Week 8 in CS 234

What is due on Tuesday, 10/24/17?

1. **Submit the tasks** that were described in the Week 7 document:
    a. Create a page as public_html/cs234/wikipedia/blog.html for the Wikipedia project and blog about the tasks you accomplish. Here is a nice example from Meha. The details are in Week 7.
    b. Submit the Notebook Time, Datetime, Dateutil, (put IPNYNB file in Week7, write takeways in your blog page about the Wikipedia project, link to HTML from your blog).
    c. Submit the Notebook Wikipedia Revisions (put IPNYNB file in Week7, write takeways in your blog page about the Wikipedia project, link to HTML from your blog).
    d. Write a summary about one research paper that is very relevant to your research question. Put that on your blog, but also fill out this form, so that we can have all the information in one place and share it with everyone.
    e. Submit the notebook that your team created to analyze one of your pages for your project topic. **If you followed Eni's request for doing a "dirty draft", please email that notebook to Eni**. If not, simply upload the IPYNB in your Week 7 folder; blog about its findings, and link the HTML version from your blog.

2. Complete the notebook Statistical Significance that we started in class and upload it in the Week 7 folder. See Friday 10/20/17 for link to empty notebook.

3. Read the article "Science isn't broken: It's just a lot hell harder than we give it credit for." [Reading time: 20 minutes]. In your Wikipedia blog write a paragraph about how some of the issues discussed in the article might affect the Wikipedia project you're working on.

What is due on Friday, 10/27/17?

1. Each student should try complete the new Notebook Time Series with pandas. In this notebook you'll learn how to manipulate timestamps in a way much more efficient and fast that we could do with pure Python code. Some of the tasks that took you several lines of code to do in Week 7, can be done with a function call with pandas timeseries. It is important that everyone learns this material. Timestamps will be an element of all our datasets in this course. Once you have completed the notebook, don't forget to blog about what you learned, link from the blog page the HTML version of the notebook, and store the IPYNB in the Week 8 subfolder in "dav".

2. Start creating a new exploration notebook to study the diffsize of edits and the contributions of editors over time (you can have your own personal notebook or work closely with the team. If you work independently, the team can explore several pages in parallel). See explanations from my email at the end of this document. Similarly to the time series of number of edits over time in the notebook from Point 1 above, you can experiment with creating a time series of diffsize of page changes over time. If you do this for a few pages related to your project, what kind of trends do

you see? Is it possible to detect events that are being reflected in the size of edits? What kind of events? Vandalism? Breaking news?

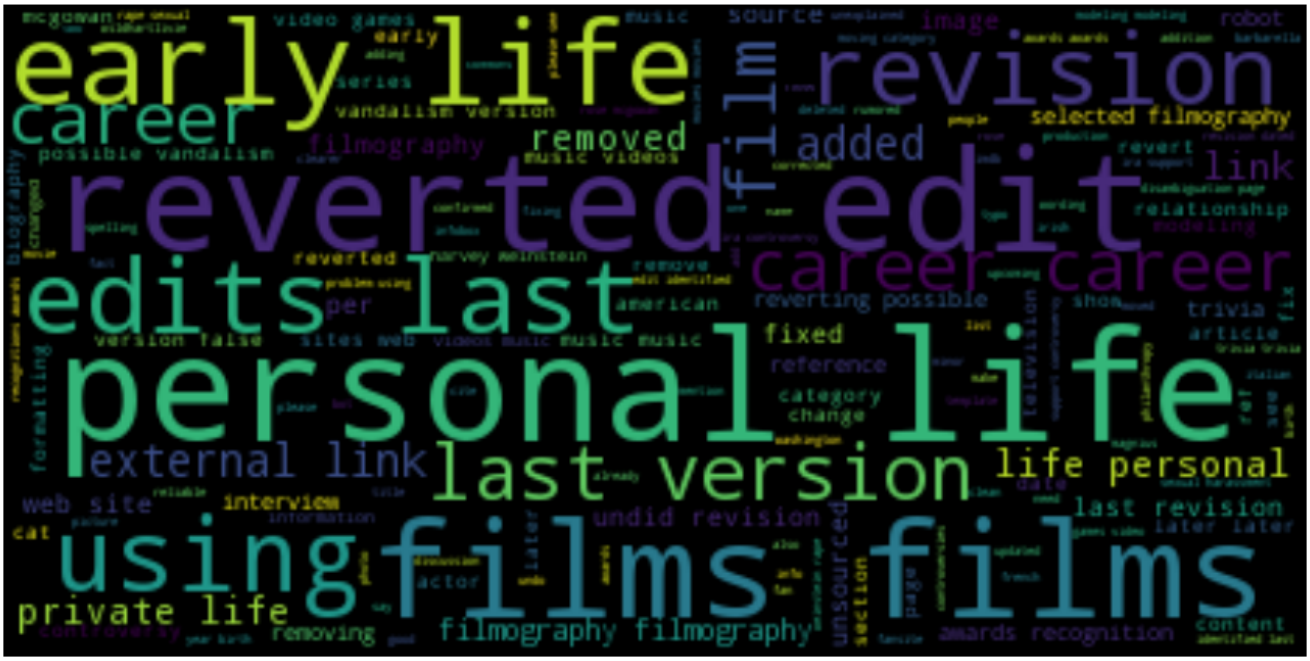
3. Short discussion about the state of your Wikipedia project. We have been keeping track of the evolution of our Wikipedia project over the past days:
    a. Wikipedia research questions (what each group wants to study)
    b. Wikipedia first results (a particular page that a group looked like)

    Working over the week on the two notebooks with time series, you'll gain new understanding about your desired topic and focus your question even more. Try to make a commitment for what you want to focus on the last week of the project (given that Nov 3 is the deadline for this project). Use this Google doc to write the current state of your project and what you want to achieve in the last week.

Challenge yourself

Word Clouds

The revision snippets contain a comment field that often has explanations about the revisions being made. It's possible to create word clouds of these comments to get a quick sense of what the revisions are about, using the Python package word_cloud. In addition to generating the cloud as an image, you can also get the frequency list of most common words and bigrams, thought sometimes these are misleading (the bigrams). Try it to learn something new that it's a common visualization technique for text. Let me know if you have questions. Here is one that I created for the revisions in the page of Rose McGowan. It's not perfect, but I didn't have to do any tweaks (which we might need to do, but not in this project).



Plotly Visualizations
You have seen that Dash uses plotly graphs together with pandas dataframes. Now that you'll learn to build time series with pandas, you can think about how to create a simple plotly graph with a dash page. Choose one of the visualizations that captures well the findings of your research question and think about how to implement it.

Study the size of edits in revisions [this was sent as email]

Our mwclient library doesn't seem to have documentation for how to compare revisions. We can use the Media Wiki API to access revisions directly. The URL will look like this:

https://en.wikipedia.org/w/api.php?action=compare&fromrev=805841725&torev=805850060&format=json&prop=size|diffsize|user

The fields `fromrev` and `torev` are unique for each revision page and we can read them from the revision list that we created with the help of the `mwclient` module:

```
{u'comment': u'',
 u'parentid': 805841725,
 u'revid': 805850060,
 u'timestamp': time.struct_time(tm_year=2017, tm_mon=10, tm_mday=18, tm_hour=1,
tm_min=13, tm_sec=4, tm_wday=2, tm_yday=291, tm_isdst=-1),
 u'user': u'User No. 99'}
```

To generate the URL above with Python, you'll write:

```
import requests, json

urlbase = "https://en.wikipedia.org/w/api.php"
rev = revisions[0] # you'll need a for loop to get many revisions

req = requests.get(url=urlbase,
                   params = { 'action': 'compare',
                              'format': 'json',
                              'prop': 'size|diffsize|user',
                              'fromrev': rev['parentid'],
                              'torev': rev['revid']})

response = json.loads(req.content)
print response['compare']['diffsize']
```

It is possible to get the text that was modified, but it's in HTML and it's a bit ugly to parse.

Have fun working on the project!