# Users' Interest Shift: A Reddit Data Analysis

– By Wendy Li, Hao-Hsiang Song, and Hao Mou

**treeNode.py:**

• Contains the class for the TreeNode objects, which are used to parse the HTML files.

• Contains the class for the edges, which store the data for the interactions between users (data from parsing the HTML)

**processor.py:**

• Main driver for parsing HTML files.

• Because of computational restrictions, we had to break the parsing into chunks and store the parsed data as partition files.

**partitioner.py:**

• Functions for splitting the partitions into four month long snapshots.

**parser.py**

• Contains functions that actually parse the HTML files to extract the relevant attributes and information.

**constants.py:**

• Contains the names of the partitions (four month snapshots) of all parsed data.

**compare.py:**

• Contains common_user_percentage, a function that finds the percentage of common users between the clusters in two snapshots.

• Contains compare, a function that prints out the usernames of the common users in two snapshots.

**movement_btw_snapshots.py**:

• Contains a helper function create_newdict that preprocesses the dictionaries of the Louvain's algorithm.

• Contains percentage_of, a function which calculates the fraction of a cluster in snapshot A that moves to a given cluster in snapshot B. Also calculates the percentage of a given cluster in snapshot B that comes from a given cluster in snapshot A. For example, 5 percent of cluster 1 in snapshot A goes to cluster 8 in snapshot B.

• Contains a print function that prints these statistics.

**graph_stats.py:**

• Contains functions that extract features like the average clustering coefficient, the degree assortativity, the degree assortativity for top categories, and the number of users for each snapshot.

• Contains functions that graph these statistics all on one graph.

**try_print.py:**

• Contains a toXGraph function that converts the snapshots (partition files) into an actual Networkx graph. toXGraph runs Louvain's algorithm and returns the clusters of the given snapshot.

• Contains a numCluster function that given the clusters of given snapshot, counts the number of clusters in the graph

• Contains a getCategory function that given the clusters of a given snapshot, prints the unique categories in the snapshot and the number of occurrences of that category. Also prints out the total number of users.

• Contains a getAggregateCats function that finds the top categories for each snapshot.

• Contains a getAvgCat function that finds statistics for number of different categories for each cluster in a snapshot. It finds the mean, standard deviation, minimum, and maximum.

To run, "python3 try_print.py -f [partition filename]". The partition filename is the snapshot we are running the program on, and we assume that file is located in a folder called "output".

**track.py:** • Contains a tracker function that, taking a rtf file with the cluster movement between two snapshots percentages, converts these statistics into bipartite graph, visualizing the old cluster moving to the new cluster.

• Contains track_sankey, a function that takes a single rtf file input and produces a sankey graph. The rtf file input has the movement statistics from one snapshot to the next one.

• Contains a track_sankey_consolid function that works like the track_sankey function, but specifies a particular threshold. Cluster movement below that threshold will not be visualized.

• Contains a overall_sankey function that constructs the sankey graph for the first four snapshots, visualizing the movements of the clusters from snapshot to snapshot.

• Contains a track_sankey_flow function that finds the mean, maximum, minimum, and standard deviation of the number of flows leaving each source and the percentage of each source that goes inactive.

To run, "python3 track.py". The command will output the overall sankey graph shown in the paper, which shows the cluster movement across the first four snapshots. There is also an option to run the other functions using commandline arguments (see the code for further documentation).

**sankey.py:**

• Contains a helper function sankey, which track.py invokes to construct the sankey graphs.

• Contains a helper function sanFlow, which track.py invokes to get the information to determine the flow statistics that the track_sankey_flow function calculates.

**print_topcat.py:**

• Contains a print_topcat function that prints the top categories and their respective percentages for each snapshot

• Contains plot_percentage_bar, a function that visualizes the percentage frequency distribution of interactions in different subreddits across snapshots.

To run, type "python3 print_topcat.py".

**rtf_broker.py:**

• Contains functions that help manage the rtf files in constructing the sankey graphs.

**get_usernum.py:**

- Returns total number of users in our Reddit dataset.

Run by calling "python3 get_usernum.py"

**output Folder:**

- Contains all the raw partition data from the parsed HTML files.
- Partition files are our starting point after HTML parsing.

**Stats Folder:**

- Contains the intermediate data used for data analysis.
- For instance, A.rtf represents the raw data for the movement between the first and second snapshots.

**Distributions Folder:**

- Contains all the distribution graphs of user interest continuity.

**Graphs with Inactive Folder:**

- Contains all the sankey graphs.
- Make sure to zoom all the way out to see the entire flow pattern.