HW 4: Link Analysis

1 Pagerank and HITS

First, we construct **a graph of email exchanges** between Enron employees, where each person is represented by a node and each email is an edge connecting two people (we remove emails to oneself). For every person we keep track of a list of origin and destination nodes using 2 distinct dictionaries. Nodes with no outgoing edges are inserted into a SinkNode set. Next we will use Pagerank and HITS to analyze the information flow within the graph.

Pagerank is used to determine the relative importance of a node in a directed graph based on how many other nodes point to it either directly or indirectly. The algorithm uses the Graph's outgoing nodes dictionary and its SinkNode set. First, it sets the Pagerank value (PR) of each node equal to $\frac{1}{N}$ (N = number of nodes). Next, the Pagerank formula is applied to every node for 10 iterations, as specified. On every iteration, each $node_i$ is set to: $PR(node_i) = \frac{1-\lambda+(\lambda\times S)}{N} + \lambda \sum_{j\to i} \frac{PR(node_j)}{out(node_j)}$

The critical steps in the algorithm are ensuring that:

- It uses the PR values of the previous iteration to update all nodes in the graph in the current iteration. Since we implemented the algorithm with dictionaries, not matrices, this required keeping two separate dictionaries to keep track of the current and the next iteration PR values for each node separately.
- The value of S, the sum of the PR for all sink nodes, is updated on every iteration. Sinks will have incoming edges so their PR can change after each iteration. We must update S and distribute the updated value on every iteration to all nodes in the graph

Hubs and Authorities (HITS) also assigns an importance score to each node in a directed graph. The two scores are defined relative to each other. The Hub score is dependent on how good a particular node is at pointing at other authoritative nodes. The Authority score is calculated based on the number of good hubs that point to it. Like with Pagerank, we ran the algorithm for 10 iterations. On each iteration we update each node's Hub and Authority values in this way: $Hub(node_i) = \sum_{i\to j} Auth(node_j)$, and $Auth(node_i) = \sum_{j\to i} Hub(node_j)$

The algorithm uses the Graph's incoming and outgoing node dictionaries. The key parts of the algorithms are those that diverge from Pagerank:

- The algorithm does not treat sinks differently from non-sinks, and it performs no random visits (note there's no λ). Because of this and to ensure convergence, on every iteration we must normalize both the Auth and Hub scores to one by dividing each score by the square root of the sum of squares of all the scores.
- We do not need two different scores for the current and the previous iteration. Instead, we update the Hub score using the Authority score we calculated in the current iteration (or viceversa).

Both Pagerank and HITS output the required results into pr.txt, auth.txt and hubs.txt respectively.

2 Visualization of the Email Flow

To visualize the most interesting data, I leverage the scores from PageRank and HITS as follows:

- I identify the most interesting people by taking the union of the 3 sets of top 100 people ranked by the PageRank, Authority and Hub algorithms respectively. I expect the output of these algorithms together to be able to identify the critical people in the company most effectively. I call this the **KeyEmployees** list. Note that I explicitly decided to exclude pete.davis@enron.com from this group since it is an automatic receipt system that would add little value to my analysis.
- For those in the KeyEmployee list, I **condense the email exchanges from** one person in the list **to** another also in the KeyEmployee list into a single communication block (i.e. every pair of employees might have two blocks associated with them). I store this communication block in **JointEmailsDict** dictionary and concatenate the sender and recipient emails as the key to this dictionary.
- I create a **pairs** dictionary which I index using the same sender-recipient string. I set the value stored to be the pair of people communicating (sender, recipient). This helps me keep track of to which pair a particular communication block belongs to.
- I use the entire subject.txt to **produce an IDF dictionary** ($k = 2$). [1] I exclude words that appear in over 90 percent of subject lines, and those that appear in 1 subject line only. The first condition helps ignore overly common words; the second intends lower the importance of typos (common in email).
- Next, **I include a query that I expect would be relevant to someone investigating the Enron scandal.**[2]
- I tokenize and stem the JointEmailsDict exchanges and the query, removing stop words. Finally, **I use TF.IDF to rank the exchanges between KeyEmployees relative to the search query.**
- I choose the six highest ranked email exchanges (i.e. two employees per exchange), which determine the 12 employees to be included in my graph, and the direction of the edges connecting them. For those employees with incoming edges only, I search for the next best ranked email exchange that involves them emailing someone else who is already in the KeyEmployee list. I add these new edges between relevant nodes.
- For each of the edges, I include a tag cloud of 10 top keywords. I choose these keywords of the edge's associated email exchangeby by ranking the words based on $IDF \times frequency$.

The output of this process is a graph that includes interesting information for anyone investigating the Enron scandal, especially in relation to their failed risk management strategies.

[1]For easy reuse I stored the values of this into idf.txt.

[2]The query used is: 'california bankrupt risk investigation downgrade credit Dynegy junk SEC accounting'. I picked these terms after reading the Enron Scandal article in Wikipedia.