Project 1: Tweet Mining & The Golden Globes

**Presentations:** There will be short presentations of the projects to the TA.

**Project Deliverables:**

1. All code must be in Python 3. You can use any Python package or NLP toolkit, but please save and share your requirements as follows: Create an environment for the project, run "pip freeze > requirements.txt", make sure it works after running "python install -r requirements.txt" in an empty environment, and then include this "requirements.txt" in your submission/repository.
2. You must use a publicly accessible repository such as Github, and commit code regularly. When pair programming, note in the commit message those who were present and involved. We use these logs to make sure all team members have been engaged in case problems are brought to our attention, and to avoid penalizing the entire group for one student’s violation of academic integrity. We don’t look at the commits unless there’s something really wrong with the code, or there’s a complaint.
3. Please use the Python standard for imports described here: [https://www.python.org/dev/peps/pep-0008/#imports (Links to an external site.)](https://www.python.org/dev/peps/pep-0008/#imports)
4. Bundle all your code together, your submission will be a .zip file on canvas.
5. If you use a DB, it must be Mongo DB, and you must provide the code you used to populate your database.
6. Your code must be runnable by the TA: Include a readme.txt file with instructions on what file(s) to run, what packages to download / where to find them, how to install them, etc and any other necessary information. The readme should also include the address for your Github repository.
7. Your code must run in a reasonable amount of time. Your grade will likely be impacted if this is greater than 10 minutes.
8. Your code cannot rely on a single Twitter user for correct answers. Particularly, the official Golden Globes account.

**Minimum Requirements:**

Fulfilling only the minimum requirements puts your group on track for a B.

A project must do a reasonable job identifying each of these componentes:

1. Host(s) (for the entire ceremony)
2. Award Names (Categories)
3. Presenters, mapped to awards\*
4. Nominees, mapped to awards\*
5. Winners, mapped to awards\*

\*These will default to using a hardcoded list of the awards to avoid penalizing you for cascading error. Please note that, when mining award names specifically, you cannot hardcode parts of these names in your solution with the only exception of the word "Best."

It is OK not to have 100% accuracy. It's very rare for any group not to have some error, especially with awards and nominees. Even getting just half of the nominees for a given award is quite good performance.

**Additional Goals:**

To get better than a B, you must do exceptionally well on the minimum requirements, or complete one or more additional goals. Some examples of additional goals:

* Red carpet – For example, determine who was best dressed, worst dressed, most discussed, most controversial, or perhaps find pictures of the best and worst dressed, etc.
* Humor – For example, what were the best jokes of the night, and who told them?
* Parties – For example, what parties were people talking about the most? Were people saying good things, or bad things?
* Sentiment – What were the most common sentiments used with respect to the winners, hosts, presenters, performances, and/or nominees?
* Performances – What were the performances, who were the performers, when did they happen, and/or what did people have to say about them?
* Your choice – If you have a cool idea, suggest it to the TA! Ideas that will require the application of NLP and semantic information are more likely to be approved.

Typical performance on the minimum requirements, plus a well-done additional goal, is likely to earn an A- or better.

**Required Output Format:**

You are required to output your results in two different formats.

1. A human-readable format. This is where your additional goals output happens. For example:  
   Host: Seth Meyers  
     
   Award: Best Motion Picture - Drama  
   Presenters: Barbara Streisand  
   Nominees: “Three Billboards Outside Ebbing, Missouri”, "Call Me by Your Name", "Dunkirk", "The Post", "The Shape of Water"  
   Winner: “Three Billboards Outside Ebbing, Missouri”  
     
   Best Dressed: Jane Doe  
   Worst Dressed: John Doe  
   Most Controversially Dressed: John Smith
2. A JSON format compatible with the autograder; this is only containing the information for the minimum tasks. For example:

{

"Host" : "Seth Meyers",

"Best Motion Picture - Drama" : {

"Presenters" : ["Barbra Streisand"],

"Nominees" : ["Three Billboards Outside Ebbing, Missouri", "Call Me by Your Name", "Dunkirk", "The Post", "The Shape of Water"],

"Winner" : "Three Billboards Outside Ebbing, Missouri"

},

"Best Motion Picture - Musical or Comedy" : {

"Presenters" : ["Alicia Vikander", "Michael Keaton"],

"Nominees" : ["Lady Bird", "The Disaster Artist", "Get Out", "The Greatest Showman", "I, Tonya"],

"Winner" : "Lady Bird"

},

**The Data:**

Uploaded to canvas: Files > Project 1 > gg2013.json.zip

Uploaded to canvas: Files > Project 1 > gg2013answers.json

The data is formatted as a list of JSON objects, as seen below:

[{u'id': 554402424728072192, u'text': u'just had to scramble to find a golden globes stream for my brother. :D', u'user': {u'id': 19904553, u'screen\_name': u'baumbaTz'}, u'timestamp\_ms': u'1421014813011'}, {"text": "What?!? https://t.co/NSPtGtbCvO", "id\_str": "950142397194821632"}, ...]

You will be graded on at least one year you have not seen.

**The Autograder:**

The autograder is your way of benchmarking your progress as you work on improving accuracy.

* The central repository is at <https://github.com/c-col/gg-project>, and it contains:
  + A copy of the autograder program, which will assess how well you did on the basic tasks. It has undergone some changes as the project format has changed, so please report bugs early and often so that the TA can get it fixed ASAP.
  + A template for the API the autograder uses, saved as gg\_api.py. Be sure to read the doc strings and ask the TA if you have any questions about how to use this file.
  + A JSON file with the correct answers for the minimum task for 2013; these answers are used by the autograder. DO NOT read this into memory in your own code. **Doing so is grounds for an automatic zero.**

The autograder is somewhat strict. Use it to guide your development. When we actually grade your project, we’ll look “near misses” as well as how well you’ve considered generality in architecting your system.

**Grading:**

1. If you do a reasonable job on the minimum requirements, and no more, you’ll get a B.
2. If you do exceptionally well on the minimum requirements, or you do a reasonable job on the minimum requirements and one or more additional goals, you’ll get an A- or better.
3. Your code will be run and graded on at least one year for which you have not been provided data. This is to ensure you don’t overfit to the provided data.
4. If you want additional feedback after receiving your grade, please email the TA and we can let you know how your group compared to others.

**Frequently Asked Questions**

**I noticed that there are these git repos of past years’ projects hanging around. Can I look at them? Can I use their ideas? Can I use their code?**

I would encourage your group members to brainstorm approaches to the project before looking at any solutions from past years. You may look at what other groups have done, but you should cite which repositories you’ve looked at in a file you keep in your own repository. You may use ideas you get from looking at these, but again, you must cite them. Also, please keep in mind that not all groups did equally well. Blindly adopting methods used by groups from previous years could serve you very poorly. You may not use their code.

**I’m not a very experienced coder, and my teammates are excluding me/There’s this person on our team who isn’t contributing. What should I do?**

It’s counter to the goals of the course to exclude a teammate for any reason, and certainly not because of their inexperience. Older, more experienced students have an obligation to provide *some* mentorship. However, extensive gaps in knowledge should be referred to the TAs. Treating more experienced students as personal tutors or technical support is also not right.

There are a number of ways that less-experienced coders can be included in the work such that they share the burden and also learn from the experience. Here are a few suggestions:

1. Plan some pair programming sessions with the less-experienced coder. They may not be able to provide many suggestions, but what they learn they can apply to other work they do for the project.
2. Choose a function that is suited to the student’s coding capabilities but not central to the project (for example, a bonus feature), and have them focus on that function. When they run into trouble, try to help them out, but again, extensive difficulties can be referred to the TAs.
3. A lot of the conceptual work, data exploration, and quality analysis can be done by someone with little or no coding experience. Inexperienced coders should be given the opportunity to code, but they can also do a larger share of these tasks.

**I’m having trouble contacting one of my team members. What should I do?**

If they are ignoring your emails, please talk to the TA.

**One of my teammates isn’t doing/didn’t do any work. What can I do about it?**

If you’ve made every effort to reach out and they still aren’t contributing, please email the TA and Larry outlining what you have done to try to include your teammate, and in what ways you feel your group is impacted. Then, do your best without the missing teammate. Your commit logs will support your claims, and we will take the reduced team size into consideration when we grade your work.