Final Project Report
CS 674 - Big Data Analytics and Text Mining

Query Social Media (Twitter) with HashTag and Analyze them by using Microsoft Cognitive Services

Instructor - Gerard de Melo (Spring 2017)

Submitted By
Abu Awal Md Shoeb and **Shahab Raji**
March 26, 2017

Abstract

Hashtags are becoming an integral part of social media post. They often appear in twitter, facebook, and instagram. Since hashtags are unique in everywhere, it helps to find any relevant posts easily from multiple social media platforms. In this project, we analyzed social media data with hashtags. By using Microsoft Cognitive services, we also analyzed images that came up with a post and hashtag. We collected about half a million twitter data for our project and showed how to extract meaningful information from them. Our work focused on a group of hashtags that reflect different seasons i.e. Fall, Autumn, Spring, etc.. In this work, we picked #fallcolor, #fallcolors, and #autumn to identify fallcolor related post and images. By analyzing twitter data, we also showed how certain features of an image can be more useful to find relationship among twitter texts, images, and hashtags.

1 Introduction

Social media has been the most prominent way of expressing opinions. Along with publishing photos and sharing news, people also use their social media accounts (e.g. Twitter, Facebook, Instagram, Google+) for expressing their instant reaction about places, products, events, occasions etc. On social media sites, hashtags are one of the most popular ways to express sentiments about particular topics. A hashtag is a word or phrase preceded by a hash or pound sign ("#") and used to identify messages on a specific topic. These metadata tags are variously used to convey both emotion and contextual information about the content of a social media post. Along with hashtags, posts often contain images as well. In this project, we used both metadata and image of the post to extract meaningful information. From Microsoft Cognitive Services, we used their Vision API to derive additional features of twitter images. We showed how to gain more insights about a post by considering those additional features of images.

Later, we showed how all attributes can play an import role to find the relationship among a photo, hashtag and post. This relevance also helps us to understand whether the post is really relevant with the specific hashtag or not. At the end, we showed several visualizations of our data for specific hashtags. Our original goal was to apply the same technique over multiple social media platforms. However, due to the limitations of time, we only showed our work based on the twitter data.

Contributions - We analyzed twitter data and their images. We also combined Microsoft Vision Analyzer to our analysis. To the best of our knowledge, this is the first work that actually finds the correlation among twitter text, hashtag and image. People have done much research on twitter data but not on the images. People also have done research on hashtags in order to visualize them popularity, locations, trends, etc. However, nobody has done anything to show the relationship of text, hashtag and image appears in a twitter post.

2 Problem Definition and Methodology

Since the number of social media users has been increasing day by day, it has become easier than ever to reach out to vast audiences through social media. On top of that, the use of the hashtag has also increased. Because social media websites allow searches for content based on hashtags, users are able to find and connect with content that matches their interest. However, not all social media supports to query using hashtags. Also they are not capable of verifying whether those posts/photos are really relevant to the hashtag or not. This problem arises very often when a single post contains multiple hashtags. Additionally, there isn't any complete service available where a single query can be placed by one or multiple hashtags over multiple social media sites. Some free online services are available but user can not have an extensive report or dataset from them [18]. Similarly some paid services are available but their report are just based on the metadata of the post; they never considered to process the image that shows up with a post.

In this project we worked on Twitter data. Twitter and Instagram have their API to query with hashtag but Facebook does not[1][2]. However, there is a third party service where Facebook data can be gathered based on hashtag[3]. We collected data from Twitter by using hashtags. All posts in the data has certain attributes collected from the twitter directly. However, we also generated additional attributes of the post when the post appeared with an image. The Microsoft Cognitive Services [7] gives many useful information including Description, Tags, Dominant Color, etc. A full set of Visual Features can be found in Microsoft website [15]. Here are some features that were helpful in our analysis especially when we tried to find a Fall Color image.

- **Categories** - categorizes image content according to a taxonomy defined in documentation.
- **Tags** - tags the image with a detailed list of words related to the image content.
- **Description** - describes the image content with a complete English sentence.
- **Faces** - detects if faces are present. If present, generate coordinates, gender and age.
- **ImageType** - detects if image is clipart or a line drawing.
- **Color** - determines the accent color, dominant color, and whether an image is black & white.
- **Adult** - detects if the image is pornographic in nature (depicts nudity or a sex act). Sexually suggestive content is also detected.

At the end, we combined all attributes of such post to visualize useful information in different perspectives.

3 Our Work

In this section, we talked about the process of data collection and cleansing, the tools we used for our work, and the feature selection process.

3.1 Designed Platform for unrestricted Data Collection

Data collection is always a big challenge for a researcher especially if someone needs a specific set of data from the scratch. In our case, we didn't have any readymade dataset available to be used. This is why we had to collect all of our data from the scratch. We used several tools to gather and process the data. The current platform can be used to gather any number of tweets from any time period with no restrictions. We collected all data from twitter which was structured and partially labeled. Later, based on the twitter data, we also used Vision API provided by Microsoft Cognitive Services to analyze images [9].

3.1.1 Twitter Search API

Twitter API requires a user to register as a developer in order to use their API. Any Twitter user can create an application by following the link https://apps.twitter.com/. The user also needs to provide their phone number for verification. This leads the user to get API access that includes API Key, API Secret, Access Token and Access Token Secret. We followed an online tutorial to setup the Twitter API [4].

Step 1. Getting Twitter API Keys - We used our twitter account to access Twitter Apps [5] and created an application for our project.
Step 2. Installing Twitter Library - Python Twitter Tool [6] was downloaded and installed on both linux and mac machine. We used python 2.7 for our project.
Step 3. Collecting Data Using Twitter REST API - We used Twitter REST API which gives us access to real-time tweet. As mentioned before, the keys and tokens, gathered in step 1, are used here for authentication. For security reason, we erased our authentication keys in the following sample code. The following sample code returns 100 recent tweets in English language. For an instance, we used the hashtag #fallcolor in this example provided below.

Twitter REST API Sample Code:

```
# Import the necessary package to process data in JSON format
try:
    import json
except ImportError:
    import simplejson as json

# Import the necessary methods from "twitter" library
from twitter import Twitter, OAuth, TwitterHTTPError, TwitterStream

# Variables that contains the user credentials to access Twitter API
```

```
ACCESS_TOKEN = '' # OUR ACCESS TOKEN
ACCESS_SECRET = '' # OUR ACCESS TOKEN SECRET
CONSUMER_KEY = '' # OUR API KEY
CONSUMER_SECRET = '' # OUR API SECRET

oauth = OAuth(ACCESS_TOKEN, ACCESS_SECRET, CONSUMER_KEY, CONSUMER_SECRET)

# Initiate the connection to Twitter REST API
twitter = Twitter(auth=oauth)

# Search for latest tweets with HashTag "#fallcolor"
# twitter.search.tweets(q='#fallcolor')
iterator = twitter.search.tweets(q='#fallcolor', result_type='recent',
lang='en', count=100)

#print iterator
print json.dumps(iterator, indent=4)
```

The above code returns data in JSON format. We stored the data in a json file that can be imported into Spark for further use[12].

Above implementation doesn't work perfectly for the timeline based search. There are several other tools available to get twitter data through twitter api. We tried out most of the tools including tweepy. Unfortunately, none of them were good enough for timeline based search. On top of this, twitter api doesn't allow to fetch tweets that is older than two years. Similarly, for a single search query, twitter only gives 100 results of the current week. So all api based search tools should have the iterative capability to walk through the past to get older data. Initially, for the project, we used twitter search api and implemented a python program to collect data from Twitter. However, we found a better way later to collect twitter data. Next section will demonstrate how to gather data from Twitter using an online tool called GetOldTweets [19].

3.1.2 Get Old Tweets

We used a tool that collects tweets through a browser. When you enter on Twitter page a scroll loader starts. Then if you scroll down you start to get more and more tweets, all through calls to a JSON provider. This is obviously a good way to get old tweets especially when twitter api doesn't support a search past than two years. Also this search doesn't have any upper cap for search results. If you can see something on the browser then for sure you can see it through this tools. Unlike twitter search api, it doesn't require any authentication key which really extends the limit of getting unlimited tweets. We modified the original tool based on the requirements of the data we want. We also enhanced the

functionality of the program to make it more user friendly. The modified code is available in our GitLab repository.

3.1.3 Microsoft Computer Vision API

Microsoft Cognitive Services have various APIs that include Computer Vision API, Content Moderator API, Emotion API, Face API, Video API [10]. We specifically used Computer Vision API in python programming language [7].

Step 1. Getting Microsoft API Keys - We used our Hotmail account to get access keys for Microsoft API [13]. One account can have two different keys and each key has a limit of processing maximum 5000 images per month. To address this issue, we had to create several accounts to obtain several keys. We also had to pause our application in every single minute as the api doesn't allow to query more than 20 images per minute.

Step 2. Installing Microsoft Vision API for Python - We used Microsoft Vision API for Python. The library and setup instructions are available online [8]. We used python 2.7 for setting up the api on both linux and mac machine.

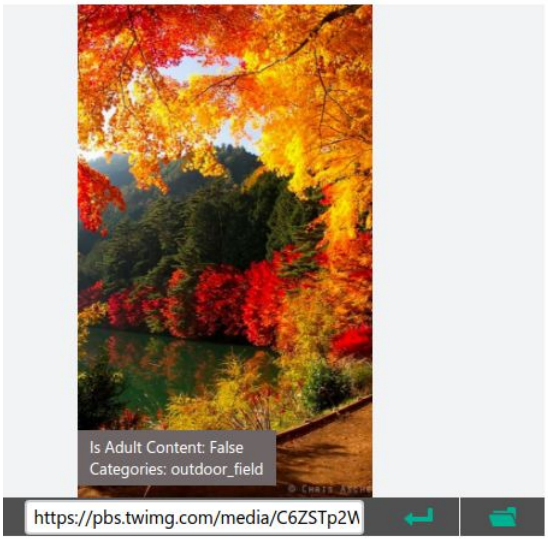
Step 3. Analyze Image Using Microsoft Vision API - After extracting data from Twitter, we applied Microsoft Vision API for analysing images. The following sample code shows how to process an image by Microsoft Vision API and get result in json format. Our modified codes are available in our GitLab repository that has the ability to process batch images.

Sample Code:

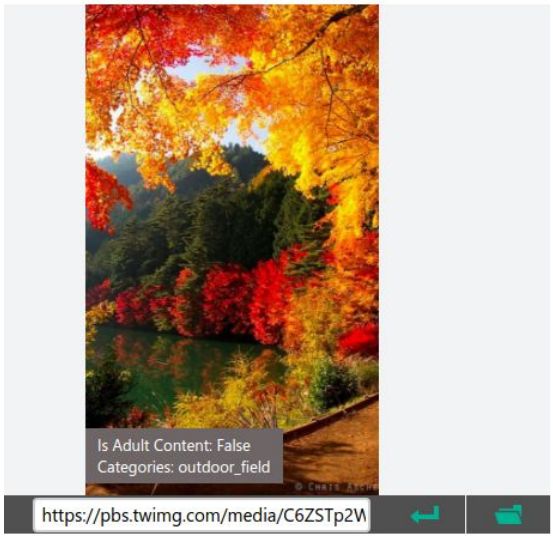
```
########### Python 2.7 #############
import httplib, urllib, base64
import json

headers = {
    # Request headers. Replace the key below with your subscription key.
    'Content-Type': 'application/json',
    'Ocp-Apim-Subscription-Key': 'YOUR-KEY',
}

params = urllib.urlencode({
    # Request parameters. All of them are optional.
    'visualFeatures':
'Categories,Tags,Description,Faces,ImageType,Color,Adult',
    'language': 'en',
})

# Provide image URL to be processed
body = "{'url':'https://pbs.twimg.com/media/C6ZSTp2WcAIwSQn.jpg'}"
```

```
try:
    conn = httplib.HTTPSConnection('westus.api.cognitive.microsoft.com')
    conn.request("POST", "/vision/v1.0/analyze?%s" % params, body,
headers)
    response = conn.getresponse()
    data = response.read()
    dataInJson = json.loads(data)
    print json.dumps(dataInJson, indent=4)
    conn.close()
except Exception as e:
    print("[Errno {0}] {1}".format(e.errno, e.strerror))
```

We picked a sample fall color image found in Twitter HashTag search [16]. Then, before we analyze an image in our implemented code, we analyze it in Microsoft API website and cross checked the results.



Feature Name	Value
Description	{ "type": 0, "captions": [{ "text": "a view of a plant photo of a tree", "confidence": 0.6209836246156966 }] }
Tags	[{ "name": "tree", "confidence": 0.9998486042022705 }, { "name": "plant", "confidence": 0.8048126101493835 }, { "name": "maple", "confidence": 0.5648788213729858, "hint": "tree" }, { "name": "forest", "confidence": 0.4051640033721924 }, { "name": "wooded", "confidence": 0.19361232221126556 }]
Image Format	Jpeg
Image Dimensions	537 x 900
Clip Art Type	0 Non-clipart
Line Drawing Type	0 Non-LineDrawing
Black & White Image	False
Is Adult Content	False



Black & White Image	False
Is Adult Content	False
Adult Score	0.009755419567227364
Is Racy Content	False
Racy Score	0.010690133087337017
Categories	[{ "name": "outdoor_field", "score": 0.75390625 }]
Faces	[]
Dominant Color Background	
Dominant Color Foreground	
Dominant Colors	
Accent Color	#BF990C

The description and tags of the photo tells us about the type of the image. The keywords and the confidence value are really helpful to identify an image. In this case, it gives us the keywords Tree, Forest, Maple with higher confidence. Similarly, it gives the information of dominant colors as well. In our implemented code, we were able to find the same results for the given image.

3.1.4 Importing data to Spark Notebook

The data collected in JSON from Twitter and Microsoft, is imported to Spark as Dataframes. One of the reasons that we decided to store everything is Spark is that for this project we have to handle a lot of files in various formats and aggregating everything in a Spark dataframe helps us to query and manipulate the data a lot more easily. We must, first, set up a SparkSession, clean up and reformat the JSON file and use Spark to read the data. In Spark version 2.x, we do not need to access SQLContext, SparkConf, SparkContext or HiveContext. All of these features are unified under SparkSession.

Step 1. We first set up a SparkSession in Spark Notebook.

```
val spark = SparkSession
   .builder()
   .appName("jsonReaderApp")
   .config("spark.sql.json.rdd", 1)
   .getOrCreate()
```

Step 2. The well-formatted JSON files from Twitter and Microsoft are not readable by Spark. For example, for JSON files, each line of the file must contain exactly one JSON object. We had to write a code in Python to reformat the JSON files.

Step 3. We import the JSON files to spark as dataframes.

```
val df = spark.read.json(path)
```

Step 4. After we input all the data to a dataframe in Spark, We prepare and preprocess the data for Machine Learning Algorithms.

The above process along with the setup for ML methods is explained, in detail, in *Appendix A* at the end of this report.

3.2 Tools and Library - Setup and Installation

We mostly used python and Spark Notebook for implementing our project. In order to use our python code, following python library should be installed on the system.

- PyQuery - a jquery-like library for python that allows to make jquery queries on xml documents.
- BeautifulSoup - a python library for pulling data out of HTML and XML files.
- urllib - a package that requests for opening and reading URLs in python.
- csv - a package for python to access .csv files.

We installed all libraries for python 2.7 via pip (a package management system for python). We tested our code on both Ubuntu and MAC operating systems.

3.3 Our Dataset

We collected around 500K twitter data in total. This data was collected based on the hashtags #fallcolor, #fallcolors, and #autumn. Since our goal was identifying a real fallcolor post, we focused on the month between September and November. We collected data for both 2015 and 2016. All data points should have following attributes.

- username - It is the name of the twitter user ie the userid, not the screen name.
- date - It represents the date of tweet in central time zone and the format is yyyy-mm-dd h:m.
- retweets - It tells how many time the post was retweeted.
- favourites - It represents the number how many time people liked it.
- texts - It is the full text of the tweet. It includes all hashtags and the name of the other twitter account that was mentioned in the post.
- geo - The geo location of the post.
- mentions - It contains other twitter id if they are mentioned in the post.
- hashtags - All hashtags that are appeared in the post.
- Id - The unique twitter id of a post.
- permalink - The url of the corresponding post.

- imageURL - The url of the image that was attached to the post.

In twitter, it is not obvious to have an image in a post. In fact, the probability of having a photo with a post is almost fifty percent. For example, when we acquired data for the month of September 2016, we found only 43185 posts have images out of total 100K tweets. This is why, in our data set, you may not find imageUrl for all posts.

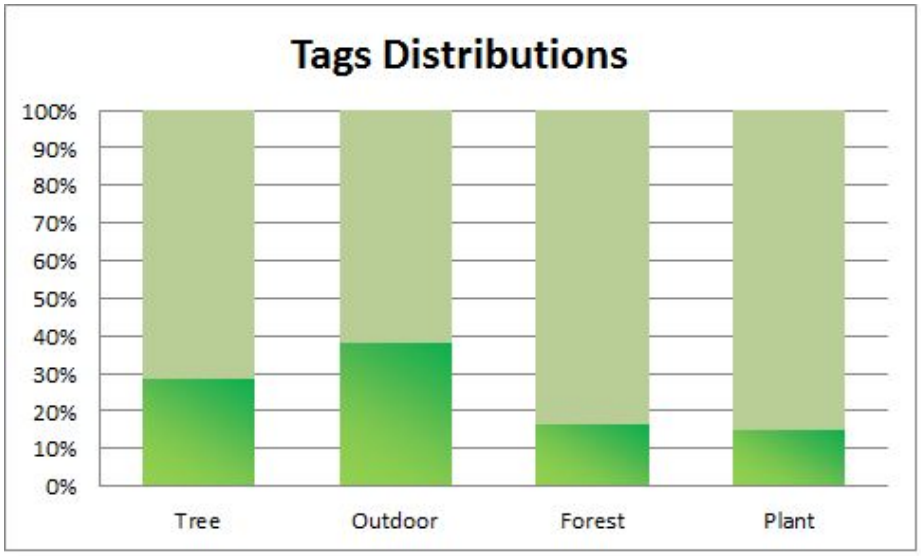
3.4 Feature Selection

By analyzing the output of the Microsoft Vision API, we selected several components as our features to identify an image as real fall color image. Their analyzer gives relevant tags, description, dominant colors of the image. By observing several real fallcolor images, we found tree, forest, outdoor, plant are most obvious tags for fallcolor with high confidence value. Similarly, for the dominant colors, we found red, orange, yellow, and brown are the most dominant colors for a real fallcolor image. We also considered the true false value of clipart and drawing attributes that helped us to eliminate colorful paintings that are not real fallcolor image at all.

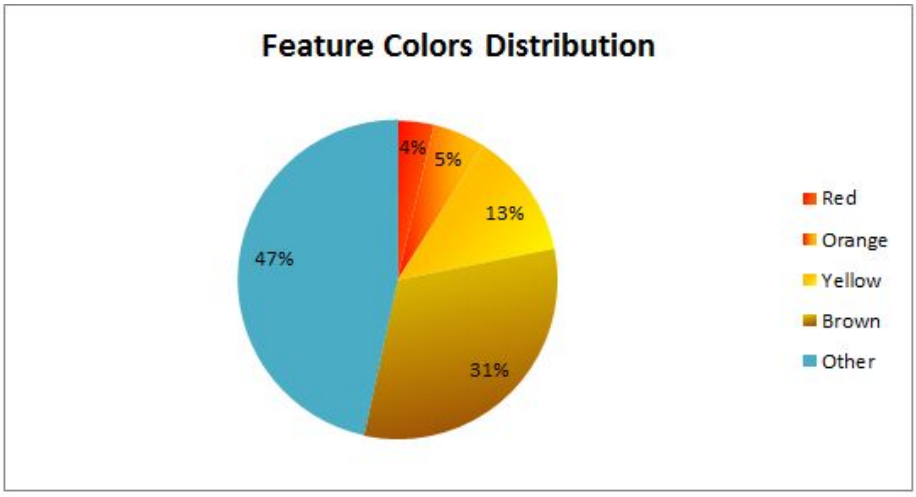
3.5 Results

In this section, we display some of the results that lead to selection of the features for images. Based on these results, we eliminated some of the labels (tags) that had very low frequency, like “leaves”. Also for colors, at the start, we only chose Yellow, Red and Orange. But after looking at the processed data we realized that Brown is also a very frequent dominant color.

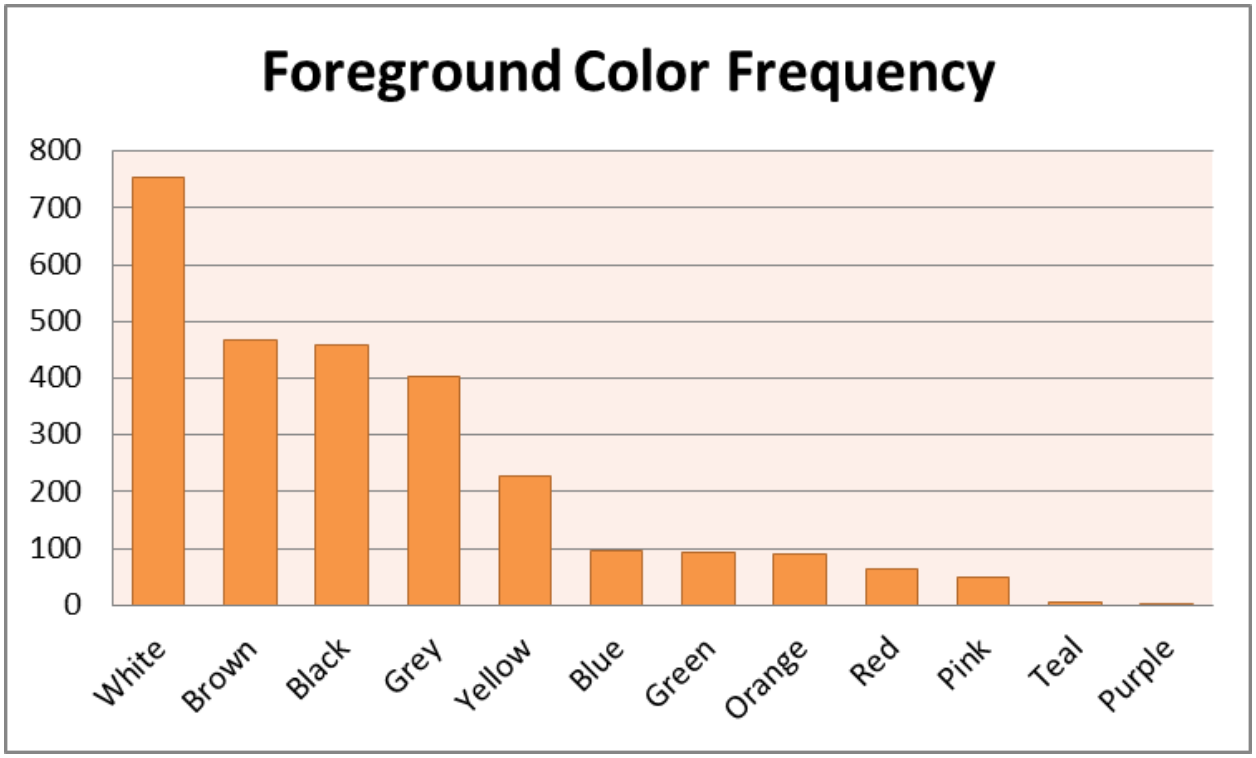
Feature Tags Distribution - This figure shows what percentage of our train data was tagged with the feature tags. These tags also has high frequency in our dataset.

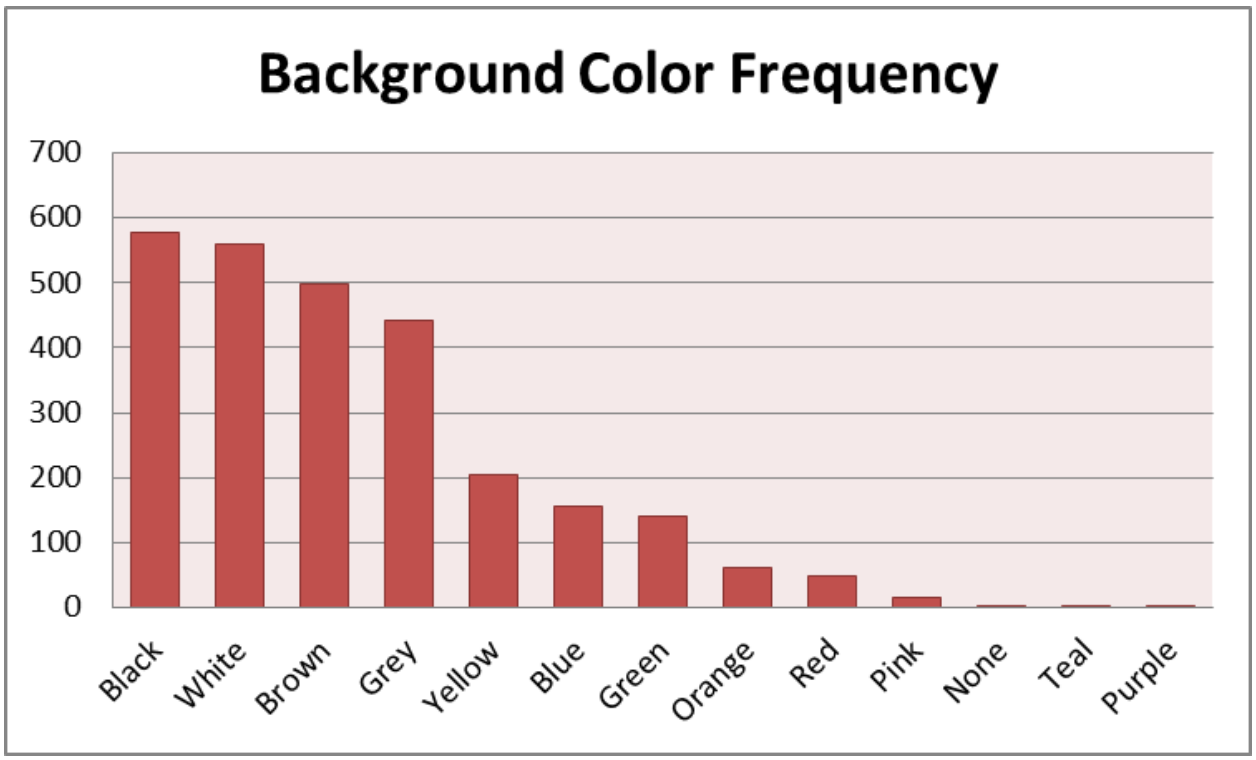


Feature Colors Distribution - This figure shows the distribution of our feature colors as dominant colors in the dataset. Dominant colors in the Microsoft API output is an array consist of Foreground Dominant Color, Background Dominant Color and Accent Color.



Foreground Colors - This figure shows the distribution of the Foreground and Background colors. Even Though neutral colors, like White, Black and Grey, have high frequency they were ignored since they do not carry the characteristics of autumn.


Foreground Color Frequency
800
700
600
500
400
300
200
100
0
White
Brown
Black
Grey
Yellow
Blue
Green
Orange
Red
Pink
Teal
Purple


Background Color Frequency
700
600
500
400
300
200
100
0
Black
White
Brown
Grey
Yellow
Blue
Green
Orange
Red
Pink
None
Teal
Purple

Relevant Tweets - Out of 500,000 tweets collected, we processed around 110,000. A little more than 40% of them (47065) were tweets with images. Filtering all the tweets that include only #fallcolor and #fallcolors, we arrived at a collection of 2707 tweets.

5 Discussion

Why hashtag - Hashtag based search is a good way to find social media post on a specific subject matter. First of all, hashtags are unique. For example, in our case, we use both #fallcolor and #fallcolors. During hashtag search, it exactly matches the hashtag, it never does a substring matching. Thus, a post with #fallcolor never showed up when we searched #fallcolors. Secondly, a user may have different userid in different social media platforms. Therefore, the user can post a same message on different platforms but it will appear under different userid. However, if the post has same hashtag then they can be searched with that hashtag regardless of userid and social media platforms.

Why not only considering the metadata but also the photo - When we thought about social media posts, we thought about the contents of a post. Along with text, hashtags, locations, and metadata, a photo is an essential component of a post as well. A good number of posts on social media shows up with photos. As we were collecting data of social media, we were thinking how to extract more information out of it. Then we decided to take advantage of the photo that comes up with a post. When a user post something on their social media, they attach a photo which is quite relevant to the post.

Why Microsoft Cognitive Services - First of all it's free for processing a small number of images. Secondly, since our goal was to analyze social media data, we didn't want to spend time to analyze the image of the post. So we found Microsoft Cognitive Services that actually provides several APIs to process images. They have Face API, emotion API, Vision API etc. We specifically used their Vision API which returns information about visual content found in an image. This api uses tagging, descriptions and domain-specific models to identify content and label it with confidence. Their vision feature analyzer tells us image types, color schemes of an image which helps us in feature selection.

We aimed to extract more features of an image that can help to relate the image with corresponding post. For example, when we search for posts of fallcolor with the hashtag #fallcolor, we always look for the colors in the image that represent the fallcolor. The Microsoft image analyzer gives the dominant color of an image with confidence. It also gives relevant tags of the image with confidence. In our case, we consider them as strong feature to identify a fallcolor post and image. The api also helps us to identify a clipart or drawing which is an obvious feature to decide on real fallcolor image.

Some images have dominant fall colors like real fallcolor images but they are actually not [17]. For an instance, one woman was wearing a colorful scarf that has all dominant colors of fall. However, the tags of the image were women, clothing, etc. which didn't include any tag of fallcolor image like tree, leaves, plants, forest, etc.

What about the resolution of the photo - For image analysis, it is expected to have a good resolution photo for a better result. In our data collection process, we found a trick that enabled us to access high resolution photo of the corresponding twitter post. Each image has a unique twitter url that is different from that twitter post. If we add a colon sign followed by large (:large) at the end of the image url, then the high resolution image becomes accessible [17]. Since we used this high resolution image in Microsoft Vision API, it always gave us a better result comparing to the low resolution image.

6 Future Work

Our initial goal was to combine hashtag data from multiple social media platforms. Due to the imitation of the time, our application can now collect data only from twitter. However, Since our system is ready to accept more data, we will continue our work to gather data from instagram, facebook, and google+. Furthermore, we will continue to develop a web based service where anyone can query with hashtag and get exact results we showed in this project.

For now, we are only analysing images collected from Twitter. Later, we will gather more images from Facebook and Instagram along with text and metadata. At the end, we are hoping to have a complete dataset for several HashTags. We will use both metadata and results generated by Microsoft API in order to make our decision to rank popular posts/tweets/images. Our application will also have a web interface to visualize all results.

7 Conclusion

This project leads us to several research ideas. We have also learned a lot by collecting and analyzing twitter data from the scratch. Our findings also proved that hashtags, appeared in a tweet, may be irrelevant to the post. Similarly, the attached image in a post is also misleading. We will continue working to develop a model that will help to identify misleading hashtags and images of social media posts.

Acknowledgement

We thank our instructor for letting us choose a project by ourselves. His critical reasoning helps us to to think about the research side of the project, rather than just as a class project.

References

[1] https://dev.twitter.com/rest/public/search
[2] https://www.instagram.com/developer/endpoints/tags/
[3]https://thenextweb.com/facebook/2015/04/03/facebook-might-be-shutting-down-access-to-hashtags-in-its-api/#.tnw_EPquT439
[4] http://socialmedia-class.org/twittertutorial.html
[5] https://apps.twitter.com
[6] https://pypi.python.org/pypi/twitter
[7] https://www.microsoft.com/cognitive-services/en-us/
[8] https://github.com/Microsoft/Cognitive-Face-Python
[9] https://www.microsoft.com/cognitive-services/en-us/computer-vision-api
[10] https://www.microsoft.com/cognitive-services/en-us/apis
[11] https://gitlab.com/abushoeb/cs674
[12] https://gitlab.com/abushoeb/cs674/blob/master/twitter/fallcolor-100-tweets-26-march-2017.json
[13] https://www.microsoft.com/cognitive-services/en-US/subscriptions
[14] https://portalstoragewuprod2.azureedge.net/vision/Analysis/1.jpg
[15] https://westus.dev.cognitive.microsoft.com/docs/services/56f91f2d778daf23d8ec6739/operations/56f91f2e778daf14a499e1fa
[16] https://pbs.twimg.com/media/C6ZSTp2WcAIwSQn.jpg
[17] https://pbs.twimg.com/media/CtijjtgXgAU50s_.jpg:large
[18] http://keyhole.co/preview
[19] https://github.com/Jefferson-Henrique/GetOldTweets-python

Appendix A

cs674_Final

CS674:Big Data Analysis

By Shahab Raji and Abu Awal Md Shoeb

Storing the Data

Preparation to read JSON format and aggregate all data from Twitter. Spark Session is used to read data from files.

```
val mySession = SparkSession
   .builder()
   .appName("jsonReaderApp")
   .config("spark.sql.json.rdd", 1)
   .getOrCreate()

val df = mySession.read.json(jsonFile)
```

With Spark Session, you can also change the input format and read CSV or Text. To read a CSV files we use the following code:

```
val df = mySession.read.option("header","true").option("delimiter",";").csv(path)
```

Creating desired Dataframe from the Twitter data

```
val columns = Seq($"id", $"created_at", explode($"entities.media.media_url")
                  ,$"geo", $"entities.urls.url", $"id_str", $"user.screen_name")

val sum = df.select(columns: _*)
```

Data from twitter then will be plugged in the code to get the Microsoft API inforamtion. After collecting the data we read it and store it as another dataframe.

Microsoft Dataframe

```
val expMsDF = msDf.withColumn("url", $"url")
  .withColumn("dominantColorFore", $"msResults.color.dominantColorForeground")
  .withColumn("dominantColorBack", $"msResults.color.dominantColorBackGround")
  .withColumn("isBW", $"msResults.color.isBWImg")
  .withColumn("isGraphic", $"msResults.ImageType.clipArtType")
  .withColumn("reqId", $"msResults.requestId")
  .withColumn("tags",$"msResults.description.tags")
  .select("url","reqId","tags","dominantColorFore","dominantColorBack")
```

Data Aggregation

The reason we store everything in Spark Dataframes is that later data manipulation from different sources and different formats is much easier in Spark, than other platforms. As shown here:

```
val output = df1.join(df2,Seq("url"),joinType="inner")
```

```
val output = df1.union(df2).union(df3).union(df4)
```

```
val output = df.filter($"hashtags".contains("#myHashtag"))
```

Data Export

We can export the dataframes in different formats.

```
df.coalesce(1).write.option("header","true").format("csv").save(path)
```

For JSON files you can use `repartition(1)` instead of `coalesce(1)`.

Preparing the Data for Machine Learning Libraries

Spark has libraries for almost every ML algorithm. But to prepare and preprocess the data for those libraries we need to convert data to Vectors and Matrices; what Follows is the procedure.

Libraries

Vectors and Matrices are usually used for clustering algorithms. And Labeled Points are for classification algorithms.

```
import org.apache.spark.mllib.regression.LabeledPoint
import org.apache.spark.mllib.linalg.{Matrix, Matrices}
import org.apache.spark.mllib.linalg.Vectors
```

Clustering with K-mean

Here we explain how to set up Spark for K-mean Algorithm.

```
import org.apache.spark.ml.feature.VectorAssembler
import org.apache.spark.ml.clustering.KMeans
```

From the dataframe, we create a table that includes our features.

```
val trainData = df.withColumn("url",$"imageUrl")
  .withColumn("dcRed", when(array_contains($"msResults.color.dominantColors","Red"),1.0
  .withColumn("dcOrange", when(array_contains($"msResults.color.dominantColors","Orange
  .withColumn("dcYellow", when(array_contains($"msResults.color.dominantColors","Yellow
  .withColumn("dcBrown", when(array_contains($"msResults.color.dominantColors","Brown")
  .withColumn("tagTree", when(array_contains($"msResults.description.tags","tree"),1.0)
  .withColumn("tagLeaves", when(array_contains($"msResults.description.tags","leaves"),
  .withColumn("tagOutdoor", when(array_contains($"msResults.description.tags","outdoor"
  .withColumn("tagPlant", when(array_contains($"msResults.description.tags","plant"),1.
  .withColumn("tagForest", when(array_contains($"msResults.description.tags","forest"),
  .withColumn("isClipArt", when($"msResults.imageType.clipArtType">0,1.0).otherwise(0.0
  .withColumn("isLineDrawing", when($"msResults.imageType.lineDrawingType">0,1.0).other
  .withColumn("isBW", when($"msResults.color.isBWImg",1.0).otherwise(0.0))
  .select("url","dcRed","dcOrange","dcYellow","dcBrown","tagTree","tagLeaves","tagOutdo
```

But to prepare the data for Kmean library, we have to change each row to a `Vector`. we use `VectorAssembler` for this purpose.

```
val assembler = new VectorAssembler()
  .setInputCols(Array("dcRed","dcOrange","dcYellow","tagTree","tagLeaves","tagOutdoor",
  .setOutputCol("features")

val kmeanInput  = assembler.transform(traindata).select("features")
```

And Now the input is ready for Kmean.

```
val kmeans = new KMeans().setK(2).setSeed(1L)
val model = kmeans.fit(kmeanInput)
```

```python
########### Python 2.7 #############
import httplib, urllib, base64
import json, time
import sys
import csv

headers = {
    # Request headers. Replace the key below with your subscription key.
    'Content-Type': 'application/json',
    #'Ocp-Apim-Subscription-Key': ''
    'Ocp-Apim-Subscription-Key': '',
}

params = urllib.urlencode({
    # Request parameters. All of them are optional.
    'visualFeatures': 'Categories,Tags,Description,Faces,ImageType,Color,Adult',
    #'details': '{string}',
    'language': 'en',
})

# Replace the three dots below with the URL of a JPEG image of a celebrity.
# Abu Shoeb modified on 2017-03-25
#body = "{'url':'...'}"
#body = "{'url':'https://portalstoragewuprod2.azureedge.net/vision/Analysis/1.jpg'}"
body = "{'url':'https://pbs.twimg.com/media/C6ZSTp2WcAIwSQn.jpg'}"

# Read input CSV file name from user
inputFileName = sys.argv[1]
outputFileName = sys.argv[1].strip()[:-4]+"-microsoft-results.json"
#outputFileName = sys.argv[1].strip()+"-microsoft-results"
#f = open('data/twitter-image-urls-result.json', 'w')
f = open(outputFileName, 'w')

try:
    conn = httplib.HTTPSConnection('westus.api.cognitive.microsoft.com')
    # 2017-04-12 : Abu Shoeb modified
    # process all urls from file
    #print '{\n  "obj": ['
    f.write('{\n  "obj": [\n')

    with open(inputFileName, 'r') as infile:
        #reader = csv.DictReader(infile)
        # Read imageUrl from input CSV
        #reader = csv.reader(infile, delimiter=',')
        reader = csv.reader(infile)
        # Skip header
        next(reader, None)
        for row in reader:
            imgUrl = row[2].strip()
            twId = row[0].strip()
            userId = row[1].strip()
            # Skip URL for thumbnils
            if "ext tw video thumb" in imgUrl:
                print "imgUrl : ext tw video thumb --> SKIP"
                continue
            elif "No meta image given" in imgUrl:
                print "imgUrl : No meta image given --> SKIP"
                continue
            #body = "{'url':'"+row[2].strip()[1:-1]+"'}"
            body = "{'url':'"+imgUrl+"'}"
            conn.request("POST", "/vision/v1.0/analyze?%s" % params, body, headers)
            response = conn.getresponse()
            print "STATUS : "+str(response.status)
            print "URL : "+imgUrl
            while response.status != 200:
```

```
                print "IN WHILE"
                data = response.read()
                #print data
                #print data['message']
                #{ "statusCode": 429, "message": "Rate limit is exceeded. Try again in 20 seconds." }
                #print "Sleep 7 seconds"
                #time.sleep(7)
                #sleepTime = 0
                sleepTime = [int(s) for s in data.split() if s.isdigit()]
                print "Sleep "+str(sleepTime)+" seconds"
                time.sleep(sleepTime[0])
                conn.request("POST", "/vision/v1.0/analyze?%s" % params, body, headers)
                response = conn.getresponse()
            data = response.read()
            #print data
            dataInJson = json.loads(data)
            tmp = json.dumps(dataInJson, indent=4)
            #print "----- Result for : "+str(line)
            # add url to the json data
            #print '    {\n      "url": "'+line[1:-2]+'",\n      "msResults":  '+tmp+'\n    },'
            f.write('{\n      "tweetId": "'+twId+'", \n "userId": "'+userId+'", \n "imageUrl": "'+imgUrl+'",\n "msResults":  '+tmp+'\n    },')
            #f.write('    {\n      "url": "'+imgUrl+'",\n      "msResults":  '+tmp+'\n    },')
            #print json.dumps(dataInJson, indent=4)
    conn.close()
    #print '  ]\n}'
    f.write('  ]\n}')
    f.close()
except Exception as e:
    print "ERROR"
    #print("[Errno {0}] {1}".format(e.errno, e.strerror))

####################################

############ Python 3.2 #############
'''import http.client, urllib.request, urllib.parse, urllib.error, base64

headers = {
    # Request headers. Replace the key below with your subscription key.
    'Content-Type': 'application/json',
    'Ocp-Apim-Subscription-Key': 'e8c73ae2c7fd4795802da8c468f7d628',
}

params = urllib.parse.urlencode({
    # Request parameters. All of them are optional.
    'visualFeatures': 'Categories',
    'details': 'Celebrities',
    'language': 'en',
})

# Replace the three dots below with the URL of a JPEG image of a celebrity.
body = "{'url':'http://0.tqn.com/d/gosoutheast/1/S/F/r/-/-/NC-fall-color-tail-of-dragon-robbinsville-vnc-1500.jpg'}"

try:
    conn = http.client.HTTPSConnection('westus.api.cognitive.microsoft.com')
    conn.request("POST", "/vision/v1.0/analyze?%s" % params, body, headers)
    response = conn.getresponse()
    data = response.read()
    print(data)
    conn.close()
```

```
except Exception as e:
    print("[Errno {0}] {1}".format(e.errno, e.strerror))'''
####################################
```

```
#importing the libraries
from urllib import urlopen
from bs4 import BeautifulSoup
import csv
import sys


inputFileName = sys.argv[1]
outputFileName = inputFileName.strip()[:-4]+"-images.csv"

with open(inputFileName, 'r') as infile, open(outputFileName, 'w') as outfile:
    fieldnames = ['twitId', 'userName','imageUrl']
    writer = csv.DictWriter(outfile, fieldnames=fieldnames)
    writer.writeheader()

    # Read input CSV
    #with open(sys.argv[1], 'r') as infile:
    reader = csv.reader(infile, delimiter=';')
    #spamreader = csv.reader(csvfile, delimiter=' ', quotechar='|')
    # Skip header
    next(reader, None)
    for row in reader:
        #print len(row)
        # Skip row if it has incomplete twit info
        if len(row) == 11:
            twitId = row[8]
            userName = row[10]
            url = 'https://twitter.com/'+str(userName)+'/status/'+str(twitId)
            webpage = urlopen(url).read()
            soup = BeautifulSoup(webpage, "lxml")

            # Get meta data
            title = soup.find("meta",  property="og:title")
            url = soup.find("meta",  property="og:url")
            imageUrl = soup.find("meta",  property="og:image")

            # Print meta data
            #print(title["content"] if title else "No meta title given")
            #print(url["content"] if url else "No meta url given")
            if imageUrl:
                if "https://pbs.twimg.com/media" in imageUrl['content']:
                    print(imageUrl["content"] if imageUrl else "No meta image given")
                    #with open(sys.argv[2], 'a') as outfile:
                    #fieldnames = ['twitId', 'userName','imageUrl']
                    writer = csv.DictWriter(outfile, fieldnames=fieldnames)
                    writer.writerow({'twitId': twitId, 'userName': userName,
                    'imageUrl': imageUrl["content"]})
            else:
                print("No meta image given")
                #with open(sys.argv[2], 'a') as outfile:
                    #fieldnames = ['twitId', 'userName','imageUrl']
                writer = csv.DictWriter(outfile, fieldnames=fieldnames)
                writer.writerow({'twitId': twitId, 'userName': userName, 'imageUrl':
                'No meta image given'})


```