01_text}, [...], {tweet_n_id: tweet_n_text}
         # }
         #
         # this takes about an hour, need to optimize
         clusters_dict = {}

         for i in range(len(clusters)):
             clusters_dict[i] = {}
             for j in range(len(deletweet)):
                 tweet_id = deletweet['id'][j]
                 if tweet_id in clusters['tweet_ids_int'][i]:
                     clusters_dict[i][tweet_id] = deletweet['content'][j]
```

1.5 EXTRACT AND RECOMMEND HASHTAGS

```
In [27]: # clusters to use as demonstration
         demo_clusters = [3, 108, 150, 263, 374, 453, 509]
```

```
In [28]: # look at the tweets in a cluster
         print('cluster_id: {}'.format(demo_clusters[0]))
         for key in clusters_dict[demo_clusters[0]].keys():
             print('{tweetid}: {tweettext}'.format(tweetid=key, tweettext=clusters_dict[demo_clusters[0]
```

```
cluster_id: 3
309057990382714880: With @SenBlumenthal at today's Veterans of Foreign Wars hearing. #vets @DeptVetAffa
228553954458488832: As result of #HCR, more than 5.2M seniors &amp; people with disabilities have saved
783109774753792002: A lot of people can't handle it. |Trump on veterans with post-traumatic stress http
169840713356423168: Gulf Coast Vietnam Veterans Salute http://t.co/wDbzAQUm
174895471893028865: Met with Fred S. Sganga, Exec. Dir. of LI State Veterans Home. He briefed me on the
```

```
In [29]: # extract hashtags from the tweets in the cluster
         cluster_tags = []

         for key in clusters_dict[demo_clusters[0]].keys():
             for i in range(len(deletweet)):
                 if deletweet['id'][i] == key:
                     tweet = json.loads(deletweet['tweet'][i])
                     # pull out hashtags from tweet object
                     for tag in [tweet['entities']['hashtags'][i]['text'] \
                                 for i in range(len(tweet['entities']['hashtags']))]:
                         if tag not in cluster_tags:
                             cluster_tags.append(tag)

         for item in cluster_tags:
             print('#{}'.format(item))
```

```
#vets
#HCR
#SticksandStones
#SAU
#NY9
```

This first example is typical of most of the clusters, and is a good indicator of the difficulties of trying to set up a hashtag recommendation system.

On a general level the tweets in the above cluster have to do with veterans and health care. More specifically, 3 out of the 4 tweets mention veterans, while the outlier tweet is about health care and seniors, the latter of which happens to apply to many veterans. The cluster is a good condidate for a hashtag recommendation system, as the tweets have two overlapping, but separate themes from which to pull potential hashtags. And in fact, if our system were to recommend the hashtag #vets from the first tweet to the others in the cluster, it would be applicable to 2 out of the 3 remaining tweets. The next hashtag - #HCR, an acronym for "health care reform" - would be slightly less successful, in that it is applicable to only 1 out of the 3 remaining tweets.

The last hashtag - #NY9, which stands for New York's 9th congressional district - presents a more complicated scenario, and one that appears frequently when attempting to recommend hashtags in this manner. While the tweet itself has been clustered correctly based on its text content, the hashtag is specific enough that the chances of it being applicable to another tweet in the dataset is very low.

This situation arose many times when looking through the clusters; for example, tweets are correctly clustered together due to the fact that they all deal with sports, but the hashtags present in the cluster all reference specific teams, and therefore are not applicable to the other tweets in the cluster.

This difficulty may be less of a problem as the size of the input dataset grows, as more data would presumably allow the clusters to become more specific. Another, potentially concurrent, method to alleviate this difficulty is to increase the number of clusters, also allowing for more specificity. However, both these potential solutions come at large computational costs, and the system is already prohibitvely costly in this domain. Increasing the number of clusters also lowers the number of hashtags available to each cluster, which is a disadvantage for recommendation.

```
In [30]: # look at the tweets in a cluster
         print('cluster_id: {}'.format(demo_clusters[1]))
         for key in clusters_dict[demo_clusters[1]].keys():
             print('{tweetid}: {tweettext}'.format(tweetid=key, \
                                                   tweettext=clusters_dict[demo_clusters[1]][key]))
```

```
cluster_id: 108
746031782907129856: RT @RepKClark: 24 hours ago, we began this sit-in to demand votes on common sense gu
745821640970342401: RT @RepKClark: Staying on the House floor to demand a vote on gun safety bills. #No
```

```
745732604293320704: RT @WoodsGoods: Ty to @dinatitus who is holding her ground in support common sense 
745779309663518720: 9hrs and still on the House floor feeling united and proud to stand up as one to re
```

```
In [31]: # extract hashtags from the tweets in the cluster
         cluster_tags = []

         for key in clusters_dict[demo_clusters[1]].keys():
             for i in range(len(deletweet)):
                 if deletweet['id'][i] == key:
                     tweet = json.loads(deletweet['tweet'][i])
                     # pull out hashtags from tweet object
                     for tag in [tweet['entities']['hashtags'][i]['text'] \
                                 for i in range(len(tweet['entities']['hashtags']))]:
                         if tag not in cluster_tags:
                             cluster_tags.append(tag)

         for item in cluster_tags:
             print('#{}'.format(item))
```

```
#Yes2Wes
#MascotMania
#LittleLeague
```

Cluster 108 is a a successful example for the recommendation system. Each tweet has the hashtag #NoBillNoBreak, but one tweet in the cluster has 2 more that are highly applicable to the group, since all the tweets in the cluster deal with holding the floor to push for better gun safety policy: #NoMoreSilence and #HoldTheFloor.

```
In [32]: # look at the tweets in a cluster
         print('cluster_id: {}'.format(demo_clusters[3]))
         for key in clusters_dict[demo_clusters[3]].keys():
             print('{tweetid}: {tweettext}'.format(tweetid=key, \
                                                    tweettext=clusters_dict[demo_clusters[3]][key]))
```

```
cluster_id: 263
529816984637034496: RT @acberka: Just voted--now it's your turn! #republican #StandWithDan #goandvote h
523959270736674816: RT @BlueDevil83: @Hogan4Governor Your youngest fan!! HoganForMD http://t.co/BQJwjbN
407556287627399168: RT @smidgiekroger: @RepDennyHeck @RedCross check with your local American Legion Au
523620595289030656: RT @KeevAdams3: Let's get t shirts printed #288MillionWentWhere @Hogan4Governor
520017037645864960: RT @zapeters: Ran into Maryland's next Governor tonight - @Hogan4Governor #mdpoliti
```

```
In [33]: # extract hashtags from the tweets in the cluster
         cluster_tags = []

         for key in clusters_dict[demo_clusters[3]].keys():
             for i in range(len(deletweet)):
                 if deletweet['id'][i] == key:
                     tweet = json.loads(deletweet['tweet'][i])
                     # pull out hashtags from tweet object
                     for tag in [tweet['entities']['hashtags'][i]['text'] \
                                 for i in range(len(tweet['entities']['hashtags']))]:
                         if tag not in cluster_tags:
                             cluster_tags.append(tag)

         for item in cluster_tags:
             print('#{}'.format(item))
```

```
#republican
#StandWithDan
#goandvote
#288MillionWentWhere
#mdpolitics
```

Cluster 263 is a good example of a cluster with mixed success: #goandvote could apply to all the tweets; #mdpolitics and #288MillionWentWhere could apply to the 3 tweets that mention @HoganForGovernor; #StandWithDan is presumably not applicable to any of the other tweets in the cluster

```
In [34]: # look at the tweets in a cluster
         print('cluster_id: {}'.format(demo_clusters[5]))
         for key in clusters_dict[demo_clusters[5]].keys():
             print('{tweetid}: {tweettext}'.format(tweetid=key, \
                                                  tweettext=clusters_dict[demo_clusters[5]][key]))
```

```
cluster_id: 453
266882923960094720: Update on #A #subway: Read the latest on rebuilding in the #Rockaways after Hurrican
269122784813273089: 'We Will Lead on #Climate Change' | Read the Gov's op-ed in @nydailynews: http://t.
262232359192129536: Breaking: Gov's Dir of State Operations Howard Glazer, MTA Chairman Joseph Lhota, P
265215852331278337: Stay Up To Date on Hurricane Sandy Recovery Efforts | #Sandy
http://t.co/ATTnkJAg via @energy
```

```
In [35]: # extract hashtags from the tweets in the cluster
         cluster_tags = []

         for key in clusters_dict[demo_clusters[5]].keys():
             for i in range(len(deletweet)):
                 if deletweet['id'][i] == key:
                     tweet = json.loads(deletweet['tweet'][i])
                     # pull out hashtags from tweet object
                     for tag in [tweet['entities']['hashtags'][i]['text'] \
                                 for i in range(len(tweet['entities']['hashtags']))]:
                         if tag not in cluster_tags:
                             cluster_tags.append(tag)

         for item in cluster_tags:
             print('#{}'.format(item))
```

```
#A
#subway
#Rockaways
#sandy
#Climate
#Sandy
```

Cluster 453 is another example of a partially successful cluster: the first 2 hashtags are unlikely to be applicable to any of the other tweets, with the exception of potentially the 3rd tweet. However the hashtags #Climate and #Sandy have high likelihood of being applicable to all the tweets in the cluster.

While analyzing the clusters, an unforeseen situation arose in which a different kind of hashtag recommendation system might have some success. This is applicable to the clusters in which all the tweets have highly correlated content, but which have no existing hashtags. In this scenario, hashtags could be recommended based purely on the content of the clusters, most likely by turning a word common to all the tweets into a hashtag. This could be made more sophisticated and robust by searching in realtime to see if there are relevant popular hashtags on Twitter.

We do not try to implement such a system here, but present some examples for potential future consideration.

```
In [36]: hashtags_from_content = [22, 31, 34, 146]
```

```
In [37]: # look at the tweets in a cluster
         print('cluster_id: {}'.format(hashtags_from_content[0]))
         for key in clusters_dict[hashtags_from_content[0]].keys():
             print('{tweetid}: {tweettext}'.format(tweetid=key, \
                                                   tweettext=clusters_dict[hashtags_from_content[0]][ke
```

```
cluster_id: 22
630483076171309056: They oppose equal pay for equal work.
522718529364443136: 1.3 million homeless students... http://t.co/yaMbgiTQkN
519914410031067136: ...a pay raise for over 25 million American workers... http://t.co/M1E7WZne3R
453607544230248448: RT @RepKevinBrady: All people deserve Equal Pay for Equal Work. Period. http://t.co
```

cluster 22: #EqualPay4EqualWork

```
In [38]: # look at the tweets in a cluster
         print('cluster_id: {}'.format(hashtags_from_content[1]))
         for key in clusters_dict[hashtags_from_content[1]].keys():
             print('{tweetid}: {tweettext}'.format(tweetid=key, \
                                                   tweettext=clusters_dict[hashtags_from_content[1]][ke
```

```
cluster_id: 31
702975909566029824: Thank you for https://t.co/qTtuzipdS2
739253869050494977: @TeamTrumpNC thank you. https://t.co/YHF0wjEhhj
703104748153499650: Thank you! Trump2016 #GOPDebate https://t.co/aV9rR1zl7o
175614937446617088: @TheSmak Thank you!
630050327371321345: Thank you #CruzCrew! #RSG15 #CruzCountry http://t.co/auapF8wQAi
466422915215675392: @MJLeavitt thank you!
461248994035773441: Thank you... http://t.co/v4ALozeg6T http://t.co/DkFdb8Wuj2
375932319884128256: RT @jtylerharrison: @RepRickCrawford thank you
266934031541743617: THANK YOU - http://t.co/qj7HEXBd
466553301652090880: @Jennifer_K1691 thank you!
203471882912137216: @DanVForbes Thank you. Appreciate the message
312724010863575040: @USAFVeteran1 thank you!
476559119466258432: @PolitiBunny Thank you!
459093074636181505: This has been a blessing, thank you! http://t.co/dPVMqMCoPg
423663010217881600: Thank you...
161518376991207424: @Sloup Well, thank you 'Sloup'
451419821976977408: Thank you sir \@daaman81: @NealMarchbanks daaman81@gmail.com"
```

cluster 31: #thankyou

```
In [39]: # look at the tweets in a cluster
         print('cluster_id: {}'.format(hashtags_from_content[2]))
         for key in clusters_dict[hashtags_from_content[2]].keys():
             print('{tweetid}: {tweettext}'.format(tweetid=key, \
                                                   tweettext=clusters_dict[hashtags_from_content[2]][ke
```

```
cluster_id: 34
808806195523911680: I liked a @YouTube video https://t.co/c681wo0UT6 Jay Z Gets Embarrassed By An Old R
```

```
622157970084888576: Add a message to your video http://t.co/hpbd1mRVlI
603271586091835394: I added a video to a @YouTube playlist http://t.co/pStiw1iZZX Obama Adminstration Fa
456614333561061380: I added a video to a @YouTube playlist http://t.co/ra93f6Zhgz Dr. Chad Mathis: Leade
353243512831098881: I added a video to a @YouTube playlist http://t.co/omrRXEboOT Gov. Abercrombie Appoi
353243514148098048: I added a video to a @YouTube playlist http://t.co/lPrZ5Kb5p5 Senator Daniel K. Akak
590987507032002561: Into'd a #RachelCarson res., thanks to activists like her bald eagles are back. Stel
175302378776567808: I added a video to a @YouTube playlist http://t.co/igRQjoPI NBC Channel 11: Rep. Sco
353243514139717633: I added a video to a @YouTube playlist http://t.co/oZhzFRvNrY 6/19/12 News conferenc
603275295848816641: I added a video to a @YouTube playlist http://t.co/tws34arkHF Rep. Collins: This Rep
487245128919052289: I added a video to a @YouTube playlist http://t.co/XfUd36pWQO McDermott on Immigrati
144891694213644289: I liked a @YouTube video from @larrymendte http://t.co/K4znTAkI Let Buddy Roemer Deb
161828958894170114: I liked a @YouTube video http://t.co/DGz84Sw9 YouTube Town Hall: Where your view cou
353241077098098688: I added a video to a @YouTube playlist http://t.co/y4lZfH6EYW Bill Signing for HB430
251817162522648576: I added a video to a @YouTube playlist http://t.co/KzOw6ocH Bannock Development Corp
253186784421347328: I uploaded a @YouTube video http://t.co/olKeBAW4 Idaho Parks Passport Program
400383072689860608: We owe our nation's #veterans a tremendous debt of gratitude. I was honored to spend
603275278446665729: I added a video to a @YouTube playlist http://t.co/uP6AsrteaV Rep. Collins: High Edu
353241075823034369: I added a video to a @YouTube playlist http://t.co/FqzNRj5HiA Bill Signing SB856 (Ci
603275278484373504: I added a video to a @YouTube playlist http://t.co/5RQHYmnqPD Congressman Collins Ta
353241075525222400: I added a video to a @YouTube playlist http://t.co/KaNdORKuLl Kamuela Wassman Day
603275278429851648: I added a video to a @YouTube playlist http://t.co/SwesdmyNCh Collins speaks in favo
372746452612956161: I added a video to a @YouTube playlist http://t.co/w7A3KKPnD5 McNerney discusses the
175302379128881152: I added a video to a @YouTube playlist http://t.co/h9r2xBBd Rep. Scott Tipton Q&A du
603278040571977728: I added a video to a @YouTube playlist http://t.co/SaasQryIxz NSA: The New Four-Lett
372746721270706176: I added a video to a @YouTube playlist http://t.co/Oly82irmM8 McNerney recognizing A
157188236589006848: I uploaded a @YouTube video http://t.co/8yNbemAO Akin Update - KTRS Debate
353241534134632448: I added a video to a @YouTube playlist http://t.co/pGdfkkfQjq Sequestration Press Co
353241077144236032: I added a video to a @YouTube playlist http://t.co/uKLHbqvJCE Bill Signing, SB1077
175302378780758016: I added a video to a @YouTube playlist http://t.co/14ruCXMs Rep. Scott Tipton Recaps
353241077140029441: I added a video to a @YouTube playlist http://t.co/BjOeOr9erD Bill Signings, SB682
372743657604276224: I added a video to a @YouTube playlist http://t.co/dPQbSdHLtw Debate on Amendment to
175302378776563712: I added a video to a @YouTube playlist http://t.co/LPceXOSP Rep. Scott Tipton House
262577631596249088: I uploaded a @YouTube video http://t.co/muzgbLUp Two Big Differences
372746324112076800: I added a video to a @YouTube playlist http://t.co/85RDr8wiHg Congressman McNerney's
372728439146823680: I added a video to a @YouTube playlist http://t.co/50146GjvHV Rep McNerney calls for
175302378747207681: I added a video to a @YouTube playlist http://t.co/2l3ITfzM Rep. Scott Tipton at Sma
```

cluster 34: #youtube

```
In [40]: # look at the tweets in a cluster
         print('cluster_id: {}'.format(hashtags_from_content[3]))
         for key in clusters_dict[hashtags_from_content[3]].keys():
             print('{tweetid}: {tweettext}'.format(tweetid=key, \
                                                  tweettext=clusters_dict[hashtags_from_content[3]][key
```

```
cluster_id: 146
410547295571046400: In case you missed it from The Wall Street Journal: How to Keep Workers Unemployed:
368388131558785024: In case you missed it, see my guest appearance on PBS NewsHour discussing the suppre
335066013500579842: "In case you missed it, here's clip from ABC's "This Week" talking about moms in Con
142388919860867072: In case you missed it, we passed more #jobs bills out of @FinanceCmte yesterday http
```

cluster 146: #InCaseYouMissedIt

1.6 CONCLUSION

While some proof of concept work has been done toward a hashtag recommendation system, there are significant opportunities for improvement.

More specific text normalization has the potential to improve the robustness of the clusters. For example, links have very little information in this context, since they are generally shortened by twitter, and therefore any relevant text that may have existed in the original link is abstracted away.

More importantly, the current system is prohibitively computationally expensive, and it seems that this approach would require hardware resources that are currently inaccessible to the avaerage consumer. Therefore this system may only be applicable in a theoretical or research context, rather than a production environment.