Quiz 3: Traversing Graphs[1]

1. TrafficRank [50 points]

The random surfer model of calculating the PageRank of a web page works by assuming a web surfer follows the outgoing links of the page with equal probability, but will restart every time with probability 1−β, and start at a new page, chosen uniformly at random. The probability β is called the decaying factor and influences the convergence of the computation of PageRank; typical values of β are about 0.85.

[1] Source: Archive of Our Own

The idea of PageRank is not limited to web pages. In this problem, we model the economic rank of areas within a city under the hypothesis that inbound taxi traffic into an area might be an indicator of its economic rank. Thus we invent the notion of TrafficRank.

The main differences between *PageRank* and *TrafficRank* are that (1) for PageRank the probabilities of outgoing links are equal but for TrafficRank they are weighted by actual usage, and (2) links within an area are also counted. The resulting transition matrix is typically a full matrix, not sparse as in the case of PageRank.

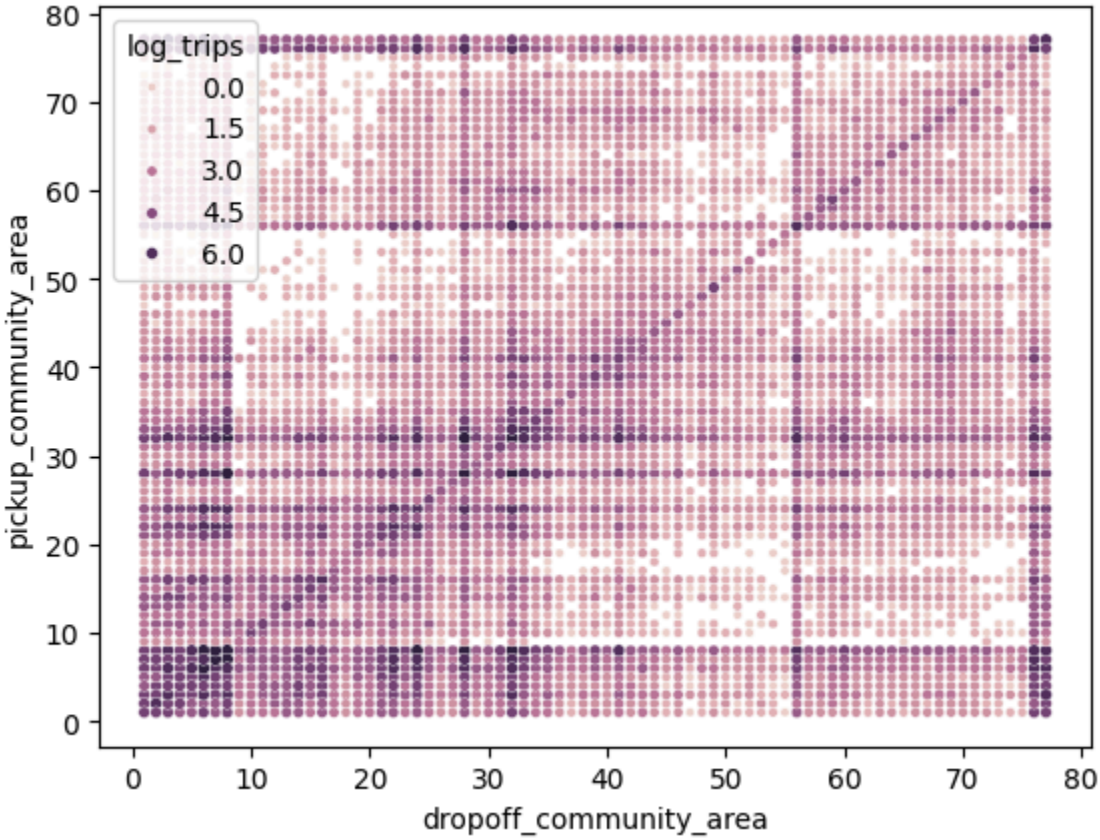
Viewing Chicago Taxi Rides [5 points].

Downloadable from my Github repository using[2]

```
wget https://raw.githubusercontent.com/singhj/big-data-repo/main/datasets/chicago-taxi-rides.csv
```

A scatter plot of the data allows us to see the "busy" spots as well as the "light" spots in the city. CS-119 is not a data visualization course even though it helps to have a view of the data.

[5 points for constructing a similar scatter plot]



[2] Pardon the microscopic font. Hopefully you're not trying to read it, just copy and paste it.

The city is divided into 77 "community areas," thus giving us a 77 × 77 matrix of T_{ij} entries where T_{ij} is the number of trips from community area i to community area j.

1. [15 points] The given data has rows (i, j, T_{ij}), sometimes known as Coordinate format. It is supported by many libraries, `SciPy.sparse.coo_matrix` among them. Since the matrix is just 77 × 77, a "big data" technique is not required here. Read the data as a matrix.[3]
2. [15 points] Using your formulation of the TrafficRank algorithm, calculate the rankings of the Chicago community areas after 0, 1, 2, 3, 4, 5 and 6 iterations of the algorithm.
3. [15 points] An alternate measure of economic rank of an area might be the inverse of a "hardship index," defined and quantified here. How well, or poorly, do your calculations of TrafficRank for Chicago community areas correspond with the inverse of hardship index? Give *a qualitative analysis in plain English*.

You may make any other reasonable assumptions you deem necessary.

2. Text Processing [50 points]

All US presidential speeches may be downloaded as a single zip by using[4]

```
wget https://raw.githubusercontent.com/singhj/big-data-repo/main/datasets/prez_speeches.zip
```

Which are the most important words in a set of documents? Please read MMDS Book section 1.3.1: Importance of Words in Documents

TF-IDF, short for *term frequency–inverse document frequency*, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. Its usefulness for this purpose is well-established.

Numerous videos available on the internet explain the statistic and how it is computed, for example this one.

About TF-IDF [10 points]

Learn about TF-IDF and answer the following:

1. [4 points] We can calculate TF values of each word in a given document. Explain why the calculation of IDF can only apply to a corpus, not to a specific document.

Scikit-Learn

- $\text{IDF}(t) = \log\frac{1+n}{1+\text{df}(t)} + 1$

Standard notation

- $\text{IDF}(t) = \log\frac{n}{\text{df}(t)}$

[3] You may use any library, e.g., *scipy* or *numpy*, for reading the matrix. A Jupyter Notebook is quite convenient but any Python platform is OK.

[4] This zip file is in a "double-compressed" format: a zip file containing <name>.tar.gz files.

2. [6 points] The implementation of Scikit-Learn's IDF calculation differs from that of the "standard" calculation. What is the justification for this change?

3. [25 points] Each president has a .tar.gz file containing his speeches. Write a procedure to calculate TF.IDF for any president's speeches and print the top-15 most important words in their speech.

4. [15 points] Examine the result carefully – at least ***some*** of the top TF.IDF words should be historically consistent with what was going on in the country at the time[5]. You only have a slight control over the outcome through starting with an initial set of stopwords and adding more words to the list of stopwords.

```
import requests
stopwords_list =
requests.get("https://gist.githubusercontent.com/rg089/35e00abf8941d72d419224cfd5b5925d
/raw/12d899b70156fd0041fa9778d657330b024b959c/stopwords.txt").content
stopwords = set(stopwords_list.decode().splitlines())
stopwords = list(stopwords)


def remove_stopwords(words):
    list_ = re.sub(r"[^a-zA-Z0-9]", " ", words.lower()).split()
    return [itm for itm in list_ if itm not in stopwords]
```

[5] Your code will generally always give you an answer; the challenge is to determine if the answer "makes sense."