Brown University DATA 2040 Spring 2018

Building a Search Engine: Part 3a

Due: ~~March 23, 2018, 9pm April 2, 2018, 9 PM~~ Monday, April 9, 2018, 9 PM

Overview

You and your Part 2 partner will implement PageRank scoring on top of your Part 2 code. In addition, you will make some changes to your code in preparation for the rest of the project.

Motivation

The three main skills we want you to gain from this assignment are listed below:

- Algorithm Design: Implementing an algorithm that you only know the name/formula for.
- Practical Techniques: Optimizing Python code with numpy.
- Software Engineering: Integrating complex code into an existing codebase.

See the Staff Note at the end of the handout for more information on why we decided to focus on the above three topics. Note that while this handout is fairly long, the amount of code you will likely write is short.[1]

1 Important Changes-- READ THOROUGHLY

1.1 Index Construction

Your program should output your indices into a directory instead of to 2 files.[2] This means that to index a collection, a user should enter 3 arguments instead of 4, as shown below:

New: `python3 create.py stopwords.dat collection.dat index/`
Old: ~~`python3 create.py stopwords.dat collection.dat postings.dat titles.dat`~~

Also update `query.py` so that it can be run as follows:

New: `python3 query.py stopwords.dat index/`
Old: ~~`python3 query.py stopwords.dat postings.dat titles.dat`~~

[1] Your PageRank implementation will probably be roughly 30 lines of code, as will the integration.
[2] If the `index/` directory does not already exist, it should made by `create.py`. If it already exists, simply overwrite the existing files inside. To create a directory in Python, you can use the mkdir function in the os module.

Your `index/` directory *might* look like this:

```
index/
  postings.dat
  titles.dat
  pagerank_scores.dat
  …
```

If a user attempts to enter the incorrect number of command line arguments, they should receive an error message which informs them about this change.

```
python3 query.py stopwords.dat postings.dat titles.dat
Error: The option to specify index files was removed in version 3a,
please input an index directory instead.
```

We recommend that you do this part *before* integrating PageRank into your implementation (described in section 2.1.3).

What we look for: Your code passes just as many Part 2 tests *with the new argument format*. Your verbose flag still works and your code prints out the error message specified above.

2 New Features

2.1 PageRank Scoring

2.1.1 Implementing PageRank

This is an open-ended task that starts with researching PageRank and ends with making sure that your implementation matches the specification. You will translate a high-level, mathematical description of the formula into a working Python implementation.

We will provide you with two files `link.py` and `link_test.py` to get started. link.py contains code to parse the links from the Wikia collections. **You should implement PageRank in the `rank()` function in `link.py`.** [3]

You are encouraged to incorporate the support code in these files into your search engine.

We use the algorithm from Wikipedia shown below. We will be grading you based on how *similar* your results are to ours, to within reasonable bounds.

[3] We use the built-in `typing` library from Python 3 for type hints in function definitions.

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

Where $p_1, p_2, ..., p_N$ are the pages under consideration, $M(p_i)$ is the set of pages that link to p_i, $L(p_j)$ is the number of outbound links on page p_j, and N is the total number of pages. d is the damping factor.[4]

We use the iterative method with a damping factor, *d,* of 0.85 and an epsilon, ε, of 0.01. These terms are defined in the Wikipedia article.

What we look for: When you run

```
python3 link.py PATH/TO/PIXAR_COLLECTION
```

Your top 3 results will depend on how you handle duplicate links.

Recommendations

- Skim the support code and test cases before you start implementing.
- Read your error messages (What does `ZeroDivisionError` tell you?)
- Split the problem into reasonable chunks. (Ex. initialize the arrays, then …)
- Add unit tests for `rank()`, you'll need them in section 2.1.3
- Think about how you solve problems in general.

2.1.2 Vectorizing PageRank

Vectorization is a technique that can be used to speed up computation-intensive algorithms like PageRank. In this section, you will optimize the code that you produced in 2.1.1.

If you have not used numpy before (or want a refresher), we recommend going through this interactive tutorial.

The best way to learn is through practice. See this StackExchange post for some other people's opinions on how to learn vectorization (in addition to 3 more advanced vectorization exercises). We've listed a few more basic examples below.

Examples

Let x and y be two numpy arrays of length 10. Each row contains equivalent code.

Plain Python	Vectorized Python
# for-loop to numpy method	result = np.sum(x[i])

[4] A damping factor of around 0.85 is generally assumed. Read more

result = 0
for i in range(10):
    result += x[i]	

# vector math 	
result = [el / 7 for el in x]
	Result = x / 7

# expanding helps 	
result= 0
for i in range(10):
    result += (x[i] + y[i]) ** 2
	result = np.sum(x * x)
	result += 2 * np.sum(x * y)
	result +=np.sum(y * y)

# filtering with masks 	
result = 0
for i in range(10):
    if (x[i] > 5):
        result += x[i]
	result = 0
	indices = np.argwhere(x > 5)
	result = np.sum(x[indices])

We want you to optimize so that one run of `link.py` on the Pixar dataset takes less than 10 seconds to complete.

What we look for: The code to run in your terminal should look like the following:

```
time python3 link.py PATH/TO/PIXAR_COLLECTION
```

Recommendations

- Work from the inner loop out.
- Know your core numpy functions (`sum`, `mean`, etc.)
- Keep changes incremental.
- **Don't forget to actually time your code before submitting**
    - Use the time function as shown above

2.1.3 Integrating PageRank

Your search engine should be able to rank query results using PageRank. This means you will need to incorporate your PageRank implementation into your search engine code. Below is a summary of what you should do.

`create.py` should **parse**, **compute** and then **index** the PageRank scores.

`query.py` should do the following for each query.

1. Find documents which match the query. (Completed in Part 1)
2. Sort the documents by decreasing PageRank score.

PageRank should **only** be used to rank queries if a user types in the following:

```
python3 query.py --rank=pagerank …
```

TF-IDF (from Part 2) should be used to rank queries if a user types in **either** of the following:

```
python3 query.py --rank=tfidf stopwords.dat …
python3 query.py stopwords.dat …
```

What we look for I: The 3 commands above should run properly on **all Wikia** datasets. Here are the expected results to some queries in the Pixar dataset.

```
    python3 query.py --rank=pagerank PATH/TO/STOPWORDS
PATH/TO/INDEX
    nemo
    2979 1829 3843  ...
    nemo AND dory
    2979 3843 3017 ...
    "nemo dory"
    1833 25474 2159 ...
```

What we look for II: We will look for new unit tests.

Recommendations

- You likely will want to *copy* some of the code provided in `link.py` into your own functions.
    - Don't call functions from `link.py`, rather copy over what you need
        - Note that you shouldn't need to change any of the functions in `link.py`
    - Make sure the `rank()` you use in create.py is the same as link.py
        - This makes grading easier for us because we'll be able to determine where ranking errors, if any, are coming from.
- Get a good idea of what functions in your code you will need to change.
- Keep changes incremental; make commits for even just 10-20 lines of code changed!
    - But make sure any commit you make is of code that compiles!
- Don't be scared to revert a couple commits if something goes wrong.
- **Don't forget to add tests for the your new code. This includes PageRank!**
- **Don't forget to fix any tests that break because of the changes you made.**

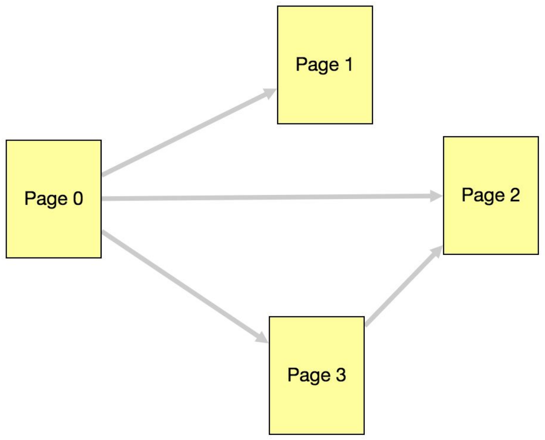
3 Report

Please include a report in PDF format answering the questions below.

1. How did your team keep on the same page? What problems did you face and what would you change in the future?
2. What sections of your Part 2 code did you need to change in order to add the PageRank scorer?

3. Let PR_i be the PageRank of the i-th page in the diagram below. Using the equation below, define PR_1 and PR_2. Note: The only variables in your equation should be PageRank scores, PR_i, and the damping factor, *d*.

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$



4. Include any known bugs or implementation details you'd like us to know about in your report.

4 Download

We have provided you with support code to help you parse the links in the Wikia collections here.

5 Submission

Push your changes to the `master` branch of your team repository (from Part 2) before the deadline.

You should include `link.py` as we will use that to grade Section 2.1.2.

6 Evaluation

We will be running your programs on **only** the Wikia collections. This includes the Pixar and Breaking Bad collections. Like in the previous parts of the project, we will grade your functionality by properly invoking `create.py` and `query.py` and comparing your results to ours.

As noted in previous sections, we will grade your results by invoking the commands in the following form:

For 2.1.1: `python3 link.py PATH/TO/PIXAR_COLLECTION`
For 2.1.2: `time python3 link.py PATH/TO/PIXAR_COLLECTION`
For 2.1.3: `python3 query.py --rank=pagerank PATH/TO/STOPWORDS PATH/TO/INDEX_DIR`

We will hand-grade your report for correctness and understanding. We will also hand-check your test cases.

Staff Note

As students ourselves, we understand that school is busy and stressful. Deadlines and unclear instructions make material that *would* be fun exhausting instead. In this note, we hope to help you understand *why* we designed the assignment in a certain way so that the challenging parts of the project are flow-inducing instead of anxiety-inducing.

What should you do if you find a bug or an unspecified detail?

Send us a private Piazza post with a description of the issue at hand and an image of the code/paragraph. If you are the first person to notify us of the problem (with a good enough description), you will receive a couple extra credit points.

Why do we give vague instructions?

We understand that it is important to learn technical knowledge well. However, we also believe that *designing* solutions to computer science problems is a much more generalizable skill that will always be relevant to you as a data scientist. By trying to encourage you to come up with your own approaches, we sometimes end up leaving out information that seems important to your grade.

We try our best to make the specification clear enough so that anxiety about grades doesn't disrupt learning. However, we make mistakes (all the time) and it's hard to catch them all. **We do want to make it clear that we try very hard to make sure that unspecified details do not affect your grade.**

Why aren't we giving you the pseudocode to PageRank?

PageRank is just one of many algorithms that you *might* use in the future. Also, we believe that you all can learn PageRank on your own. Therefore, it is much more important that you gain your own style of learning and implementing algorithms. Data science is a field where new variations of algorithms pop up all of the time.

Some related articles to this assignment

- https://hortonworks.com/blog/using-pagerank-detect-anomalies-fraud-healthcare/
- http://treycausey.com/software_dev_skills.html

- https://blog.statsbot.co/data-structures-related-to-machine-learning-algorithms-5edf77c8bbf4