Social Scope: Enabling Information Discovery on Social Content Sites

Authors: S.A. Yahia (Yahoo Inc), L.V. Lakshmanan (UBC), Cong Yu (Yahoo Inc)

Presenter: Tony Wu

Date: 11/20/2014

Overview

- Motivation
- Goals
- Architecture
- Social Content Graph Model
- Information Discovery Layer
- Collaborative Filtering Example
- Content Management Layer
- Information Representation Layer
- Conclusions

Motivation

- Integration of social and content sites has led to the emergence of social content sites and social content graphs.

- Traditional IR system only return results that are semantically relevant, but not socially relevant.

<table>
<thead>
<tr>
<th></th>
<th>general (e.g., things to do)</th>
<th>categorical (e.g., family)</th>
<th>specific</th>
</tr>
</thead>
<tbody>
<tr>
<td>with locations</td>
<td>32.36%</td>
<td>22.52%</td>
<td>8.37%</td>
</tr>
<tr>
<td>w/o locations</td>
<td>21.38%</td>
<td>5.34%</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Summary Statistics of 10 Million Y!Travel Queries.

Motivation - Example

- John has a day off from his conference in Toronto, he searches for “Toronto Attractions” on Y! Travel

- John has in the past visited a few “baseball fields” on Y! Travel and on Facebook he has many friends with interests in “baseball”

- Traditional IR systems would return all the generic popular attractions based on semantic relevance (e.g. CN Tower), but not baseball related (e.g. Rogers Center)

- Social-Scope solves this problem by incorporating social relevance its search results (e.g. consider John’s friends interests)

Goals

- Query results must be both semantically relevant and socially relevant
- Efficient content management
- Effective navigation of search results

Figure 1: Architecture of SocialScope.
Social Content Graph Model

Nodes:
- Represent users and entities like restaurants and attractions
- Each node has an ID and a set of attributes (‘type’ attribute is mandatory)
- E.g. \( n_1 = \{ \text{id} = 1; \text{type} = \text{‘user, traveler’}; \text{name} = \text{‘John’} \} \)

Links:
- Represents relationships between nodes (e.g. “visited”)
- Can also used to represent user tags
- E.g. \( L_{12}(n_1, n_2) = \{ \text{id} = 12; \text{type} = \text{‘act, tag’}; \text{date} = \text{‘2008-8-2’}; \text{tags} = \text{‘Rockies baseball’} \} \)

Information Discovery Layer

Unary Operators:

- Node Selection: $\sigma^N_{<C,S>} (G)$
- Link Selection: $\sigma^L_{<C,S>} (G)$

Outputs and a set of nodes (for NS) or links (for LS) that satisfies condition C, and their scores based on the scoring function S

Basic Binary Operators (Set theoretic Operators):

- Union: $G_1 \cup G_2$
- Intersection: $G_1 \cap G_2$
- Difference: $G_1/G_2$

Applies to both links and nodes

Information Discovery Layer

- Advanced Binary Operators:
  - **Composition**: \( G = G_1 \odot_{<\delta,F>} G_2 \)
    - Creates new links between the nodes in G1 and G2 satisfying the directional condition \( \delta \)
    - Directional constraints have values src or tgt (e.g. \( \delta = (\text{src}, \text{tgt}) \) means source node of G1 matches target node of G2
    - \( F \) specifies the composition function of how the attributes of the new links should be determined based on the attributes of the input links
  - **Semi-Join**: \( G_1 \bowtie_\delta G_2 \)
    - Produces a sub-graph of G1 where the links of G1 matches the links of G2 based on the directional condition \( \delta \)
    - E.g. \( G_1 \bowtie_{(\text{tgt},\text{src})} G_2 \) produces a sub-graph of G1 where the all links of the sub-graph have target nodes that matches the source nodes of G2

Information Discovery Layer

- Aggregate Operators:
  - **Node Aggregation**: $\gamma_{<C,d,att,A>}^N(G)$
    - Aggregates the links of the nodes based on aggregate function $A$ satisfying condition $C$ and directional parameter (src or tgt)
    - The aggregation function produces a new node attribute $att$
  - **Link Aggregation**: $\gamma_{<C,att,A>}^L(G)$
    - Replaces the set of links satisfying $C$ and have the same source and target nodes where the new link has attribute $att$ of which the value is determined by $A$

---

Collaborative Filtering Example

Problem:
Recommendation of travel destinations to John based on his social network

Solution:
1. \( G_1 = \sigma_{type=’visit’}^L \left( G \bowtie_{src,src} \sigma_{id=101}^N (G) \right) \)
   - User John and places he has visited
2. \( G_1' = \gamma_{type=’visit’,src,vst,A}^N (G_1) \)
   - Set of destinations John has visited and stores in the vst att of node JOHN
3. \( G_2 = \sigma_{type=’visit’}^L \left( G \bowtie_{src,src} \sigma_{id\neq 101}^N (G) \right) \)
   - Users other than John and the places they have visited
4. \( G_2' = \gamma_{type=’visit’,src,vst,A}^N (G_2) \)
   - Set of destinations that other users have visited and stores in the vst att of the nodes

Collaborative Filtering Example

Solution – Cont’d

5. \( G_3 = G_1 \odot \langle \delta, F \rangle \ G_2 \)
   - \( \delta = (tgt, tgt) \) and \( F \) is the composition function that computes the Jaccard similarity, which is stored in links produced from John to another user.

6. \( G_4 = \gamma_{<\text{sim}>0.5,\text{type},A'}^L(G_3) \)
   - Replaces score of the links with similarity > 0.5 with string “Match”

7. \( G_5 = \sigma_{\text{type}='\text{visit}'}(G) \bowtie_{\text{tgt}, \text{src}, A'}(G_3) \)
   - All destinations that users have visited

8. \( G_6 = (G_4 \bowtie_{\text{tgt}, \text{src}} G_5) \odot \langle (\text{tgt}, \text{src}), \text{sim}_{\text{sc}}, F' \rangle (G_5 \bowtie_{\text{src}, \text{tgt}} G_4) \)
   - For each of John’s similarity network friends who has visited a destination, a new link is added from John to that destination. \( F' \) copies sim score from link of John to user to new link

9. \( G_7 = \gamma_{\text{score}, \text{AVERAGE}}^L(G_3) \)
   - Replace set of links from John to destination node with attribute score, which is the average of the similarity score, score can be used to rank destinations

Content Management Layer

- Provides efficient storage of indexes to support keyword-based queries

- Inverted list (IL) is used to store the item scores for each (tag, user) pair, each entry of the list consists of the form \((i, \text{score}_k(i,u))\), where \(k = \text{tag}, l = \text{item} \) and \(u = \text{user}\)

- Score for each item is calculated by summing the item score across each IL of all (tag, user) pair

- Problem: Storing a list of scores for every (tag, user) pair can consumes a huge amount of storage since the number of items and users can be huge (requires 1 terabyte for a site with 100,000 users, million items and 1000 distinct tags!)

Clustering will reduce the storage complexity of storing inverted list.

Score of the cluster will be calculated as follows:

\[ \text{Score}_k(i, C) = \max_{u \in C} \text{score}_k(i, u) \]

- **Network-based Cluster:**

  \[
  \left| \frac{\text{network}(u_1) \cap \text{network}(u_2)}{\text{network}(u_1) \cup \text{network}(u_2)} \right| \geq \theta
  \]

- **Behavior-based Cluster:**

  \[
  \left| \frac{\text{items}(u_1) \cap \text{items}(u_2)}{\text{items}(u_1) \cup \text{items}(u_2)} \right| \geq \theta
  \]

Information Presentation Layer

- Results are grouped to allow users to navigate the results more effectively

- Social Grouping:

  \[
  \frac{|\text{taggers}(i_1) \cap \text{taggers}(i_2)|}{|\text{taggers}(i_1) \cup \text{taggers}(i_2)|} \geq \theta
  \]

- Provides explanation to items to allow user realize social provenance:
  - Content-based strategy
  - Collaborative filtering strategy

Conclusions

- Social Scope is able to search for results not only semantically relevant, but also social relevant

- Information discovery layer is based on a set of graph processing operators to return socially relevant results from the social content graph

- Content Management layer leverages clustering to efficiently store inverted index for key-word based search

- Search results are grouped in the presentation layer based on social grouping and descriptions are attached to results to realize social provenance

Thank You!

Questions?