INFO 290T - Data Mining Final Project
Trends from Google, Youtube, Instagram, Yahoo: Datamined

Introduction

In popular media, there is much talk about social networking trends, and "what's trending" online. For our final project, our team was interested in exploring trends published on social media platforms. Specifically, our group wanted to explore the trends published on Google hot trends, the Twitter trends API, the Instagram 'most popular' API, Yahoo! trends and YouTube trends.

Specifically, our group aimed to answer the following questions:
I) Do similar internet trends appear across all platforms? II) How can we identify whether or not two 'trends' are actually different expressions of some underlying fundamental 'conceptual trend'? Can we identify this underlying 'parent' trend? III) How similar are web platforms to one another, as far as what is trending on a given day? Does this fluctuate over time? IV) Can we find out what 'category' or 'subject matter' a given trend is about? If so, can we compare platforms to one another to see if their 'category fingerprints' are similar?

To address the above questions, our group broke down our project into three components. We created a series of extract transform and load (ETL) tools to aggregate and collect social networking trends and then group like trend topics by their relative similarity. At this point, we passed these outputs through to a code that calculated the similarity of platforms to one another according to what trends they shared in common. Below, we give a description of the methods we used to develop and implement each component, the problems we encountered and the insights/findings that we gained from working on this project.

1. Getting the Data: ETL Tools and Web Scraping

As previously mentioned, Google, Twitter, Instagram, Yahoo, and Youtube, all publish information about self-reported top trends; however, prior to this project, no person or organization known to us had aggregated this data into a single source, let alone attempted a process to do so in an automated way. Our team took this as a challenge and developed a series of tools and processes to gather this disparate information and transform it into a usable data set. Applying this ETL process, we were able to collect data 22 days, from April 22 through May 14, and data collection continues to be ongoing.

Process

For each of the five social networks, Janine wrote customized object oriented ruby scripts that obtained the self-reported top trends from the social network's API, or in cases where no API existed, scraped the website where the trends were published directly. As the data was collected, it was cleaned up into a standardized format and then dumped into separate json files. For each trend item reported by a given social network, we collected the title of the trend, the network it appeared, the time and date when the trend appeared, the country/geolocation that trend originated in, and the number of likes, searches, or views the trend had received. (To see the exact data collected and sources of this data, please refer to Table 1.1)

The scrape tools were hosted on an Amazon ec2 server. A series of Crontab jobs enabled us to automate the process of collecting data. Due to the nature of “trending” data, the self reported top trends on each social network change multiple times of day. To capture these changes, we set Crontab jobs to run each data collection Ruby script every four hours; at the end of each day, we had 6 json files for each social network. Using Crontab, we ran a script each night that combined these smaller files into a larger aggregate json file for each social network for the day. The use of automation via Crontab allowed us to collect internet trend data 24/7.

Challenges

Several aspects of our ETL tools evolved from one of Janine’s class projects from the fall; unfortunately, most of this code turned out to be unusable and had to be re-written. Parts of the code from Janine’s earlier work was unstable and inefficient and/or served a purposes that were different purpose than needed for this project. It was challenging to have to review and revise old code and either have to scrap it or heavily revise it.

The most difficult part of this project turned out to be the various complexities required by the web scraping. When possible, we tried to make API calls to collect the data, however, only Twitter and Instagram publish their self-reported via an API. To collect trend data from Google, YouTube, and Yahoo, we had to extract from the website HTML directly. The process of scraping was multilayered owing to the fact that javascript renders the HTML for websites directly to the browser -- meaning that a simple wget call to the website will not return the actual display HTML for the page. Additionally, some of the data we collected appeared on multiple pages of a given platform. To circumvent these complications, we used the Ruby library called Headless and Watir. These libraries enabled us to create an in-shell instance of a FireFox browser. From within the inshell instance of FireFox, we could make HTTP get requests to Google, Youtube, and Yahoo. The website’s javascript would then render the HTML into the in shell instance of firefox, and which point, we could parse the HTML from page using other ruby HTML parsing library. The ruby watir library is a webdriver library that enabled us grab trend data from multiple pages; specifically, Watir enabled us to automate the process of automate the process of clicking links and menu buttons to navigate through a series of webpages, so that we could collect trending data from each page.

Lessons Learned

Using the library s3fuse enabled us to write our trends data directly to s3 data bucket rather than write to our ec2 server. Writing to s3 directly minimized the chance for data loss in the event that our ec2 ever went down; it also saved us money server costs, as it cheaper to pay for s3 storage space than amazon EBS storage.

Additionally, Creating a completely automated process to collect data was painstaking and time consuming, however, the overall benefit was worth it; was the ETL was up and running, we could “set it and forget it.” (A great tip we learned from this semester!)

2. Trends and Parent Trends: Jaccard to the Rescue

Process, Goals

One question that we hoped to answer was, “Do trends regularly appear on some platforms before appearing on others?” Before we could begin analysis of this question, Xavier aimed to write a python script that would take the json file output from each platform for each day and combine them into at least two dictionaries addressable in the following ways:

- Input: Platform Name | Output: List of Unique Terms for each day for that social network (essentially a dictionary of the format { PlatformName : {Date: set(Terms)}, {Date:... })
-Input: Trend Term | Output: For each day, whether it appeared on a platform or not (essentially a dictionary of the format { TrendTerm: {Date: Google:0, Yahooo:1,Instagram:0}, {Date:...} }

Parsing and correctly structuring the data into the dictionary formats was greatly slowed down by the fact that different platforms provided data with unpredictable encoding formats, and in many cases attempting to encode these formats would fail. This is the point at which Murphy’s Law first made it’s appearance in our project. After extensive troubleshooting, it was decided that non-encodable terms would be stored in a separate dictionary (these mostly include foreign alphabets like Arabic) for analysis later, while the targeted data would be output to two separate json files, for ease of import and use in the next section.
This portion of the project was implemented in Python, and there is an additional goal of implementing it in MapReduce, though for the purpose of this project deliverable that was not necessary. Once the data was in the correct dictionary format, we could begin our analysis.

As happens often in projects like these, we encountered a stumbling block early on that ended up preventing us from accomplishing this particular aspect of our goals. In this case, the problem of the diversity with which a trend might present across websites eluded us entirely until we tried it out. We wrote and ran a code that evaluated which terms appeared on more than one platform only to discover that only 6 of some 4,000 terms made more than one appearance over the totality of days and platforms. After much code checking, it was only then that we realized the depth of a new issue: even on the same platform, these sites made no effort to normalize terms, if they happened to trend under different spellings. Having learned of different ways of calculating similarity of two document vectors in class, the team realized that calculating the Jaccard coefficient for each individual ‘trend’ formulation against each other term might reveal which trends were likely to represent the same underlying trend underneath.

Results

To calculate a Jaccard score for each term against one another, we relied on running each term in our ‘All Unique Terms’ list output from the previous process against one another. At the time, we had a total of 4,400 unique terms - meaning that there were a possible over 9,000,000 different permutations possible. After removing certain trends that seemed to reflect the presence of terms that did not encode correctly, we put 3528 terms through the code over some 10 hours and were pleasantly surprised with our results.

Ultimately, we found were able to successfully compare and identify groupings of terms present within the trends we extracted. Having looked over the results for different steps of Jaccard score, we settled on a Jaccard similarity of 0.7 or over as expressing with strong confidence that the terms involved represented the same underlying trend. Here is a sample output:

Parent Trend:
bayern real
Child Trends:
['real madrid', 'real bayern', 'real madrid vs bayern munich', 'bayern munich vs real madrid']

Parent Trend:
star wars episode
Child Trends:
['star wars ', 'star wars 7']

Overall, we discovered that some terms with a Jaccard Similarity of over 0.7 and at least one match accounted for 17% of all terms (600/3528). Taken together, we were very pleased to have discovered and implemented a way to group together trends so as to identify them in a variety of different iterations across networks.

4. Measures of Similarity between Social Network Trends

One of the goal of project was to find out how similar web platforms as far as what is trending on a given day, as well as how this similarity fluctuates over time. To answer these questions, we choose to implement the cosine similarity and Jaccard similarity algorithms in Mr. Job to calculate the similarity of trending topics between each social network for each day. We also wrote a python script that runs the runs mr. job over a batch of data files, calculates the cosine and Jaccard similarity measure for each day and then writes the resulting measures to csv files.

We felt that it was important to implement both the Jaccard similarity and cosine similarity each measures a different aspect of similarity. Jaccard similarity measures the existence of terms in present in two documents against all other terms that two documents don’t share. Cosine similarity takes into account the length of the document and the frequency of the number of times the terms appears in the document. The tables below show the cosine and jaccuard values when comparing the trend terms between each social network. Since a trend term is a very specific instance, we also calculated the cosine and jaccard values of individual words that appeared on social networks to get a overall measure of the content that appeared in each social network. For full results, please see our spreadsheet, results_all_day.xlsx

Cosine Results - Comparing the trends of each social network as a document

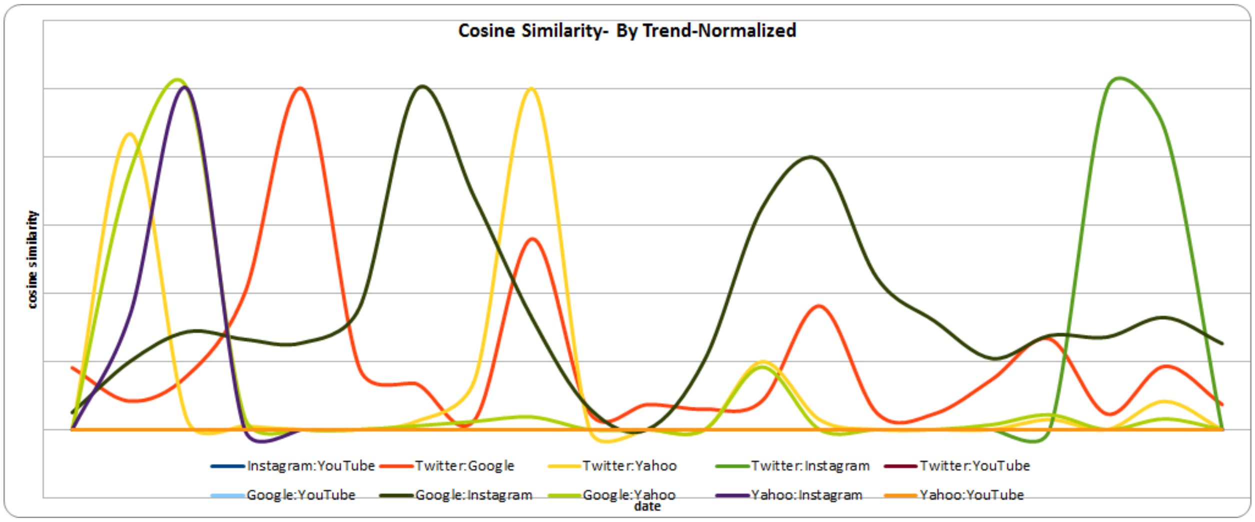
network_combination	Cosine-Trend –Normalized Avg	Cosine-Word-Normalized Avg
Instagram:YouTube	0	0.7020404286
Twitter:Google	0.1982389524	0.3387584286
Twitter:Yahoo	0.115133619	0.1937661905
Twitter:Instagram	0.0895792381	0.1678589524
Twitter:YouTube	0	0.3476499048
Google:YouTube	0	0.5141721905
Google:Instagram	0.344177	0.4553547619
Google:Yahoo	0.101621	0.2878583333
Yahoo:Instagram	0.0635414762	0.1886628571
Yahoo:YouTube	0	0.2068022381

In the above table, trending topics on Instagram and Google and Instagram tend to have the most overlap. Additionally, trending topics on Twitter and Google also tend to also have some overlap. When one looks at the words that appeared in trending topics on each social network, Instagram and YouTube are the most similar.

Jaccard Results- Comparing the trend terms on each social network as a document

network_combination	Jaccard-Trend –Normalized Avg	Jaccard-Word-Normalized Avg
Instagram:YouTube	0	0.5386800952
Twitter:Google	0.4505527143	0.5476936667
Twitter:Yahoo	0.3211435238	0.4660362857
Twitter:Instagram	0.0911710476	0.5471145238
Twitter:YouTube	0	0.5035234762
Google:YouTube	0	0.4998411429
Google:Instagram	0.2303976667	0.3726265238
Google:Yahoo	0.1779825714	0.4779102381
Yahoo:Instagram	0.0942320476	0.353606381
Yahoo:YouTube	0	0.532532619

The results from running Jacard are similar to Cosine, however, Jaccard indicates that Google and Twitter have the most similar trend topics. Here, using Jaccard reveals that a surface level, Google and Twitter have the most number of terms that exist on both platforms, however, at a deeper level, the cosine similarity results reveal that the two are less similar when the length of document and the frequency terms is considered.



The graph above shows the normalized cosine similarity values for the 22 day period in which we collected data. The graph above, as well as the as well other graphs of our results (in the spreadsheet results_all_day.xlsx) show that similarities between trending topics on social network platforms fluctuate from day to day.

6. Findings, Insights

At the outset of this project, we asked:
I) Do similar internet trends appear across all platforms?
II) How can we identify whether or not two 'trends' are actually different expressions of some underlying fundamental ‘conceptual trend’? Can we identify this underlying ‘parent’ trend?
III) How similar are web platforms to one another, as far as what is trending on a given day? Does this fluctuate over time?

as well as

IV) Can we find out what ‘category’ or ‘subject matter’ a given trend is about? If so, can we compare platforms to one another to see if their ‘category fingerprints’ are similar?

Our project was a lesson learned in how much effort is required to answer these questions. After all of our effort, it seems that in answering questions II and III), we have made great headway to finding the answer to questions I). At the outset, we had identified question IV) as falling below the priority of answering the first three questions -- but nevertheless, we learned a great deal about this task as well. We did not successfully match each term to a given subject matter, but the code we developed for this project has demonstrated that this is possible and sets the stage for future explorations in datamining these categories for this project in the coming months.
We have successfully built an automated process for extracting data from very different web interfaces, we have established and tested code for identifying trends that represent the same

‘parent trend’ and we have been taught a great lesson in humility regarding our assumptions that ‘trends’ are as unique as they are easy to track across the web.

The lessons of this project have not given us direct access to the statistics we sought, but they did reveal a great deal about the nature of trends, and gave us a great opportunity to directly apply datamining techniques towards uncovering the results we had originally sought out. We have included in our folder an excel file containing the results of comparisons of social networks to one another, as well as the code we wrote for each step of the process. We are proud of the work we’ve done for this project, and hope the sum total reflects the extent to which we’ve taken the lesson of the class to heart and applied them, to the best of our ability.

6. Future Work

At the outset, we had hoped to identify whether it might turn out that certain platforms tend to present with some trending terms and influence the terms that present on other platforms. While we successfully developed the code that outputs dummy variables for the presence of given trending terms for each given day, a decision was made to hold off from any correlation analysis until such a time as parent trends could be identified (finding that only 6 of some 4,000 terms matched for more than one network over all of our data seemed to be sufficient justification for holding off!). Now that we have established a way to identify parent trends across social networks, the immediate next step would be to modify the code so that its output can be assessed for its Pearson’s R^2 value in a first time, and for additional regression tests in addition to that.

Additionally, our research shows that just looking at the similarity of trending topics might not yield any similarity, as trend topics represent a very specific instance of something. In thinking about trends, we thought it would be interesting to find the category of each trending topic. This lead us to ask the question, “Can we find out what ‘category’ or ‘subject matter’ a given trend is about? If so, can we compare platforms to one another to see if their ‘category fingerprints’ are similar?”

We believe that the freebase API (http://wiki.freebase.com/wiki/Freebase_API) could help automate the process of discovering ‘category’ or the ‘subject matter’ of a given trend. The Freebase API has a semantic search functionality in which one calls the API with a given string of text; the API returns the likely categories that string of text is associated with.
We were able to develop a ruby script that grabs a trend term and runs it through the freebase API to obtain category information associated with that given term. During this project, we tried to run this script on all the trend terms we had collected, but after running the script continuously for two days; we realized that this approach would take too long because simply had too many terms to look up. As part of this project, we developed a way to group together trends so as to identify them in a variety of different iterations across networks. In the future, we believe that we could use this grouping to limit the number of trend terms that we run through the Freebase API to obtain category information about trending topics.

Appendix

Table 1- Self-Reported Trending Data Collected: Description and Source

Social Network	Data Location	Data Collected
Twitter	Twitter API resource, GET trends/place	● trend location, trend title, time of trend, href link to trend o twitter ● data available for 412 different geographic locations
Google	http://www.google.com/trends/hottrends	● trend location,trend title, trend rank, search count, link to trend image, href link to website about trend, time of trend ● data available for 40+ countries
Youtube	http://www.youtube.com/trendsdashboard	● trend location, trend title, number of views, link to trending video, time of trend ● data available for 40+ countries
Instagram	Instagram API resource, GET Most Popular	● trend location, trend/photo caption, trend/photo tags, href link to trending photo, likes count, time of trend ● data available for a variety of lat/long coordinates
Facebook	www.facebook.com facebook homepage (located in the upper right corner of the user's homepage	● trend rank, trend title, description of trend, href link to photo of the trend, href link to trend news story, time of trend ● specific location data not available
Yahoo	www.yahoo.com (trends located on right side of page)	● trend rank, trend title, time of trend ● specific location data not available

Division of Labor:

Team Member	Task	Associated Code File Name Worked On (attached)	Sample Output

Xavier	Extract Terms and Encode Them, Matter the Encoding format	getparentdict->Transform1.py	UniqueTerms=[term1, term2,term3]
			BadEncodingTerms=[term1,term2]

Xavier	Create Dictionaries by Platform, by Trend Term	getparentdict->Transform2.py	{ PlatformName : {Date: set(Terms)}, {Date:... }) { TrendTerm: {Date: Google:0, Yahooo:1,Instagram:0}, {Date:...} }
Xavier	Create 'Parent Trend Term' Analysis Script, Using Jaccard Similarity to determing and output Parent term and All Child Terms	getpartentdict->ParentTrends.py	
Janine	Platform Scraping	see folder ETL	→ ETL folder contains many classes and scripts
Janine	Final Cosine Similarity and Jaccard Analysis Script for Comparing Networks to one another	see folder similarity_measure	folder contains several python scripts to run Mr Jobs for cosine and jaccard. The full results are also this folder
Janine	Future work- categories	see folder- future_get_categories	contains scripts to get freebase categories
Irina	MapReduce for comparing plarforms' trends to one another	similarity.py	