Evolution of Friend Lists in Social Networks

Motivation

An unweighted and undirected social graph can change in three ways:

- **Adding Members**
  - Alice adds Bob to her friends list.
  - Alice adds Carol to her friends list.
- **Removing Members**
  - Alice removes Bob from her friends list.
- **Changing Connections**
  - Alice removes Carol from her friends list.
  - Alice removes Bob from her friends list.

Friend lists likely change alongside the social graph in similar ways:

- **Adding Members**
  - Lunch Group adds Dave to their members.
- **Removing Members**
  - Lunch Group removes Bob from their members.
- **Changing Connections**
  - Lunch Group removes Dave from their members.

Approaches

Since this is a first attempt at this problem, we restricted our evolutions to users who only add and never remove members from their social graph. We then tested three approaches:

**Manual**

<table>
<thead>
<tr>
<th>Current Social Graph</th>
<th>Old Social Graph</th>
<th>Old Friend Lists</th>
<th>User Effort</th>
<th>Specified Friend List Evolutions</th>
<th>New Friend Lists</th>
</tr>
</thead>
</table>

**Full Recommendation**

<table>
<thead>
<tr>
<th>Current Social Graph</th>
<th>Existing New Friend List Recommender</th>
<th>User Effort</th>
<th>New Friend Lists</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Existing New Friend List Recommender</th>
<th>User Effort</th>
<th>Edit and label new recommended friend lists</th>
<th>New Friend Lists</th>
</tr>
</thead>
</table>

**Change Recommendation**

<table>
<thead>
<tr>
<th>Current Social Graph</th>
<th>Existing New Friend List Recommender</th>
<th>User Effort</th>
<th>Match and merge friend lists and recommender results</th>
<th>Evolved Friend Lists</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Old Social Graph</th>
<th>Old Friend Lists</th>
<th>Match and merge friend lists and recommender results</th>
<th>Evolved Friend Lists</th>
</tr>
</thead>
</table>

**Match and Merge Details**

- 1-to-1 mappings of old friend lists to recommended lists
- Expects friend lists to grow at the same rate as the social graph
- Expects members that already existed in the social graph to be apart of the same friend lists

```java
integratedMatchAndMerge(oldLists, recommendations) {
    recommend_evolutions = empty set
    threshold = 0
    while length(oldLists) > 0 && length(recommendations) > 0 {
        foreach oldList in oldLists {
            matchedVals = {}
            foreach recommendation in recommendations {
                if closeness(oldList, recommendation) <= threshold {
                    matchedVals.append(recommendation)
                }
            }
            if matchedVals.size() == 1 {
                recommended_evolutions.add(merge(oldList, matchedVals[0]))
                recommendation.remove(matchedVals[0])
                oldLists.remove(oldList)
            }
            threshold += 1
        }
        return recommended_evolutions
    }
    return <expected growth, 0 adds, 0 deletes>,
    <actual growth, added old social graph members,
    removed old social graph members>
}
```

**Evaluation**

We evaluated the two recommendation approaches relative to manual in terms of cost (adds and deletions) required by the users.

**Dataset**

Using data from a user study of 12 individuals in Facebook, generated an old state of each users friend list by removing a randomly selected set of members from both the social graph and its friends lists.