Tag Usage in Highly Viewed Photos on Flickr

Sana Malik
maliks@cs.umd.edu

November 8, 2012

1 Project Description

Flickr.com is a popular online photo-sharing community that contains millions of images from various users around the world. Many of the images are tagged by the user with useful information about the image’s subject, location, or technical details. Given this information, I try to answer two main questions:

- What is the correlation between views and tags?
- Can tags be leveraged to garner more views on a photo?
- How do the most popular users on Flickr use tags?

2 Data

Source

The data used is from the Stanford Network Analysis Project, which consists of over 14,000 photos from Flickr. The photos are a subset of the MIRFLICKR-1M data set, which have been manually annotated.

Description

I began by selecting the 500 most viewed photos from this data set. Each photo had the following properties:

- ID: the photo’s unique ID, as assigned by Flickr.
- Author: the username of the photo publisher.
- URL: the URL to the photo’s page.
- Views: the total number of times a photo has been viewed, ranging from 12,472 to 475,296.
- Comments: the number of comments left for the photo, ranging from 1 to 365.
- Tags: an alphabetized string of all the photo’s tags.
- Tag Count: the total number of tags assigned to the photo, ranging from 0 to 75.

An edge between two photos means that the photos had at least one tag in common. Edges had the following properties:

- Number of Tags in Common: the number of tags shared between the two photos.
- Number of Total Tags: the total number of unique tags the two photos had.
- Similarity: the number of tags in common divided by the total number of tags.
Graph Overview

<table>
<thead>
<tr>
<th>Graph Type</th>
<th>Undirected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertices</td>
<td>500</td>
</tr>
<tr>
<td>Edges</td>
<td>7798</td>
</tr>
<tr>
<td>Connected Components</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 1: Complete graph with no filtering or grouping. Force-directed layout. Vertex size is view count. Intended to show the overall structure of the nodes.

3 Insights

3.1 Popular users re-use the same tags on multiple pictures.

[Graph shown in Figure 2]

- Edge filter: Show only if Similarity is above 0.2
- Vertex filter: None
- Vertex size: Number of views
- Vertex color: Group
- Vertex label: None
- Layout: Fruchterman-Reingold, Group in a box
- Grouping: By author
- Group filter: Show only if group contains 1 or more edge
The strongest connections are between photos by the same user, as expected. This graph had edges filtered by photos with 20% similar tags, and when grouped by user, the majority of the edges are within each box. An obvious cause of this is that a single user may take multiple photos of the same event or object. However, considering these photos are the 500 most viewed photos out of 14,000 possible pictures, it is more likely that popular users create a set of "base tags" that includes less subject-relevant information, such as their username and camera model.

3.2 The most popular photos are not similar each other.

[Graph shown in Figure 3]

- Edge filter: None
- Vertex filter: Show view count above 150,000 only
- Vertex size: Number of views
- Vertex color: Group
- Vertex label: Author
- Layout: Fruchterman-Reingold, Group in a box
- Grouping: Wakita-Tsurumi clustering

The subgraph of the most viewed photographs is very sparse, indicating that these photos are not similar to each other. Further, the vertices are not centralized to a single cluster, meaning when photographs are similar to each other, there are only a few that become extremely popular (150,000+ views).
3.3 In general, more tags will decrease the number of views.

[Graph shown in Figure 4]

- Edge filter: Hide all
- Vertex filter: none
- Vertex size: Number of comments
- Vertex color: Group
- Vertex label: None (explained below)
- Layout: None (XY coordinates)
- X-axis: Number of Tags
- Y-axis: Number of Views
- Grouping: Wakita-Tsurumi clustering

Surprisingly, adding more tags to a photo will not increase the amount of views it receives. It may seem that having more tags for a photo increases its visibility in searches and thus garners more views, but Figure 4 shows this is not the case. Instead, there is a loose correlation in the opposite direction, with one notable outlier (shown in tooltip). There does seem to be a “sweet spot” where adding more tags increases the visibility (between 5 and 20 tags), but in every case, the correlation is loose.

Here, the vertex labels are omitted since we are looking for a trend between the photos, instead of information about particular nodes. The tooltip is set to the photo ID though, so upon finding interesting nodes, one can identify the photo easily. This is augmented by NodeXL’s brushing marker feature which will select the corresponding row in the Vertices spreadsheet when a node is selected.
4 NodeXL: Pros and Cons

4.1 Benefits

NodeXL’s primary advantage is its flexibility in multiple areas. For example, NodeXL supports 4 different ways to import data (including both directly importing from a network/website or from an existing file), 10 algorithm layouts that serve a variety of purposes (force-directed, geometric, or layered), and 3 methods for clustering (by vertex attribute, clustering algorithm, or connected component). The ease with which NodeXL integrates with already existing tools is especially important because it allows for a wider user base. NodeXL also features automatic graph metric analysis. Though the graph metrics are not novel by themselves, I felt the inclusion of basic word frequency analysis to be a characterizing difference between this tool and others. This was particularly helpful in viewing the most frequently used tags for each group in the cluster.

4.2 Weaknesses and Suggested Improvements

While the features NodeXL has are extremely useful, I feel it is cumbersome to use on data sets larger than 500 nodes. Many times, I had to wait over a minute for the graph to render after making small changes. Further, while the coordinated windows between the sheet and graph are helpful, I feel this functionality could be augmented by two changes: 1) automatically grouping the selected vertices’ rows in the excel spreadsheet, and 2) automatically scrolling to the corresponding rows of selected vertices.

The current filters are useful when removing noise from the graph, however they are less useful for seeing changes in the graph structure without the filter edges and nodes. NodeXL does allow vertices and edges to be hidden from the structure of the graph via the "Visibility" column in their respective sheets, however these changes do not update across the sheets. For example, if a set of edges are set to ”hidden” and this results in some vertices to be edgeless, those vertices are still displayed on the graph with no way to hide them, or remove them from the structure without creating an entirely new workbook. There is a similar problem with the group-in-a-box layout: when all vertices from a given group a hidden, the algorithm still displays the empty box.
With the group-in-a-box layout, the force-directed, geometric, and layered graph layouts work very well. They apply the algorithm among each subgraph/box. However, with the (x,y)-coordinate layout, the node positions are always set from the bottom left, and the group-boxes are entirely ignored, as shown below in Figure 5.

![Figure 5](image.png)

Figure 5: When the "None" layout is selected with the "Group-in-a-box" option, a non-sensical graph is shown.

This could be remedied one of two ways. Either have the (x,y) positioning be relative to the bottom left of each box, or remove the box boundaries when the "None" layout option is selected.