Internship Progress Report

My work was centered around the application of hypergraphs to location-based social networks (LBSNs). We started with implementing the LBSN2Vec algorithm in Python. The original implementation of this paper was based on MATLAB 2017b and C which made working with it tougher than it needed to be, so we implemented the complete algorithm in Python for location-prediction. The main idea behind the algorithm is to carry out a random walk on the friendship network and while sampling each node, check for edges in the other domains. While LBSN2Vec had the random walk approach, another algorithm referred to as DHNE (Deep Hyper-Network Embedding), argues that treating nodes in a hyperedge pairwise assumes homogeneity in variety of nodes which may not be true for every situation. So they propose to map the group of nodes to a probability space depending upon if they belong to the same hyperedge. This concept applies to LBSNs as well as there are 4 different domains of nodes. Thus we adapted the code to our problem. A similar concept of non-decomposable hyperedges was talked about in the paper that introduced the Hyper-SAGNN architecture. The difference between the architecture of Hyper-SAGNN and that of DHNE is that the former can work with different sizes of hyperedge whereas DHNE can work only with k-regular hypergraphs. Besides the Hyper-SAGNN architecture aims to model the relationship between the nodes within a hyperedge by comparing the embeddings of the nodes within each hyperedge, whereas DHNE models the same by mapping the embeddings to a probability space. For LBSN, the network can be modeled into a k-uniform hypergraph, so we stuck to DHNE.

Next point of action was to compare the results of location prediction by these two algorithms. The LBSN2Vec algorithm uses the social network as an additional information as compared to the DHNE algorithm. In order to remove that, we modified the former to not use the friendship graph. Instead, to achieve the random walk with stay procedure, we operated the random walk on a shuffled sequence of locations ids and for each node on that walk, sampled nodes from the other 3 domains.

Due to the size of the data and limited computational resources, I haven’t been able to get proper results for these implementations. Although I tried running for a smaller number of users, due to convergence issues, I doubt they depict the accuracy of the algorithms.

The LBSN2Vec algorithm, after training on 10 users, 196 training checkins, 2 dimensions, for 1 epoch, gives an accuracy of 4% for 20 test checkins and 16% for 50 test checkins. We could check DHNE’s performance on the entire data, again divided in the 80-20 ratio for testing and training data and trained for 5 epochs and 2 dimensions, that gave an accuracy of 0.7% for top 10 test checkins.

I am still working on fixing a reasonably small size of data that can run in viable time for both the algorithms. Current steps are aimed at deciding a reasonably small size of data along with other parameter updates that produce better results. Precisely, we aim to observe the results by tuning the number of epochs, embedding size, batch size, learning rate, hidden layer’s size, number of negative samples considered and the length of top results considered for testing. We hope to tune the model well and compare the performances of modified LBSN2Vec and DHNE in a faithful setting.