

**Project Proposal:
A Functional-Logic Model of Folksonomic Tagging and Search
for Travel Photos**

Ke Deng, Chenguang Gao

Supervised by: Dr. Harold Boley

March 26, 2007

Problem Statement

The quickly growing popularity of digital cameras has enabled more and more people around the world to share their travel photos through the Internet. An effective and efficient search engine can greatly reduce users' effort on locating the correct photos. Due to synonyms (different words having the same meaning) and homonyms (the same word having different meanings) in natural languages, a search engine based on *words* in, say, photo caption *texts* is not capable of obtaining high recall and precision. Some search engines thus allow users to browse the *categories* of a predefined *taxonomy* to find a photo, but this browsing process can be tedious, and becomes impractical for leaf categories of the taxonomy that still contain a large number of photos. An intermediate approach uses *tags* that are user-defined in a travel *folksonomy* (a light-weight, socially-constructed taxonomy) and associated with photos. We decided to build our photo search engine based on folksonomic tags. Therefore, it is necessary to design and implement a folksonomic tagging and search model for sharing travel photos.

Proposed Solution

In this project, we are going to design and implement a functional-logic model of folksonomic tagging and search for travel photos using Relfun. The index is modeled as a user-defined Relfun knowledge base of tags, and search queries are answered from it returning photo URLs as results.

In the **indexing phase**, photos are tagged in the folksonomic way, which means users can freely add tags to photos and later these tags can be shared by all users. Each tag is associated to one or more photo URLs; conversely, each URL can also have one or more tags. The tag-photo associations can have different *certainty* values. Tags can be specified to be folksonomic subtags/supertags of other ones.

In the **querying phase**, one or more photo tags are given, possibly with certainty thresholds, and the most accurate photos (i.e. URLs) are searched based on the index.

If no exact matches are found, tags similar to the given one will be retrieved and another search be attempted based on those weaker tags and a correspondingly strengthened threshold value. This requires a series of well-defined `similar()` functions.

A `similar()` function returns a value (between 0 and 1) that represents the degree of similarity between two tags. The returned similarity value is also used for calculating the strengthened threshold value of the `search()` function. Strengthened threshold values can be calculated using a logarithm-like function. In our sample code we use the original threshold value as a placeholder for a function call within the `strengthen()` function. Also, `subtag/supertag` relationships can be employed to search for the URLs of more special/general tags.

Moreover, this model will be generalized to query a conjunction of n tags, each with its own threshold value. Every tag and its threshold value create an independent search subquery, whose returned URLs are forced to be identical throughout the conjunction via Relfun's `.=` primitive: Each subquery returns a set of result URLs, and the result of the conjoined query is the `.=` intersection of the sets returned by all subqueries.

Reasons for Using Relfun

Relfun is a functional-logic programming language. Our project makes use of both functional and logic aspects of Relfun. For example, we used non-determinism to enumerate (all) URLs with given tags. Non-determinism originated in logic programming, but is used here in a functional program. This extended functional program is furthermore based on logic programming facts (e.g., defining the unary relation `threshold()`).

Our model allows a function call with non-ground arguments; again, having non-ground terms is a typical feature of logic programming. Thus, we use logic facts as well as functional non-ground arguments and non-determinism of functional-logic programming in our model. This makes the design and implementation easier than using either only functional programming or only logic programming.

Optional Functionalities

Cyclic similarity chains could cause an internal infinite loop, which would create a hazard for our program. To prevent such infinite loops, we consider to lexicographically order the two arguments of similarity relations and an upper bound on the length of similarity chains. On the other hand, we will try to cope with the symmetry property of the `similar()` relation. Since it is redundant to show duplicate results we will consider to collect solutions with Relfun's `tupof` primitive and eliminate duplicates in the resulting explicit list.

Description of Sample Code

In the following sample code, there are two `search()` functions. The first search function serves as the base case of the recursive search. The second one searches through the tags recursively. The `obj()` function associates tags with URLs at some certainty; the `similar()` function defines the degree of similarity between two tags; the `strengthen()` function calculates a new threshold value based on the current threshold value and the similarity value returned by the `similar()` function; the `threshold()` relation is used to ground the variable `Threshold`, e.g. because Relfun built-in relations such as `'>='` do not accept free variables as arguments.

Sample Queries

```
rfi-p> tupof(search(china, 1.0))  
["http://www.chinapictures.org",  
 "http://china.travelphoto.net",  
 "http://www.travelchinaguide.com/picture"]
```

```
rfi-p> tupof(search(china, 0.5))  
["http://www.chinapictures.org",  
 "http://china.travelphoto.net",  
 "http://www.travelchinaguide.com/picture",  
 "http://historylink101.com/china_history.htm",  
 "http://www.maps-of-china.com/",  
 "http://www.chinapictures.org/type/nature-scenes/yangtze-river"]
```

```
rfi-p> tupof(search(river, 0.8))  
["http://www.chinapictures.org/type/nature-scenes/yangtze-river",  
 "http://www.imagequest3d.com/photos/water/index.htm",  
 "http://www.goobix.com/black-sea-pictures"]
```

```
rfi-p> search(river, X)  
"http://www.chinapictures.org/type/nature-scenes/yangtze-river"  
X=0.9
```

```
rfi-p> tupof(search(river, X))  
["http://www.chrs.ca/Rivers_e.htm",  
 "http://www.chrs.ca/Rivers_e.htm",
```

```

"http://www.chrs.ca/Rivers_e.htm",
"http://www.chrs.ca/Rivers_e.htm",
"http://www.chrs.ca/Rivers_e.htm",
"http://www.chrs.ca/Rivers_e.htm",
"http://www.chinapictures.org/type/nature-scenes/yangtze-river",
"http://www.chinapictures.org/type/nature-scenes/yangtze-river",
"http://www.chinapictures.org/type/nature-scenes/yangtze-river",
"http://www.chinapictures.org/type/nature-scenes/yangtze-river",
"http://www.chinapictures.org/type/nature-scenes/yangtze-river",
"http://www.chinapictures.org/type/nature-scenes/yangtze-river",
"http://www.chinapictures.org/type/nature-scenes/yangtze-river",
"http://www.chinapictures.org/type/nature-scenes/yangtze-river",
"http://www.chinapictures.org/type/nature-scenes/yangtze-river",
"http://www.chinapictures.org/type/nature-scenes/yangtze-river",
"http://www.imagequest3d.com/photos/water/index.htm",
"http://www.imagequest3d.com/photos/water/index.htm",
"http://www.imagequest3d.com/photos/water/index.htm",
"http://www.imagequest3d.com/photos/water/index.htm",
"http://www.imagequest3d.com/photos/water/index.htm",
"http://www.imagequest3d.com/photos/water/index.htm",
"http://www.imagequest3d.com/photos/water/index.htm",
"http://www.imagequest3d.com/photos/water/index.htm",
"http://www.goobix.com/black-sea-pictures",
"http://www.goobix.com/black-sea-pictures",
"http://www.goobix.com/black-sea-pictures",
"http://www.goobix.com/black-sea-pictures",
"http://www.goobix.com/black-sea-pictures",
"http://www.goobix.com/black-sea-pictures",
"http://www.goobix.com/black-sea-pictures",
"http://www.goobix.com/black-sea-pictures",
"http://www.goobix.com/black-sea-pictures"

```

```

rfi-p> tupof(Url.=search(china, 0.7), Url.=search(river, 0.5))
["http://www.chinapictures.org/type/nature-scenes/yangtze-river"]

```

Sample Code (in Relfun):

```

% Return the corresponding URL(s) based on a tag and its threshold value.
obj(china, 1.0) :& "http://www.chinapictures.org".
obj(china, 1.0) :& "http://china.travelphoto.net".
obj(china, 1.0) :& "http://www.travelchinaguide.com/picture".
obj(china, 0.6) :& "http://historylink101.com/china_history.htm".

```

```
obj(china, 0.5) :& "http://www.maps-of-china.com/".
obj(china, 0.8) :& "http://www.chinapictures.org/type/nature-scenes/yangtze-river".
obj(china, 0.3) :& "http://www.orientaltravel.com/china/china_city.htm".
```

```
obj(river, 0.6) :& "http://www.chrs.ca/Rivers_e.htm"
obj(river, 0.9) :& "http://www.chinapictures.org/type/nature-scenes/yangtze-river".
obj(water, 0.8) :& "http://www.imagequest3d.com/photos/water/index.htm".
obj(sea, 0.9) :& "http://www.goobix.com/black-sea-pictures".
```

```
% Return the degree of similarity between two tags.
similar(river, water) :& 0.8.
similar(river, sea) :& 0.9.
```

```
% Tag search engine core.
```

```
% Base case
search(Tag,Threshold) :-
    Found .= obj(Tag,Certainty),
    threshold(Threshold),
    >=(Certainty,Threshold) &
    Found.
```

```
% If no exact tag matches, use similar tags instead.
search(Tag,Threshold) :-
    Similarity .= similar(Tag,Simtag),
    threshold(Threshold),
    >=(Similarity,Threshold) &
    search(Simtag,strengthen(Threshold,Similarity)).
```

```
% Only a placeholder (later will be changed to a logarithm-like function of
% Threshold instead)
strengthen(Threshold,Similarity) :& Threshold.
```

```
% Enumerating legal thresholds:
```

```
threshold(1.0).
threshold(0.9).
threshold(0.8).
threshold(0.7).
threshold(0.6).
```

```
threshold(0.5).  
threshold(0.4).  
threshold(0.3).  
threshold(0.2).  
threshold(0.1).
```

```
% Conjunction of n tags (will be implemented later)  
% searchn(china, 0.7, river, 0.5)  
% Url.=search(china, 0.7), Url.=search(river, 0.5)
```

Filename: ProjectProposalFolksonomicTravelPhotos.doc
Directory: C:\Documents and Settings\boleyh\Desktop
Template: C:\Documents and Settings\boleyh\Application
Data\Microsoft\Templates\Normal.dot
Title:
Subject:
Author:
Keywords:
Comments:
Creation Date: 3/11/2007 12:39:00 AM
Change Number: 129
Last Saved On: 3/26/2007 5:59:00 PM
Last Saved By:
Total Editing Time: 2,227 Minutes
Last Printed On: 3/31/2007 6:27:00 PM
As of Last Complete Printing
Number of Pages: 6
Number of Words: 1,485 (approx.)
Number of Characters: 8,469 (approx.)