Invisiable Edge
CS 526 Challenge Problem 3, Group 11

Martin, John Urffer, Matthew Wan, Lipeng

December 4, 2012

Todo list

1 Introduction

1.0.1 Data Distribution

The distribution of the data was investigated in order to get a scope of the problem. Each attribute of an edge was binned into a histogram. Figure 1 shows the distribution of the `calls, texts, secs and days` of Moria, while Figure 2 is for Standelf. It can be observed that most of the distributions are skewed to the reflect that a very large number of people use their phones normally, with a long tail reflecting that the number of users who make a large amount of communications falls off quickly. An intresting fact gleaned from this analysis is that the communcation between two people is not normally distributed and symmetric. This was surprising because it was thought that there would be the similar number of people making short calls compared to long calls (or similary for the number of texts).

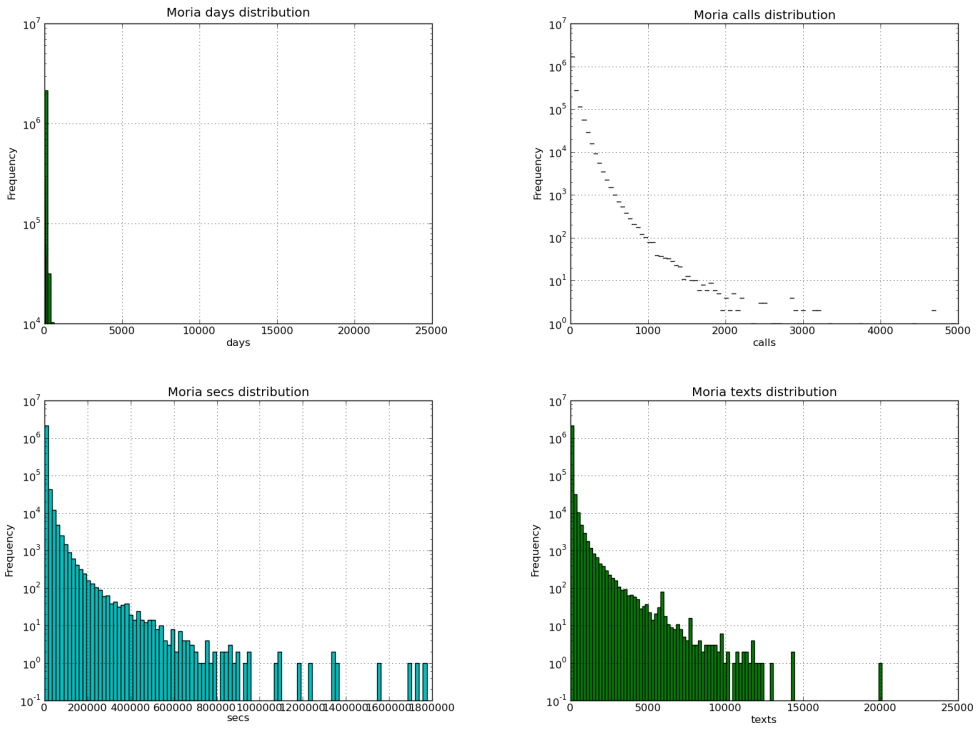


Figure 1: Moria Data Distribution

1.0.2 Vertex Attributes

In addition to the edge distribution the distribution of the the calls, texts and seconds of each vertex (as well as the degree) of the vertex was examined. This was computed by summing all of the call, texts and seconds of the edges connecting to a vertex. The days data was discarded because there is the possibility to have a commulative number of calls than the days in the month which does not have a physical basis. It was observed that there were a few (less than 5) verticies with a large number of entries, these were classified as anomolies in the network (probably call bots). Figure 3 shows the vertex distributions of Moria, while Figure 2 is for Standelf.

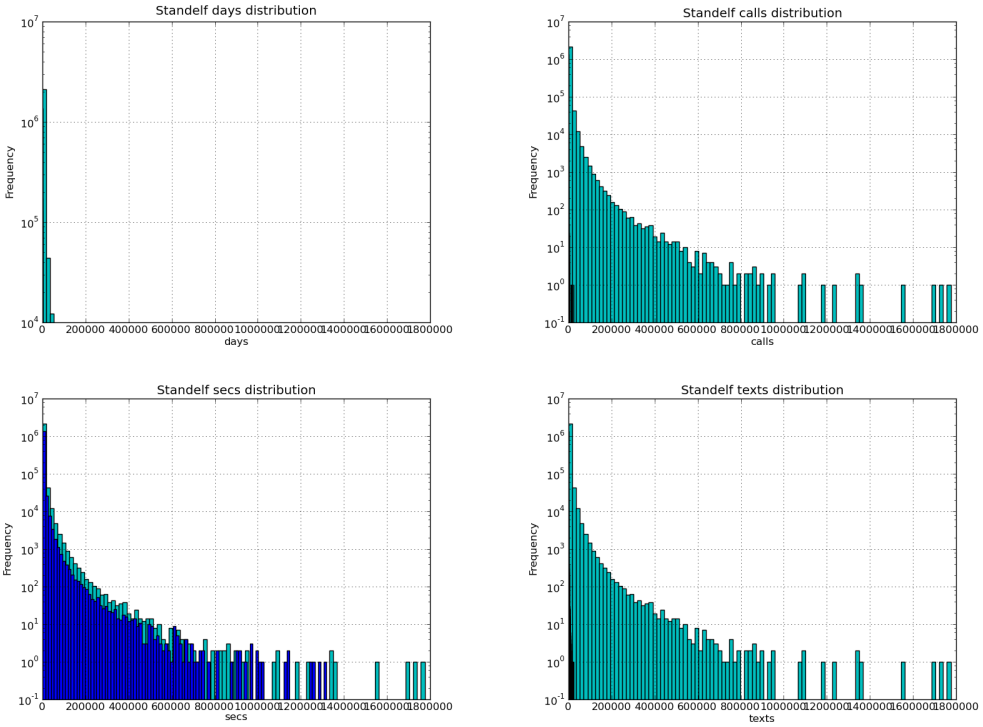


Figure 2: Standelf Data Distribution

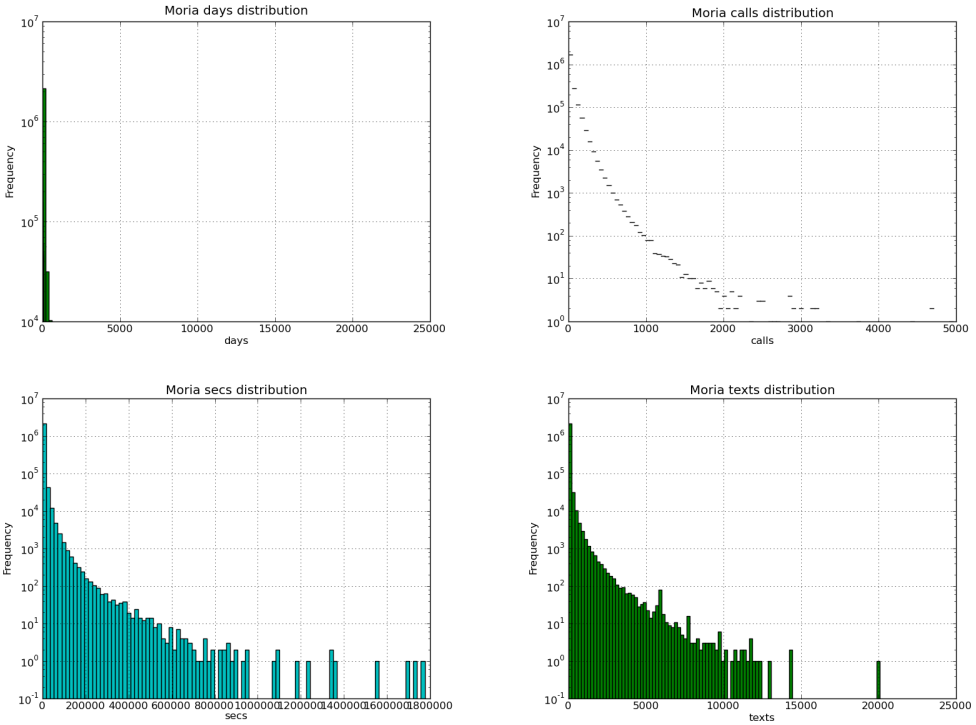


Figure 3: Moria Vertex Distribution

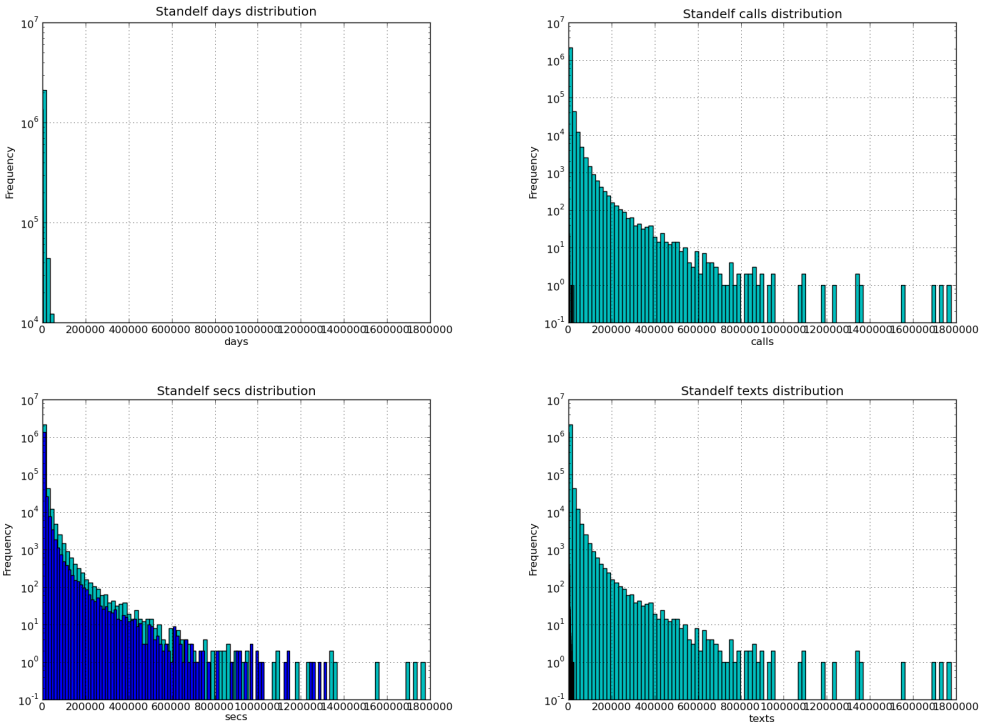
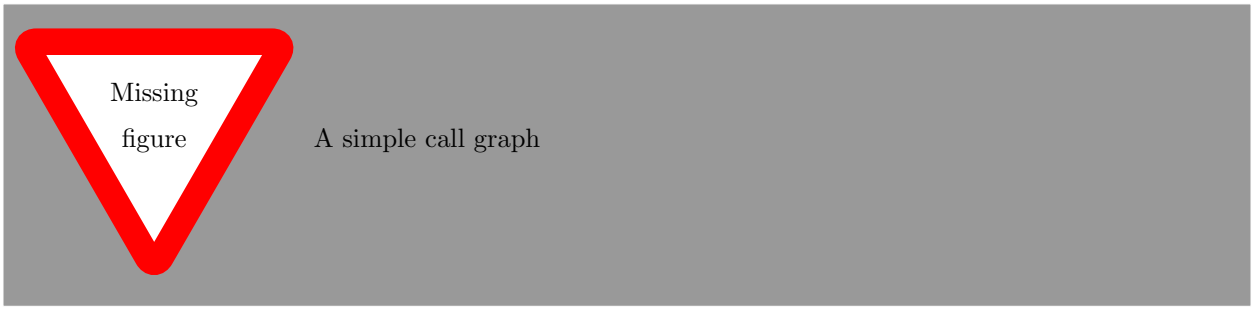


Figure 4: Standelf Vertex Distribution

2 Methods

The call graph was provided as a labeled edge list, and was modeled as a weighted, undirected, multigraph The graph was organzied as each of the nodes being the unique identifier to a cutomer, with edges between the customers representing:

- `days` - the number of distinict days that the two nodes communicated during the month,
- `calls` - the number of distinct calls that the the nodes made during the month,
- `secs` - the cummulative sum of all calls that the modes made during the month and,
- `texts` - the number of distinct texts that the two nodes made during the month.



A simple call graph

2.1 LSA Based Similarity Measure

Based on some work we did a few years ago looking at personality profile matching using an LSA based system, I wondered if the premise might hold that a node (person) could be effectively "described" by the set of other people they were connected to. This description could then be measured for similarity to other node descriptions and hopefully we would find that "similar" nodes were connected. In the past work we did there was some promise in the initial results for group analysis, but it was not pursued very far and there were other data items that were being considered as well (and it had nothing to do with telephone usage patterns).

To test the theory, I initially constructed a sparse matrix for each of the graph files for both the Moria and Standelf. For each connected node, the connection weight was determined by taking each of the 4 attribute values, dividing by thier respective standard deviation, and summing the results into a single weight for each connection. I also reran the experiment using a simpler weighting scheme giving a 1 if any call was made and another 1 if any text was made.

This sparse matrix in any case was extremely sparse, being around 0.00051% nonzero for Moria and 0.00059% for Standelf. We usually deal with matrices that have a nonzero rate of 0.001% to 0.01% for text mining applications. I computed a truncated SVD for these sparse matrices, forming an LSA space at approximately 250 dimensions for each. Then using these 250 dimenstion spaces I looked at comparing the first 1,000 nodes of each graph to all the other nodes in the graph computing their vector cosine similiarity and noting if the nodes were connected or not. This takes a while to run so there has not been much opportunity to tweak any of the parameters, therefore I cannot conclude with any certainty, but the initial results do not look promising. The first sets that I processed did not show any clear indication of connectivity determined by the similarity measure.

2.2 Artifical Neural Network Classification

The next approach implemented was an artifical neural network classification system. Using the `PyBrain` tool module a feed forward neural network was constructed with 8 inputs represting the edge between node `u` and `v`:

- the degree of node `u`,
- the closeness of node `u`,

- `days` of the edge,
- `calls` of the edge,
- `secs` of the edge,
- `texts` of the edge,
- the closeness of node `v`, and,
- the degree of node `u`.

The closeness of node was calucated using the `networkx` module with `closeness_centrality`, defined to be 1 over the average distance to all other nodes. The distances where not weighted, and all distances where normalized by the graph size. The degree was calculated with `degree` as the the sum of the edge weights of adjacent nodes for a particular node (completed for all nodes). The other elements of the training vector were simply filled with the values from the edge.

Training data for the neural network was then the set of all example nodes and edges presented in the target class of 1 (the edge exists). This was ultimately a flaw in the design of the neural network (or any classification based system) as discused in Section 3.

3 Resuls

3.1 LSA Based Similarity Measure

3.2 Artifical Neural Network Classification

The artifical neural network trained to a very low (less than 1%) error using 5-fold cross valdiation and back-propogation in 20 itterations. However, when tested the accuracy of predicting wether a node existed or not was 51%. This low accuracy is no better than flipping a coin. This is because the neural network was not feed any negative class examples; all the network had to learn was that an example input pattern corresponded to an edge. Negative input patters could not be created as network input because there is no guarentree that the input pattern would not exists, thus the network could be taught incorrect data.

4 Conclusions

It is concluded that due to the lack of negative training examples (edges that should not be there) any classification based system will preform with an accuracy of 50%.

5 Acknowledgements

The work was distributed as follows. John completed the analysis of the data with the LSI based similarity measure. Matthew completed the distributions of the data as well as the neural network classification system. He also implemented the probability based approach.