Lab 5, HTML, Mashups, Madlibs and Mockery

Due date: Monday, April 25, 11:59am

Lab Assignment

Assignment Preparation

This is a team lab. We have 27 people in our class. Please form nine teams of three people each. For this lab I want Statistics majors to join separate teams. There are no other restrictions on team lineup. I want teams formed by the end of the *April 13* lab period. Anyone missing from that lab period will be assigned to a team by me.

Each team shall email me the team lineup by the end of the *April 13* lab period.

Note: You are expected to work together on solving the problems assigned to each team. You will *eventually* split the work, and each of you will wind up developing on your own part of the code for your team, but I want all teams to be in tight communication over the major design and implementation issues, and to be present when the actual analyses using the developed code are performed. I want teams discussing the results of analyses before the report is written.

Assignment Overview

For this assignment you are asked to complete a number of data collection and analytical tasks that originate in accessing HTML documents on the World Wide Web and extracting information from them. Some of the specifics of the tasks may allow for *reuse* of Lab 4 code, which, if you need it, is *allowed*.

Task: The Pulse of the Nation. You will use a popular news aggregation site as a starting point to collect information about the important news stories of the day (you will have a choice of political, technology, media or celebrity gossip news stories as your target, although due to the related nature of the sites, your news discovery software should work on any of the sites without changes). You will present an analysis of current news topics as well as short term trends (you would need to collect about a week worth of data from the news aggregator site) in the news.

The task will involve the following stages:

1. **Data Collection.** You are provided with the web site information for starting your data collection. You may need to write some code to collect and crawl extra URLs from the same web site in order to build the appropriate set of HTML documents for processing.
2. **Data Cleaning.** The HTML documents you are collecting will contain a lot of content not necessary for the purposes of our assignments. You will write software to extract only the information that is of interest to you w.r.t. each specific task from each HTML document you are analyzing. *As a*

simplification, you are expected to work with HTML documents that all have the same format, so a single HTML parser/scraper per task should suffice.

3. **Data Modeling.** For each task, you must collect very specific information from the raw HTML documents. This information shall be stored in a collection of data structures that makes it easily accessible for processing and analysis.

4. **Data Analysis.** Most of the analysis you need to perform for this assignment is relatively straightforward: frequency analysis of various terms, and some analysis of data over time for one of the tasks. Where necessary, the details are described below.

5. **Data Visualization.** You will present your results as a number of graphs, charts and diagrams as appropriate. For this assignment, if you already know how to use `matplotlib`, feel free to us it. If you **have not** used `matplotlib` before, use the data visualization tools you are familiar with (Excel or other spreadsheet software, matlab, R, or anything else you want).

6. **Discussion.** You will submit a report discussing what you have observed in the data.

Things you need to know. To succesfully complete this lab, you will need to work with the following features of `Python`.

1. HTML file downloaders. In `Python3` the library you will need to use is `urllib.request`. Information about downloading HTML documents is provided in the lecture notes.

2. Raw HTML parser. Raw HTML parsers parse incoming HTML documents. While you will not be working directly with the raw HTML parser, `BeautifulSoup`, the DOM-style HTML processor that you will be using *does* need a raw HTML to work properly. The default HTML parser used by `Python3` is `html.parser`. A somewhat better parser, `libxml` is available in some cases as well, and if available, can be used.

3. `BeautifulSoup`. `BeautifulSoup` is an HTML processing library that represents an input HTML document in a form of a (DOM) parse tree and provides simple and powerful means of navigating the HTML parse tree and searching it for necessary information. You will make extensive use of `BeautifulSoup` in your code.

4. Dictionaries. To store information about the frequencies of occurrences of various terms, bigrams, and so on, you will need to use `Python` dictionaries. As an aside, for storing information about bigrams (and other multi-grams, if you need those for your analysis) you will need to use `Python` tuples as the dictionary keys. It also goes without saying that `Python` lists are going to feature prominently in this assignment.

Analysis of Current News

General Idea. Monitoring of the news stories in attempts to find out the proverbial pulse of the nation (or the world) is a popular pastime in data analytics. You will attempt to gain some insight into the realm of daily news cycles by collecting the information about the news stories that appear to be important and by looking into the nature of these stories.

Data Sources. Your main data source for the news stories is a collection of news aggregation websites created and maintained by Gabe Rivera[1] and his team:

Site name	URL	Type of news
Techmeme	http://www.techmeme.com	Technology news
Memeorandum	http://www.memeorandum.com	Political news
Mediagazer	http://mediagazer.com	Media news
WeSmirch	http://wesmirch.com	Celebrity gossip

In addition to accessing the front pages of the sites, you may benefit from accessing the following pages:

- The River. Gabe Rivera calls the reverse chronological list of discovered news stories the river. This list is available by adding the `/river` suffix to any of the front page URLs. For example, for Memeorandum, its river is available here:

  http://www.memeorandum.com/river

- Snapshots. Each site is updated as soon as news stories are discovered and deemed worthy of placing on the front page. Fortunately for you, you can find out how the front page of each site looked like at any particular moment of time, going all the way back to the origin of the site. To achieve this, you need to add the following suffix to the url of the website:

  ```
  /<yy><mm><dd>/h<hh><mm>
  ```

  Here:

parameter	explanation
<yy>	two last digits of the year
<mm>	two-digit month code (01–12)
<dd>	two-digit day (01–31)
<hh>	two-digit military hour (00–23)
<mm>	two-digit minutes (00-59)

  For example, if you want to see how TechMeme looked like on April 1, 2014 at 1:25pm, use the following URL:

  ```
  http://www.techmeme.com/160401/h1325
  ```

  Each river entry contains the author of the story (the by-line), the web site where the story is published, the title of the story and the discovery time. The river typically contains about seven days worth of discovered stories.

  Note: These URLs work at the granularity of 5 minutes. So, trying to retreive the information for `/h1323` (1:23pm) will not work.

News Aggregation. Each of the websites listed above aggregates news stories that appear in on-line news outlets: news websites, newspaper web sites, on-line news organizations, blogs, etc. The sites are powered by a ”secret sauce” process that combines automated determination of the importance of the news story to the current daily cycle (this largely depends on the authority of the news sources that publish the story and the number of times the story is picked up and references in other news sources) with manual curation of the stories which supports some of the sites (techmeme for example). Each time a visitor downloads

[1] Full disclosure: in the days of yore, Gabe and I were office mates at the University of Maryland.

the front page of one of the sites, (s)he observes a ranked list of currently developing news stories. In addition to the original (or the most informative) news article about a given news story, the site may contain information about other articles supporting the news story, as well as link to more coverage of the story on a variety of sites where this coverage appeared. By looking at the number of corroborating news articles one can determine how wide the story is spreading in the news media. By looking at the front page of the site at different times during the day, one can see how an individual story is progressing through time. A proprietary algorithm is set to ”sunset” the stories after a certain period of time, if new coverage for the story does not appear. If a certain news story line emerges (e.g., Donald Trump's performance at a Republican debate, or Google's announcement of new features of one of their services) over time, the original news article may be replaced with newer news articles as the story progresses from day to day.

What we are interested in. We are interested in analyzing the news cycles and determining what the important news stories are about. Most of the questions below can be asked about any of the four types of news aggregated by the sites you will be working with.

Question 1: What are today's stories about? This question can be answered by analyzing the headlines of today's stories, determining the key words and phrases found in these headlines, and collecting the information about the frequency of occurrences of the specific words/phrases in the headlines[2].

Question 2: How are today's stories compared with stories from yesterday, the day before yesterday, and so on? To answer this question, the frequency of term occurrence analysis needs to be performed for a few previous days, comparisons between the data collected for the different days can needs to be made.

Question 3: What themes/topics persist from one news cycle to another? For different types of news, this can be approached somewhat differently (celebrity gossip is person-driven, while technology news is more company-driven, but in both cases there are cross-cutting concerns: e.g., celebrity divorces, or mobile applications). In general, to answer this question, you should consider the data collected on different days, find similar themes in the data, and track them over time.

Question 4: How do specific topics progress through the news cycles over the period of study? For each type of news, there are some topics that are of special interest to outside observers. For example, topics related to the Republican presedential primary races may of an interesting political topic to track, while Apple's progress with mobile devices may be an interesting technology topic. Finding news stories relevant to a specific topic (based on the headline text), and then tracking the number of occurrences of such stories will do the trick here.

Question 5: Which news sites appear to be influential? Gabe Rivera has his own proprietary way of figuring out which of the news organizations are thought leaders. We will not try to reverse-engineer what he does. Instead, we can simply observe how many top new articles (i.e. articles whose headlines get published on the aggregator sites) each news organization produces in a single news cycle as well as over a period of time. Additionally, we can see how many references to each news site (actual links) are found on the front pages of the news aggregators throughout the same periods of time. This may provide a crude estimate of the influence of a news organization.

[2]For the core of this assignment, you will be working with the headlines, i.e. the article titles published on the aggregator sites. You are not expected to scrape the text of the news stories themselves, at least not yet in this assignment.

Data Collection and Data Cleaning

The first step of your process is to collect the necessary data and to extract the information you need from it. You will proceed as follows.

HTML scraper. Using `urllib.request` library to access the URLs on the World Wide Web, and using `BeautifulSoup` to traverse the HTML parse tree (essentially a DOM tree representing the HTML document you downloaded), build an HTML scraper tool that searches through the HTML tree and extracts from it the following information:

- Titles of all news stories available on the front page. Our primary concern is to capture news stories found in the main frame of each of the sites. Additionally, you can capture the titles of the articles located in the "Latest News Finder", "More News" and "Earlier Picks" areas of the site. You can also access the river pages to see if any article titles found there can be added to your discoveries.
- Names of the news sites that produced the leading news articles about a news story.
- For each collected news story, names of all news sites that featured a related story, as aggregated on the front page. Notice that a link to a news orgaization under a specific story line is a link to the actual story, so you can use these URLs for unique identification of articles that cover a specific news story.

All of the information collected above must be associated with a specific day and time of day. To capture a full picture of a news cycle, you may want to collect the data from more than one page during the day. For example, you may choose to collect the data from the 9:00am (morning), 12:00pm (lunchtime), 4:00pm (late afternoon) and 9:00pm (late evening - end of news cycle) versions of the aggregator front page. When doing so, *please make sure* to collect all headlines **exactly once**: if the same news story, with the same headline is available in multiple snapshots, only one copy of its title should be collected for future analysis. Note however, that the same story (i.e., the same actual news article URL) may be associated with different headline text throughout the day (the website's human editors may change the headline to make it more readable, or the news organization itself may do that), in which case, you can collect all versions of the headline, and connect them to the same article.

Stopword removal. For this assignment, you are allowed to create your own list of stopwords. In fact, when working with multiple aggregators, you may want to tailor the list of stopwords to the specific nature of the aggregator. Certain words that may be strong keywords for one type of news, are so common in another, that they can be considered stopwords. Feel free to design your lists of keywords to suit your intuition about what you need and do not need to capture.

Data Modeling

What are we trying to model. To successfully answer the questions asked above, we need to construct three models:

1. **Model of an individual news story.**
2. **Model of a news cycle/day.**
3. **Model of a news organization.**

These models are discussed below.

Individual news story model. For the purpose of this lab, a *news story* is defined as an actual event that received covereage by the news organizations in a form of one or more *news articles* published on the World Wide Web and discovered by the appropriate news aggregator. For us, the news story is represented by the following information:

- The collection of news article headlines that were published on the front page of the news aggregator.
- The set of all "leading" articles about the news story: i.e., the articles whose headlines were published.
- The full set of discovered articles covering the story.

Our model of a news story faithfully reflects the observations above. It formally shall consist of the following components:

1. **A bag of words representation** of the collection of headlines associated with the news story. You can use *term frequency* as the weights of each individual keyword. No need to use inverse document frequency as it is very difficult to determine in applications with dynamic content.
2. **A list of URLs** for the leading news articles about each news story. Each URL should be annotated with the name of the news organization/site that published it. For example, you may use Python tuples to represent such information: `(URL, newsOrg)`, or you can use simple dictionaries:

   ```
   {URL: <actualURL>,
   Site: <SiteName>
   }
   ```

3. **A list of URLs and news sites** for all news articles about each news story. You can use similar techniques for storing this information as for storing the list of leading news articles.

All of this can be combined into a single data structure at the end - e.g., a tuple, a list, or a dictionary. Once constructed, the representations do not have to be *mutable*, so using tuples is ok.

Collecting the data. As mentioned above, in order to construct a proper representation of a news story throughout its life cycle, you may want to take multiple snapshots of the front page of a given aggregator site throughout a single day. The question arises:

How do we match news stories across different snapshots of the front page?

The difficulty arises from two facts: (a) the position of the news story on the front page may change, and (b) the leading article for the news story may change.

To properly match news stories, use the following set of rules. This set of rules may not be able to cover all circumstances, but it covers most, and it requires only *simple computations*.

1. Let U_1 be the URL of a lead article for a news story from the aggregator front page at time t_1, and U_2 be a URL for a lead article for a news story from the aggregator page at time t_2. If $U_1 = U_2$, then the news stories headlines by U_1 and U_2 match. Notice, that this is true *regardless of the actual text of the headline used on each of the front pages*. The latter can differ, but as long as the URLs are the same, it is the same news story.

2. Often, an important news story is represented by multiple URLs on a single web page - more than one article has its headline published. This is done in a hierarchical way. Let $U_1, \ldots U_k$ be a list of URLs of headline articles for a news story collected from the front page of an aggregator at time t_1. Let U^* be the headline URL for a story from the front page at time t_2. If U^* is equal to some U_i for some $i \in 1, \ldots, k$, then we can match the news stories represented by $U_1, \ldots, U_k$ and U^*.

   The latter is true even if the aggregator site decided to *split* the original news story into two or more on the later web site.

3. If a news story has lead story URLs $U_1, \ldots, U_k$ representing it, based on a sequence of front page snapshots collected at times $t_1, \ldots, t_m$, and a news story is represented by URLs $U_1^*, \ldots, U_l^*$ on the front page collected at time t_{m+1}, **and** the two sets of URLs **are disjoint**, then, we say that they represent **different news stories**, *even if in the future, these stories can potentially be linked.*

This set of rules is *easy to implement and to apply to the incoming data*. It may assign the same headline to multiple news stories under some edge case-style circumstances. However, under most normal scenarios of news story development, it will match each URL to exactly one news story.

Model of a news cycle. The model of a news cycle is essentiall the union of all models representing individual news stories that happned in this cycle. This model consists of the following items:

1. The bag-of-words representation of all news article headlines for the news cycle/day. Achieved by simply combining the bag-of-words vectors for each of the news stories. Recall that *term frequencies* are additive, so we can simply add the bag-of-words vectors to obtain the representation of a single news cycle/day.

2. The aggregated news organization activity. We do not need to store individual URLs in the news cycle model. Instead, you will simply store the list of news sites featured on the front page of the aggregator during the day, together with **the number of unique lead article URLs** from each organization as well as **the number of all article URLs** from each organization (not that the latter must count the former as its part). That is, you can represent each organization as a triple:

   ```
   (siteName, NumberLeadArticles, NumberAllArticles)
   ```

   This triple can be represented in any way you want in Python: as a single tuple with three elements (as suggested above); as a dictionary entry of the form

   ```
   siteName:  (NumberLeadArticles, NumberAllArticles),
   ```

   as dictionary entries

   `siteName:NumberLeadArticles` and `siteName:NumberAllArticles`

   that are parts of two *different dictionaries*, and so on. You are responsible for picking the representation for your software.

Model of a news organization. A news organization persists over time, and each day contributes news articles to the news cycle. To represent the news organization/news site, you will map each news cycle – represented by a single date – to the news organization's participation in discourse. You will store the following information about each site:

1. Site name as used by the aggregator portal.
2. Front page URL of the site. You will have to properly determine it by taking apart the URLs of news stories (or via other means).
3. Activity. The activity of a single news organization is represented is a map from dates to the lists of unique URLs of news articles published on the site and linked from the front page of the news aggregator. You will, in fact, maintain two such maps: one for the headline news articles, and one for all other news articles.
4. Aggregate activity. For each news cycle/date, store two numbers: the number of headline articles placed on the aggregator portal that day and the total number of articles (including the headline ones) linked from the aggregator site.

Note: If you are analyzing multiple aggregator portals, you may find that certain news sites show up more than one aggregator. For the sake of simplicity, you shall maintain a **separate aggregator portal-specific** model of each news site in such cases (that is, you can perform all you analyses related to one aggregator portal in isolation from what you do for another).

Data Analysis

After constructing the data models and representing them as collections of Python data structures, you are ready for data analysis.

Question 1: What are today's stories about? To answer this question, perform the following analytical tasks.

1. **Frequent keywords.** Represent each news cycle/day by a collection of frequently found in the day's news stories keywords.

   Each team can decide on how many keywords to use to represent a given day, as well as how frequent a keyword needs to be in order to be reported. The reported words should not contain any stopwords or words that are otherwise not very strong indicators of what the stories are about. Your decisions though must be consistently applied to each day's worth of data (that is, you cannot just arbitrarily decide that for day 1 you will use 10 keywords, but for day 2 - 17).

2. **News site participation.** Report the most active news sites of the day - both in terms of lead articles as well as in terms of total articles indexed by the aggregator portal.

The answers to this question are straightforward displays of somewhat filtered data collected in a representation of a single day/news cycle.

Question 2: How are today's stories compared with stories from yesterday, the day before yesterday, and so on? To address this question, first and foremost, you need to look at the news story content.

The most straightforward way to address this question is to compare the bag-of-words representations of every pair of news cycles for which you have collected the data. You can use the *cosine similarity* coefficient as a similarity measure. As a reminder, given two vectors, $d_1 = (x_1, \ldots, x_n)$ and $d_2 = (y_1, \ldots, y_n)$, the *cosine similarity* between them can be computed as follows:

$$sim(d_1, d_2) = \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

You can determine the threshold of the *cosine similarity* that corresponds to a decision that two news cycles share a significant number of common news stories.

Simply determining that two days/news cycles are similar though is not sufficient. You also need to attempt to determine what similar themes are present on both days. To do so, you can look at the two bags-of-words representations of the daily headlines, and find common, frequently occurring keywords. This can be done in a number of ways. A common way is to compute the *bag intersection* of the two representations.

Given two bags X and Y, where $X(a)$ is the number of occurrences of a in X and $Y(a)$ is the number of occurrences of a in Y, the *bag intersection* Z of X and Y, denoted $Z = X \cap Y$ is defined as follows:

$$Z(a) = \min(X(a), Y(a)).$$

After computing the bag intersection, you can determine any thresholds on the frequency of common term occurrences that you find necessary prior to reporting the common terms/keywords in the news stories for both days.

An alternative way to perform the same analysis is to compare individual news story representations from the two days to each other, rather than the representations of the full news cycles. This requires more intricacy, and more work, but may ultimately lead you to better understanding of which stories were common - a high similarity coefficient between two stories suggests commonality of a theme.

Question 3: What themes/topics persist from one news cycle to another? This question is an extension of the previous one. Rather than looking at the common topics present in a pair of days, you are asked to track popular topics through a given period of time (e.g., a week).

One way to represent common topics, is to concentrate on a number of keywords that appear with high frequencies in all or some of the news cycles, and create a temporal mapping from each day for which you have data to the frequency of occurrence of the topic.

You can represent each keyword as a vector of frequencies for each day:

$$t = (tf_1, \ldots, tf_k),$$

where for $i = 1, \ldots, k$, tf_i is the frequency of occurrence of term t in headlines on day day_i.

Using such representation, you can compare the patterns of term occurrences in news stories for similar trends. As before, you can use *cosine similarity* for it. However, you can also use *Pearson correlation*, which is typically considered a better measure of similarity under the given set of circumstances. Given two vectors, $t_1 = (tf_1, \ldots, tf_k)$ and $t_2 = (tf'_1, \ldots, tf'_k)$ representing the presence of two terms t_1 and t_2 in the news discourse, the Pearson correlation between them can be found as follows:

$$Pearson(t_1, t_2) = \frac{\sum_{i=1}^{k}(tf_i - \mu_{t_1}) \cdot (tf'_i - \mu_{t_2})}{\sqrt{\sum_{i=1}^{k}(tf_i - \mu_{t_1})^2}\sqrt{\sum_{i=1}^{k}(tf'_i - \mu_{t_2})^2}},$$

where μ_{t_1} and μ_{t_2} are the mean values of tf_is and tf'_is respectively:

$$\mu_{t_1} = \frac{1}{k}\sum_{i=1}^{k} tf_i; \;\; \mu_{t_2} = \frac{1}{k}\sum_{i=1}^{k} tf'_i$$

Pearson correlation is a bit more sensitive as a similarity measure than the cosine similarity. It ranges from -1 to 1, with the value of 1 meaning that the two frequency vectors behave in perfect unison, the value of -1 meaning that the two frequency vectors are opposites of each other (when one keyword rises in frequency, the other falls, and vice versa), and the value of 0 representing the complete *lack of relationship* between the behavior of the first and the second keyword in the news stories.

Your goal is to find any interesting relationships between keywords and news story topics by tracking their occurrence over time. Both high *positive correlations* (e.g., on most days with a lot of stories about "Trump", there are also a lot of stories about "Cruz"), and high *negative correlations* (e.g., on days with a lot of stories about "Apple" the news about "Google" is scarce, and the other way around) are of interest. Additionally, some non-correlations may also be of interest (e.g., news stories about "Kardashian" and news stories about "Pitt" do not correlate to each other).

Question 4: How do specific topics progress through the news cycles over the period of study? The first three questions belong to the category of unsupervised data analysis: we set of to discover insight in exsiting data without any specific guidance of what may be of actual interest to us.

Question 4 introduces supervised data analysis. Here, we are interested in very specfic topics. For the purpose of this lab assignment a *topic* is defined as a single theme around which multiple news stories from the same or different days may appear. A topic may be represented by one or more keywords, and different topics may share certain keywords.

For example, the topic "Republican presidential primary" can be (at least partially) described by the following set of keywords (consider capitalization irrelevant and used only for convenience):

```
Republican, Republicans, GOP, Presidential, primaries, caucuses,
primary, caucus, Ted, Cruz, Donald, Trump, John, Kasich, delegate,
delegates, convention, Cleveland, candidate
```

The topic "Democratic presidential primary" can be described by the following set of keywords:

```
Democratic, Democrats, Dems, Presidential, primaries, caucuses, primary,
caucus, Hilary, Clinton, Bernie, Sanders, delegate, delegates, convention,
candidate
```

As you can see, some of the terms are shared between these two topics.

To analyze each topic you need to:

1. Determine the keywords relevant for it.

2. Build a model for this topic. In your case the model is a bag-of-words representation of the keywords you selected. You may use term frequencies greater than 1 to highlight higher importance of certain terms to your topics.

3. Compare your topic models[3] to the representations of different news stories/news cycles to determine how many stories (and which ones and when) address the topics you constructed.

I want to ask you to study ten topics. I am providing descriptions of some of the topics for each of the aggregator sites. You will choose **five topics** from the list provided below. Additionally, using your common sense and curiosity, you will come up with **five more topics**. For each topic selected, you will perform the analysis described above.

The instructor-selected topics are:

1. Democratic presidential primary (memeorandum)
2. Republican presidential primary (memeorandum)
3. Situation in the Middle East, including Israel and Syria (memeorandum)
4. Any political events in the state of Califronia (memeorandum)
5. What is Google up to? (techmeme)
6. What is Apple up to? (techmeme)
7. Sale of Yahoo (techmeme)
8. Failing Bay Area startups (techmeme)
9. Celebrities getting married (wesmirch)
10. Celebrities getting divorced (wesmirch)
11. Celebrity gossip related to pop musicians (wesmirch)
12. The Kardashian family (wesmirch)
13. New services launched by media companies (mediagazer)
14. Any news related to Buzzfeed (mediagazer)[4]
15. Corporate acquisitions of media companies (mediagazer)
16. Ethics and corruption in journalism (mediagazer)

[3]The term ”topic model” means something slightly different in machine learning, however, since we are not going to cover topic modeling via latent probabilistic methods in this course, I feel justified borrowing it for our purposes.

[4]That is: news stories about something going on at Buzzfeed-the organization, **not** stories published by Buzzfeed itself.

Question 5: Which news sites appear to be influential? Use your data model for the news sites to determine their respective influence, and produce a ranked list of news sites/organizations based on that influence.

For this particular question, I am going to let you develop your own measures. They do not have to be complex, but they do have to reflect the following common sense observations:

- All other things being equal, sites whose articles are featured more often as leading articles are more influential than sites whose articles are features less often.
- All other things being equal, sites that have fewer stories discovered by the aggregator portal are less influential than sites that have more articles discovered.
- (optional) All other things being equal, sites that cover more diverse topics are more influential than sites that cover less diverse/fewer topics.

Devise your measure(s), measure each organization according to the measures you devised, and produce a ranking of the sites/news organizations based on your measure/measures.

Visualization and Report

For each of the analytical questions, you shall visualize the results using appropriately selected visualization techniques.

The analytical questions do not require highly involved visualizations. In fact, most of the data can be visualized by exporting data into a CSV file, uploading the file to a spreadsheet software, and constructing appropriate graphs and charts to porperly represent the results of the analysis. Some data can be visualized by simply outputting it in a nicely formatted way (for example, a ranked list of organizations by their measured influence can simply be reported as a table).

Note: In general, if you know how to use `matplotlib`, you can use it for this assignment. **It is not a requirement**, though, so if you have not used it before, there is no need to use it now - we will discuss `matplotlib` in detail later in the class.

You will incorporate your visualizations into the report you will write about your work on the assignment.

The report shall contain the following information:

- Proper title, list of authors with affiliations (Cal Poly) and email addresses.
- Short introduction discussing the nature of the tasks performed.
- Implementation overview: a quick walk-through the code base you built to complete this assignment.
- Analysis. For each question, present the description of what you did, what decisions (not specified in this handout) you made. Show the obtained results, and discuss them.
- Conclusion. Include an short introspection about the assignment from each team member (please identify each team member explicitly). In your introspections, concentrate on (a) the specific tasks you performed on this project, (b) the parts of the project that were simple, (c) the parts that were interesting, (d) the parts that were hard, (e) the parts that were boring, (f) anything that was unexpected. Summarize your personal experience.

Submission

For this lab we will use CSL `handin` tool.

You shall submit the following.

- All the code you wrote to fulfill the requirements of this lab.
- A `README` file with instructions on how to run your code to perform different tasks of the assignment.
- Your report in PDF format.

Place **all** files and directories *except* for the report file, into a single archive named `lab05.tar.gz` or `lab05.zip` and submit it. Submit the PDF version of the report separately, outside of the archive.

Use `handin` to submit as follows:

```
$ handin dekhtyar lab04 <FILES>
```

Good Luck!