Enhancing Personal File Retrieval in Semantic File Systems with Tag-Based Context

Hung Ba Ngo[1,2], Frédérique Silber-Chaussumier[1], Christian Bac[1]

Institut National des Télécommunications-France[1], Cantho University-Vietnam[2]
{hung.ngo_ba, frederique.silber-chaussumier, christian.bac}@int-edu.eu

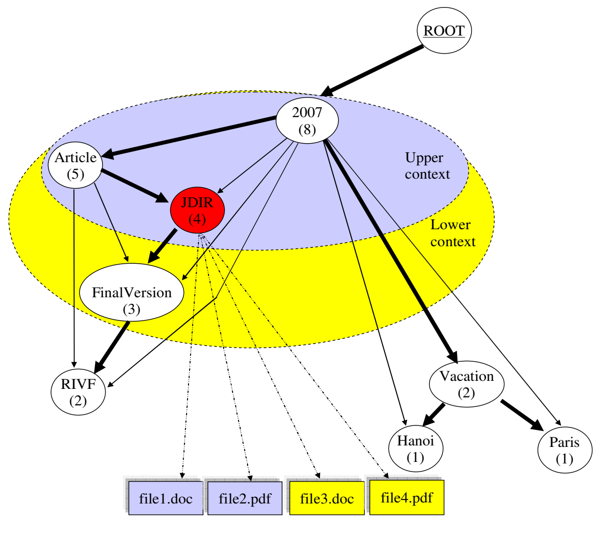
Abstract. Recently, tagging systems are widely used on the Internet. On desktops, tags are also supported by some semantic file systems and desktop search tools. In this paper, we focus on personal tag organization to enhance personal file retrieval. Our approach is based on the notion of context. A context is a set of tags assigned to a file by a user. Based on tag popularity and relationships between tags, our proposed algorithm creates a hierarchy of contexts on which a user can navigate to retrieve files in an effective manner.

1 Introduction

Nowadays, tagging systems such as (Delicious) are widely used on the Internet. These tagging systems enable users to add keywords (or tags) to Internet resources without relying on a controlled vocabulary. On the desktop, tags are also supported by some semantic file systems and desktop search tools. Users in LFS (Padioleau, 2005), for example, can manually assign tags to a Jpeg file to annotate the names of persons in that photo for later retrieval. With tags, users are flexible in describing their opinions and interests on files (or resources). As a result, users' personal files are classified per tags and each user has a *personal vocabulary* of tags. Users then can retrieve files using logical expressions of tags. By default, tagging systems are more suitable for file retrieval using querying than browsing. However, experiments in personal information management (Barreau et al., 1995), and (Khoo et al., 2007) show that most users prefer browsing than querying (logical search) as retrieving their files from a desktop. That is the reason why recently tagging systems such as Delicious on web or LFS (Padioleau, 2005) and TagFS (Bloehdorn et al., 2006) on desktop concentrate on tag organization to help users to browse tags for file retrieval. We continue to enhance personal file retrieval in tagging systems with *context-based searching*. A context in our approach is a set of tags assigned to a file (or resource) by a user. Based on tag popularity and relationships between tags, our proposed algorithm creates a hierarchy of contexts on which a user can navigate to retrieve files in an effective manner. In this paper, we first present the interesting techniques for tag organization in section 2; introduce tag-based context and how to enhance tagging systems with context-based searching in section 3. Our algorithm for creating a *Directed Acyclic Graph of Tag*s (DAGoT) based on tag popularity and relationship of tags is in section 4. This DAGoT is used to organize contexts into a hierarchical structure so that we can enhance personal file retrieval with context-based searching. An implementation and experimental results using real data are presented in section 5. Our conclusion and perspectives are in the last section.

2 Related Works

(Delicious) is a well-known online bookmark server where users can use their own tags to organize and retrieve their bookmarks. In Delicious, two or more tags associated to the same bookmark are considered as *related*. The number of bookmarks associated to a tag by a user is called the *popularity of tag*. When a tag is chosen, a list of bookmarks tagged with it and a list of its related tags are returned. Related tags are the way to navigate between tags to revisit interesting bookmarks. However, when the number of bookmarks and tags increase, parsing the result for a bookmark or choosing a right tag from related tags to refine search result becomes a tremendous task for a user. On desktops, tags are also used for personal file retrieval. Users in Spotlight (Apple Computer, 2005) can assign a keyword to a set of semantically related files and retrieve those files using a simple keyword search. In the file system domain, LFS (Padioleau, 2005) allows users to associate files with tags representing file properties. LFS supports axioms between tags as a parent-child relationship. Users can manually create axioms between their tags. From the axioms, a taxonomy of tags is created. Users can navigate on the taxonomy for file retrieval as they do with traditional directories. TagFS (Bloehdorn et al., 2006) organizes tags in related tags as the way of Delicious. Consequently, TagFS has the same difficulty when the number of files and tags increases.



C. – *The Corresponding DAGoT*

File	Tags assigned to file
file1.doc	{Article, JDIR, 2007}
file2.pdf	{Article, JDIR, 2007}
file3.doc	{Article, JDIR, 2007, FinalVersion}
file4.pdf	{Article, JDIR, 2007, FinalVersion}
file5.doc	{Article, 2007, RIVF}
file6.pdf	{2007, FinalVersion, RIVF}
file7.jpg	{2007, Vacation, Paris}
file8.jpg	{2007, Vacation, Hanoi}

A. – *Example of postings made by a user.*

Tag	Popularity
2007	8
Article	5
JDIR	4
FinalVersion	3
RIVF	2
Vacation	2
Paris	1
Hanoi	1

B. – *Popularity of tags*

FIG. 1 – *Example of a DAGoT created from tag popularity and relationship of tags.*

3 Definition of the tag-based context

A user in a tagging system makes a *post* to assign a file with a set of tags. Each tag represents a concept or an object related to the file. And the set of tags assigned to the file represents a topic, or a subject that file owner thinks the file concern. For example, a user may assign the set of tags {Vacation, Paris, 2007} to file myphoto.jpg to recall that the photo was taken during vacation of summer '07 in Paris. We call a set of tags assigned to a resource by a user a *tag-based context* (or context for short). The meaning of a context is aggregated from its elements. A context is more meaningful than a tag. For example, the context {Vacation, Paris, 2007} is quite more relevant than the tag 2007. In fact, when assigning a set of tags to a file, a user wants to classify the file using context represented by that set of tags for later retrieval. So tagging systems should provide users with file retrieval replying on context - characteristics used when classifying files.

The figure 1.A is an example of postings made by a user. If the user makes a tag-based searching with tag Article, five files (from file1 to file5) will be returned. These files belong to three contexts {Article, JDIR, 2007}, {Article, JDIR, 2007, FinalVersion} and {Article, 2007, RIVF}. A tag usually participates in many contexts. The figure 1.A above shows that tag JDIR participates in two related contexts {Article, JDIR, 2007} and {Article, JDIR, 2007, FinalVersion}. The former is more general than the latter. In the next section we propose a method to classify the contexts into a hierarchical structure: from general to specific contexts.

4 Context-based File Retrieval

We propose a system that will automatically transfer a user to the most popular context that a given tag participates in. Files that hit the context will be returned. If not satisfied, the user can go down to a more specific context to refine his request or change up to a more general context. Our solution is based on the tag *popularity* and *relationship* between tags to classify contexts into a hierarchical structure: from general to specific contexts. Our proposed algorithm classifies personal tags into a Directed Acyclic Graph according to tag popularity. This Directed Acyclic Graph of Tags (DAGoT) is used to automatically organize files into suitable contexts, to identify the most general context of a tag and allow a user to navigate from a context to another one to retrieve files in an effective manner.

A DAGoT has three node types: tag nodes, leaf nodes and a root node. A *tag node* represents a tag created by a user. It has a label and popularity. A tag node can have many parents and many children. A *leaf node* represents a file tagged by a user. It has a location, such as a URL, from which the file can be accessed. A leaf node has one or more tag nodes as its parent nodes. A *root node* is the beginning of a DAGoT. A root node is the most popular tag node. There are three types of edges: related edges, least popular edges and leaf edges. When two tags are assigned to the same document, we say they are two *related tags*. A *related edge* connects two related tag nodes together. The direction of the edge is from the more popular tag node (*upper tag node*) to the less popular one (*lower tag node*). If two related tag nodes have the same popularity, the one which has the smaller label is the upper. The tag with no parent will take the root node as his parent. A *least popular edge* is a related edge which connects a tag node and its least popular upper nodes. The least popular upper nodes of a tag node are called the parents of the tag node. The parents of a tag node do not have to be related with each other. We support that a tag t has three upper tag nodes which have the corresponding labels "t1", "t2" and "t3" and popularities 3, 2 and 2. In this case, "t2" and "t3" are candidates for the parents of t. If "t2" and "t3" are not related tags, then they are both accepted as the parents of t. If not, only the tag "t3" will be accepted as the parent of t because the label "t3" is greater than the label "t2". When a tag is assigned to a file, a *leaf* *edge* is created from the tag node to leaf node respectively. The figure 1.C represents the DAGoT of the tags in FIG 1.A. The thin, the thick and the dot arrows represent respectively the related, the least popular and the leaf edges. To be concise, we just show files associated with tag JDIR. The DAGoT shows that tag JDIR accepts tag Article as its parent and FinalVersion as its child. JDIR participates in the two contexts {2007, Article, JDIR} and {2007, Article, JDIR, FinalVersion}. The first one is its most popular context which contains its more popular related tags. This is the context that the system will return when a user makes a context-based searching with the given tag JDIR. From JDIR, the user can refine his search query by moving to its child tag FinalVersion. The least popular edges maintain a hierarchical relationship between contexts. They are used as guideline for the user to navigate from one context to another.

(1) Res(t) ←{r∈R | (r,t)∈P}: *The leaf edges from a tag to all its resources*
(2) Tag(r) ←{t∈T | (r,t) ∈P}: *The leaf edges from all associated tags to a resource*
(3) Pop(t) ←card({r∈R | (r,t)∈P}): *The popularity of a tag*
(4) Rel(t1,t2) ←∃r∈R | (r,t1)∈P & (r,t2)∈P: *Check if a related edge exists between tags*
(5) Upper(t') ←{t | Rel(t',t) & Pop(t)>Pop(t')}: *Related edges arrive at tag t'*
(6) Upper(t') ←{t | Rel(t',t) & Pop(t) = Pop(t') & label(t)<label(t')}: *If tags t' and t have the same popularity, the label of t has to be smaller than the label of t'*
(7) Pmop(t) ← min{Pop(p) | p∈Upper(t)}: *The smallest popularity among the upper tags of t*
(8) Parent(t) ←{p | p∈Upper(t)} & Pop(p) = Pmop(t) & !∃p' & p'∈Upper(t) & Rel(p,p') & Pop(p') = Pmop(t) & Label(p') < Label(p)}: *Least popular edges arrive at tag t*
(9) Children(t) ←{ c∈T | t∈Parent(c)}: *Least popular edges starting from tag t*

TAB. 1 – *Formal model for a DAGoT.*

For each user, a tagging system is formally represented as a tuple U:=(R,T,P), where R and T are finite sets that represent the files and tags managed by the user. P represents the postings made by the user. A posting represents the relationship between a resource and a tag, P = RxT. The formal model for a DAGoT is described in TAB 1.

(10) Rsat(t)←{r∈ Res(t) | Tag(r) ⊂ (Upper(t) ∪ {t}) }
(11) Psat(t)←Parent(t)
(12) Csat(t)←Children(t)
(13) Cbfr(t)←[Rsat(t), Psat(t), Csat(t)]

TAB. 2 – *Simple context-based file retrieval.*

In a simple case, context-based file retrieval can be defined as in TAB 2. Given a tag t, its context-based file retrieval Cbfr(t) contains three types of information: a set of files Rsat(t) that hit the most popular context containing t, a list of parent tags Psat(f) guiding to more general contexts, and a list of children tags Csat(f) guiding to more specific contexts. In the above example, we have Cbfr(JDIR) =[{file1.doc, file2.pdf},{Article},{FinalVersion}]. In fact, Rsat does not always return a value for every tag. There are some tags for which Rsat is empty. We call t an *empty tag* if its Rsat(t) is empty and it has only one parent and one child. We propose to pass through an empty tag. The searching result of an empty tag is automatically replaced by the searching result of its unique child. In the above example, Article is an empty tag. Therefore Cbfr(Article) is automatically replaced by Cbfr(JDIR). In addition, the parent role and children role of an empty tag are also replaced by its parent and child. So the context-based file retrieval is redefined as the expressions in TAB 3 below.

(14) $\mathrm{Empty}(t) \leftarrow \mathrm{card}(\mathrm{Rsat}(t))=0 \,\&\, \mathrm{card}(\mathrm{Children}(t))=1 \,\&\, \mathrm{card}(\mathrm{Parent}(t))=1$

(15) $\mathrm{Cbfr}(t) \leftarrow [\mathrm{Rsat}(t), \mathrm{Psat}(t), \mathrm{Csat}(t)] \mid !\mathrm{Empty}(t)$

(16) $\mathrm{Cbfr}(t) \leftarrow \mathrm{Cbfr}(c) \mid \mathrm{Empty}(t) \,\&\, c \in \mathrm{Children}(t)$

(17) $\mathrm{Psat}(t) \leftarrow \{ p \in \mathrm{Parent}(t) \mid !\mathrm{Empty}(p) \}$

(18) $\mathrm{Psat}(t) \leftarrow \mathrm{Psat}(p) \mid p \in \mathrm{Parent}(t) \,\&\, \mathrm{Empty}(p)$

(19) $\mathrm{Csat}(t) \leftarrow \{ c \in \mathrm{Children}(t) \mid !\mathrm{Empty}(c) \}$

(20) $\mathrm{Csat}(t) \leftarrow \mathrm{Csat}(c) \mid c \in \mathrm{Children}(t) \,\&\, \mathrm{Empty}(c)$

TAB. 3 – *Complete context-based file retrieval.*

5 Implementation and testing

First we downloaded posts of 46 random persons from (Delicious) to calculate the number of tags and related resources per person and the number of resources and related tags per tag. Next, we implemented the algorithm for creating DAGoT and used the above tagging data to test the algorithm. We made statistics on the 46 created DAGoTs in order to validate our approach. The TAB 4 compares useful values for file retrieval in two cases. The average values of compared characteristics in context-based model using DAGoT are all smaller than the ones in the Delicious model. In the Delicious model, a user has about 717 tags to choose. Reversely, a user in context-based file retrieval model has only 145 contexts to choose. This proves that the DAGoT model better supports users in personal file retrieval. The smaller ranges of the values of compared characteristics in the context-based model using the DAGoT show that the DAGoT has a balance structure. This prevents from the cases where there are thousands of hit resources or hundreds of related tags with a given tag.

Delicious Model (Tag-based File Retrieval)		**DAGoT Model (Context-based File Retrieval)**	
Tags per user	Average: 717	Contexts per user	Average: 145
	Range: 2-4590		Range: 2-663
Resources per tag	Average: 4.6	Resources per context	Average: 2.2
	Range: 1-1426		Range: 1-96
Related tags per tag	Average: 16.5	Parents per tag	Average: 1.1
	Range: 1-3715		Range: 1-23
		Children per tag	Average: 2.7
			Range: 1-113

TAB 4. – *Comparison between the Delicious model and the DAGoT model.*

6 Conclusion and Future Works

We have proposed to enhance personal file retrieval with context-based searching. We support that each user has an own personal vocabulary of tags that are semantically grouped into different contexts. The set of tags associated to a file by a user creates a context. We proposed an algorithm creating automatically a DAGoT based on tag popularity and relationships of tag. This DAGoT is used to identify automatically the hit context for a given tag. Using DAGoT, users can navigate from one context to others to retrieve his files in an efficient manner. For the future, we will integrate this tagging system into Ontology-based file system (Ngo et al., 2007) and propose a complete method for file retrieval in which we take into account both extrinsic file semantics and intrinsic file semantics.

References

Apple Computer (2005). *Inc: Tiger Developer Overview Series - Working with Spotlight.* http://developer.apple.com/macosx/spotlight.html.

Barreau, D., and B. Nardi, (1995). *Finding and reminding: file organization from the desktop*. SIGCHI Bulletin, 27(3), 39-43.

Bloehdorn, S., O. Görlitz, S. Schenk, and M. Völkel, (2006). *TagFS --- Tag Semantics for Hierarchical File Systems*. Proceedings of the 6th International Conference on Knowledge Management (I-KNOW 06), Graz, Austria, September 2006.

Delicious. http://del.icio.us/

Khoo, C., B. Luyt, C. Ee, J. Osman, H.H. Lim, and S. Yong (2007). *How users organize electronic files on their workstations in the office environment: a preliminary study of personal information organization behaviour*. Information Research, 12(2), paper 293.

Ngo, H.B., C. Bac, and F. Silber-Chaussumier (2007). *Toward ontology based semantic file systems*. Proceedings of the 5th International Conference on Research, Innovation & Vision for the Future, Hanoi, Vietnam.

Padioleau, Y. (2005). *Logic File System, un système de fichier basé sur la logique*. Thèse de doctorat, Université de Rennes 1.

Résumé

Depuis peu, les étiquettes sont utilisées largement pour identifier des contenus aussi bien sur le bureau informatique des utilisateurs que sur les sites coopératifs du Web dit 2.0. Notre recherche se focalise sur l'organisation assistée des étiquettes personnelles afin d'améliorer la pertinence des recherches de fichiers personnels associés à des étiquettes. Notre proposition utilise la notion de contexte comme point central. Un contexte est constitué à partir d'un ensemble d'étiquettes affectées par un utilisateur à un fichier. Nous proposons une infrastructure qui permet à un utilisateur de naviguer à travers les contextes pour retrouver ses fichiers.