

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

Hung Ba Ngo^{1,2}, Frédérique Silber-Chaussumier¹, Christain Bac¹

Institut National des Télécommunications-France¹, Cantho University-Vietnam²
{hung.ngo_ba, frederique.silber-chaussumier, christian.bac}@int-evry.fr

Abstract. Recently, tags are widely used to tag various content on the Internet and collaborative systems. On the desktop, tags are also supported by some semantic file systems and desktop search tools. Tags are very relevant in personal file retrieval because the users are flexible in describing their opinions and interest on files. As the number of tags and resources managed by a user increase in numbers, the user gets difficulty in retrieving files. While researches in the domain mostly are related to social characteristics of tagging systems to explore resources, we focus on personal tag organization to enhance personal file retrieval. Our proposal is based on the notion of *context*. A context is created from the set of tags assigned to a file by a user. A user can navigate in the hierarchy of contexts to retrieve files in an effective manner. In this paper, we propose an algorithm to compute context hierarchy and a prototype implementation.

1 Introduction

Files in semantic file systems (SFS) are automatically organized according to their semantics and can be retrieved by introducing semantic expressions which describe retrieval criteria. While intrinsic file semantics, such as the artist, the album title and the year of a mp3 file, is automatically collected, extrinsic file semantics, such as user's opinion on a music file or names of persons in a photo, is manually assigned to files by users. The web community uses the notion of *tag* for this information. Systems that support tags are called tagging systems. ([Flickr](#)) or ([Del.icio.us](#)) are the two well-known tagging systems on the web. We reuse the tag notion for extrinsic file semantics. Some SFS, such as LFS (Padioleau, 2005), TagFS (Bloehdorn et al., 2006), or desktop search tools, such as Spotlight (Apple Computer, 2005), also support tags. These systems let users assign tags or keywords to files for later retrieving. Tags are very relevant in personal file retrieval because the users are flexible in describing their opinions and interests on files. Each user has a *personal vocabulary* of tags easier to use than an external existing classification. The main issue of tagging systems is to control the meanings or semantics of tags. Marchetti et al. (Marchetti et al., 2007) focus on how to map a tag to a concept. Several other issues are identified as: how to recognize that two different tags in two personal vocabularies of tags represent the same concept or reversely, how to recognize that the same tag in two personal vocabularies of tags represent two different concepts. Many researches concentrate on matching user-assigned

Titre court de votre article en 10 mots maximum

tags with existing classifications (Abbasi et al., 2007). This difficult problem still does not have a satisfying solution. Other researches are related to social characteristics of tagging systems to explore resources. In this paper we focus on how to organize personal tags to enhance personal file retrieval. Our approach is based the notion of *context*. A context is a set of tags assigned to a file (or a resource in general) by a user and is calculated from the tag popularity and relationship between tags. The proposed algorithm to compute a context makes it possible for a user to navigate in the hierarchy of contexts to retrieve his files in an effective manner. In this paper, we first present tagging systems that are widely used and interesting techniques for tag organization in section 2. In section 3, we introduce tag-based context and how to use it to enhance tagging systems from tag-based searching to context-based file retrieval. We introduce our algorithm for automatically creating a Directed Acyclic Graph of Tags (DAGoT) based on tag popularity and relationship in section 4. This DAGoT is used to organize tag-based contexts in a hierarchical manner. We show how to use the DAGoT to enhance personal file retrieval in tagging system with context-based file retrieval in section 5. An implementation and experimental results using real data are presented in section 6. Our conclusion and perspectives are in the last section.

2 Related Works

In this section, we present widely used tagging systems and interesting techniques for tag organization.

[Del.icio.us](http://del.icio.us) [2] is an online bookmark server where users can use their own tags to organize and retrieve their bookmarks. The benefit of this is that bookmarks can be accessed from any computer Two or more tags associated to the same bookmark are considered as *related*. For example, if `file1` is assigned with tags `t1` and `t2`, then the tag `t1` is related to the tag `t2` and reversely. The number of bookmarks associated to a tag by a user is called the *popularity* or *frequency* of the tag. The user's homepage located at <http://del.icio.us/username> displays a list of all tags used by the user . Tags can be sorted according to alphabet or popularity. When a tag is chosen from the tag list, a list of bookmarks associated with this tag and a list of related tags are returned as a filter result. Related tags are the way to navigate between tags for finding interesting bookmarks. However, when the number of bookmarks and tags increase in numbers, parsing the filter result for a bookmark or the tag list and related tags becomes a tremendous task for users.

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. Spotlight does not take into account any relationship between tags, even through the related tags.

In the file system domain, LFS (Padioleau, 2005) lets users associate files with tags that represent file properties. For example, file `file1.pdf` can be assigned by his owner with tags such as `Paper`, `Conference`, `RIVF`, `2007`, and `Hanoi` to describe that `file1.pdf` is a paper for the conference on Research, Innovation & Vision for the Future (RIVF) at Hanoi of Vietnam, in 2007. LFS supports three interesting characteristics. First, a file query in LFS is a logic expression of tags with AND, OR and NOT operators. Second, LFS supports axioms between tags as a parent-child relationship. Users can manually create

axioms between their tags. When the tag `Hanoi` is defined as a child of `Vietnam`, a file that is assigned with `Hanoi` is also deduced as being assigned with `Vietnam`. At last, from the axioms, a taxonomy of tags is created. Users can browse the taxonomy for file retrieval as they do with traditional directories. In addition, the taxonomy is also used for classifying search results so that users can browse the result to refine their request.

The tag file system, TagFS (Bloehdorn et al., 2006), presents a mapping of tag-based systems to the traditional file system paradigm by assigning new semantics to traditional file system operations such as `open`, `create`, `delete`, ... While preserving the notions of directories and files, and providing all standard file system operations, the semantics of these operations are changed to take into account tag annotations. Legacy applications have still access to file systems with traditional file operation interfaces. In fact, file system operations are mapped to metadata modification. On the other hand, TagFS provides new tagging-based file retrieval operations for new semantic applications. Files can be retrieved by a conjunction of tags. The retrieval result contains not only hit files but also a list of related tags such as Del.icio.us does. Consequently TagFS has the same difficulty when the number of files and tags increases. Looking for a file in the retrieval result or a tag in the related tags becomes a tremendous task for users.

3 Definition of the tag-based context

In order to improve personal file retrieving when the number of files and tags increases, we define the *tag-based context* of a file.

When creating or downloading a new file, the user assigns the file with a set of tags or keywords using his own vocabulary. Each tag represents a concept or an object related to the file. For example, a user may assign the set of tags `{Vacation, Paris, 2007}` to the file `myphoto.jpg` to recall that the photo was taken during vacation of summer '07 in Paris. We call a set of tags that is assigned to a resource by a user is a *tag-based context* (or a context for short). The meaning of a context is aggregated from its element tags. A context is more meaning than a tag. So it is more interesting to return in file retrieval result a set of files hit a context than a set of files hit a tag.

	Article	JDIR	2007	Final Versio n	RIVF	Vacation	Paris	Hanoi
file1.doc	x	x	x					
file2.pdf	x	x	x					
file3.doc	x	x	x	x				
file4.pdf	x	x	x	x				
file5.doc	x		x		x			

Titre court de votre article en 10 mots maximum

file6.pdf			x	x	x			
file7.jpg			x			x	x	
file8.jpg			x			x		x

TAB. 1 – Example of postings made by a user.

Table 1 is an example of postings made by a user. The rows are the files managed by the user and the columns are the tags created by him. The checked cells show that the files are assigned with the corresponding tags. If the user makes a tag-based searching with the tag `Article`, five files (from `file1` to `file5`) will be returned as the result. These files belong to three contexts as described in Table 2.

No	Contexts	Files belonged to context
1	{ <code>Article, JDIR, 2007</code> }	file1.doc, file2.pdf
2	{ <code>Article, JDIR, 2007, FinalVersion</code> }	file3.doc, file4.pdf
3	{ <code>Article, RIVF, 2007</code> }	file5.pdf

TAB. 2 – Example of tag-based contexts

In order to improve file retrieval, we want to enhance tagging systems with context-based searching instead of tag-based searching. A tag usually participates in many contexts. The table 2 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. The system should have a strategy to choose the most relevant context among the contexts that a tag participates in. In the next section we propose a way to classify the contexts into a hierarchical structure : from general to specific contexts.

4 Context-based File Retrieval

We want to build a system that will automatically transfer the user to the most popular context that the 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 go up to a more general context.

Our solution is based on the tag *popularity* and *relationship* between tags to classify the contexts into a hierarchical structure: from general to specific contexts. The *popularity* of a tag is the number of resources assigned to the tag by a user.

We propose an algorithm that classifies personal tags into a Direct 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 one context to another context to retrieve files in a effective manner. The algorithm for creating a DAGoT is presented in the following sections.

4.1 Construction of the Directed Acyclic Graph of Tags (DAGoT)

We propose to build a directed acyclic graph of tags (DAGoT) in order to compute context hierarchy. 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 retrieved. A leaf node has one or more tag nodes as its parent nodes.

A *root node* is the beginning of a DAGoT. It has no parent and many tag nodes as its children. A root node is the most popular tag node.

There are three types of edge: 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 parent. 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 with its least popular upper nodes. The least popular upper nodes of a tag node are called the parent of the tag node. The parent of a tag node have not to be related each other. We support that tag t has three upper tags node t_1 , t_2 and t_3 with the popularities respectively are 3, 2 and 2. In this case, t_2 and t_3 are the candidates for the parent of t . If t_2 and t_3 are not related tags, they are both accepted as the parent of t . In case where t_2 and t_3 are related tags, only the tag t_3 will be accepted as the parent of t because the label “ t_3 ” is greater than the label “ t_2 ”.

When a tag is assigned to a file, a *leaf edge* is created from the tag node to leaf node respectively.

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

TAB. 3 – Popularities of tags

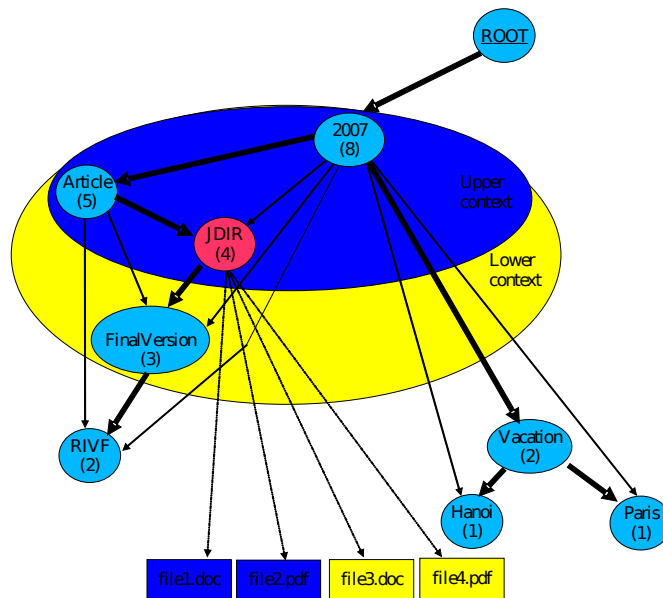


FIG. 1 – An example of a DAGoT

Table 3 represents the popularities of tags in Table 1. Figure 1 presents the DAGoT of the tags in table 1. The thin, the thick and the dot arrows represent respectively the related, the least popular and the leaf edges. Because of the space restriction, we just show the files associated with the tag JDIR. The DAGoT shows that the tag JDIR accepts 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 automatically 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 – the tag FinalVersion. The least popular edges maintain a hierarchical relationship between contexts. They are used as a guideline for the user to navigate from one context to another.

4.2 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 tags, $P = R \times T$. The formal model for a DAGoT is described in the Table 4.

$Res(t) \leftarrow \{r \in R \mid (r,t) \in P\}$	(1) The leaf edges from a tag t to all its resources
$Tag(r) \leftarrow \{t \in T \mid (r,t) \in P\}$	(2) The leaf edges from all associated tags to a resource
$Pop(t) \leftarrow \text{card}(\{r \in R \mid (r,t) \in P\})$	(3) The popularity of a tag
$Rel(t_1,t_2) \leftarrow \exists r \in R \mid (r,t_1) \in P \ \& \ (r,t_2) \in P$	(4) Check if a related edge exists between two tags t_1 and t_2 .
$Upper(t) \leftarrow \{t' \mid Rel(t,t') \ \& \ Pop(t') > Pop(t)\}$	(4) Related edges arrive at tag t
$Upper(t) \leftarrow \{t' \mid Rel(t,t') \ \& \ Pop(t') = Pop(t) \ \& \ \text{label}(t') < \text{label}(t)\}$	(5) Related edges arrive at tag t . In the case where tags t' and t have the same popularity, the label of t' has to be smaller than the label of t .
$Pmop(t) \leftarrow \min\{Pop(p) \mid p \in Upper(t)\}$	(6) The smallest popularity among the upper tags of t .
$Parent(t) \leftarrow \{p \mid p \in Upper(t) \ \& \ Pop(p) = Pmop(t) \ \& \ \exists p' \ \& \ p' \in Upper(t) \ \& \ Rel(p,p') \ \& \ Pop(p') = Pmop(t) \ \& \ \text{Label}(p') < \text{Label}(p)\}$	(7) Least popular edges arrive at tag t .
$Children(t) \leftarrow \{c \in T \mid t \in Parent(c)\}$	(8) Least popular edges starting from tag t .

TAB. 4 – Formal model for a DAGoT

5 Using DAGoT for context-based file retrieval

The previous section shows how a hierarchy of tags can be automatically created using our algorithm. This section introduces how to use this DAGoT for context-based file retrieval.

$$Res(t) \leftarrow R_{sat}(t) \cup R_{ref}(t) \quad (9)$$

$$R_{sat}(t) \leftarrow \{r \in Res(t) \mid Tag(r) \subset (Upper(t) \cup \{t\})\} \quad (10)$$

$$R_{ref}(t) \leftarrow Res(t) - R_{sat}(t) \quad (11)$$

TAB. 5 – Classifying the hit files in tag-based searching in to satisfied and refined groups

Titre court de votre article en 10 mots maximum

With a given tag, tag-based search simply returns a list of files (or resources) associated with the tag. While Del.icio.us [2] and TagFS (Bloehdorn et al., 2006) extend the list of files with a list of related tags, LFS (Padioleau, 2005) uses axioms to classify hit files into children tags where users can navigate to refine their request. In our solution, the hierarchy of tags can be used to identify automatically files that hit the most popular context and guide user from one context to another.

A user begins file retrieval by giving a tag, for example JDIR. The system can simply return all files associated with the tag JDIR such as `file1`, `file2`, `file3`, and `file4`; see figure 5. However this is not interesting as the number of hit files increases. If t is the current tag, $Res(t)$ (1) is the set of hit files. We propose to classify $Res(t)$ into two groups. A group, called $R_{sat}(t)$, contains files associated with the given tag and some or all of its upper tags. The other group, called $R_{ref}(t)$, contains files that are associated with the given tag and has at least one tag that is not the upper tag of the given tag. Then they can be defined as figure 6.

$$Cbfr(t) \leftarrow [R_{sat}(t), P_{sat}(t), C_{sat}(t)] \quad (12)$$

$$P_{sat}(t) \leftarrow Parent(t) \quad (13)$$

$$C_{sat}(t) \leftarrow Children(t) \quad (14)$$

TAB. 6 – Simple context-based file retrieval

In a simple case, the context-base file retrieval can be defined as Figure 7. Given a tag t , the user will receive three types of information: a set of files $R_{sat}(t)$ that hit the most popular context of t , a list of parent tags guiding to more general contexts, and a list of children tags $C_{sat}(f)$ guiding to more specific contexts. In the above example, we have

$$Cbfr(JDIR) = \{file1.doc, file2.pdf\}, \{Article\}, \{FinalVersion\}$$

In fact, R_{sat} does not return a value for every tag. $R_{sat}(JDIR)$ is not empty, because JDIR is the least popular tag among the tags belonging to the most popular context {2007, Article, JDIR}. Otherwise, $R_{sat}(2007)$ and $R_{sat}(Article)$ are empty. If R_{sat} of a tag is not empty, then the tags together with its upper tags represent a most popular context of the tag. The objective of context-based file retrieval is to return the files from a context from a given tag. However, when giving the tag 2007, no file is returned. We accept this, because the tag 2007 participates in many contexts and have many children. Its children will guide the user to a more specific context {2007, Article, JDIR} or {2007, Vacation, Paris}, or {2007, Vacation, Hanoi}. In the case where a tag has only one child, such as the tag Article, the system will automatically navigate to its child tag, the tag JDIR. So the context-based file retrieval is redefined as the expressions (15)-(17).

$$Cbfr(t) \square \leftarrow [R_{sat}(t), P_{sat}(t), C_{sat}(t)] \mid card(R_{sat}(t)) > 0 \quad (15)$$

$$Cbfr(t) \square \leftarrow [R_{sat}(t), P_{sat}(t), C_{sat}(t)] \mid card(R_{sat}(t)) = 0 \ \& \ card(Children(t)) > 1 \quad (16)$$

$$Cbfr(t) \square \square \leftarrow Cbfr(c) \mid card(R_{sat}(t)) = 0 \ \& \ card(Children(t)) = 1 \ \& \ c \in Children(t) \quad (17)$$

$$\text{Psat}(t) \leftarrow \{ p \in \text{Parent}(t) \mid \text{card}(\text{Rsats}(p)) > 0 \} \quad (18)$$

$$\text{Csats}(t) \leftarrow \{ c \in \text{Children}(t) \mid \text{card}(\text{Rsats}(c)) > 0 \} \quad (19)$$

$$\text{Csats}(t) \leftarrow \{ c \in \text{Children}(t) \mid \text{card}(\text{Rsats}(c)) = 0 \ \& \ \text{card}(\text{Children}(c)) > 1 \} \quad (20)$$

$$\text{Csats}(t) \leftarrow \text{Csats}(c) \mid c \in \text{Children}(t) \ \& \ \text{card}(\text{Rsats}(c)) = 0 \ \& \ \text{card}(\text{Children}(c)) = 1 \} \quad (21)$$

$$\text{Psats}(t) \leftarrow \{ p \in \text{Parent}(t) \mid \text{card}(\text{Rsats}(p)) = 0 \ \& \ \text{card}(\text{Children}(p)) > 1 \} \quad (22)$$

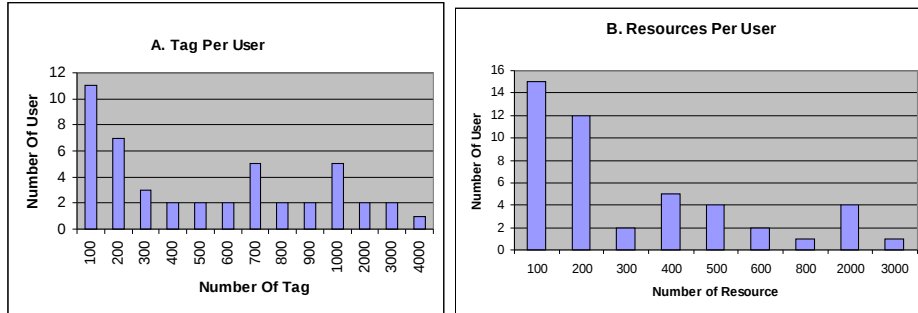
$$\text{Psats}(t) \leftarrow \text{Psats}(p) \mid p \in \text{Parent}(t) \ \& \ \text{card}(\text{Rsats}(p)) = 0 \ \& \ \text{card}(\text{Children}(p)) = 1 \} \quad (23)$$

TAB. 6 – – Complete context-based file retrieval

In our context-based file retrieval, **Csats** and **Psats** represent the paths that allow a user to navigate from one context to another. **Csats** contains a set of children tags that guide a user to a more specific context. For example, **Csats(2007)** contains two children tags **Article** and **Vacation**. While **Article** guide the user to the context $\{2007, \text{Article}, \text{JDIR}\}$, **Vacation** guide the user to the context $\{2007, \text{Vacation}, \text{Paris}\}$ or $\{2007, \text{Vacation}, \text{Paris}\}$. On the other hand, **Psats** contains the parent tags that guide a user to a more general context. In addition, the tags which **Rsats** are empty and that have only one child and one parent, such as the tag **Article**, should be passed through when it is considered as a parent or a child of other tag. Its parent role (or child role) should be replaced by its unique parent (or children). For the tag **JDIR**, instead of taking **Article** as its parent, **JDIR** will take the tag **2007** as its parent. In the same way, the tag **2007** will take the tag **JDIR** instead of the tag **Article** as one of its children tags. So we redefine **Psats** and **Csats** as the expression (18)-(23).

6 Implementation and testing.

To test our proposal, we decided to use a panel of pages from Delicious. This panel was made with the public data of 46 persons. The pages were downloaded, then we made some statistics on the data to have a vision about personal tag.



Titre court de votre article en 10 mots maximum

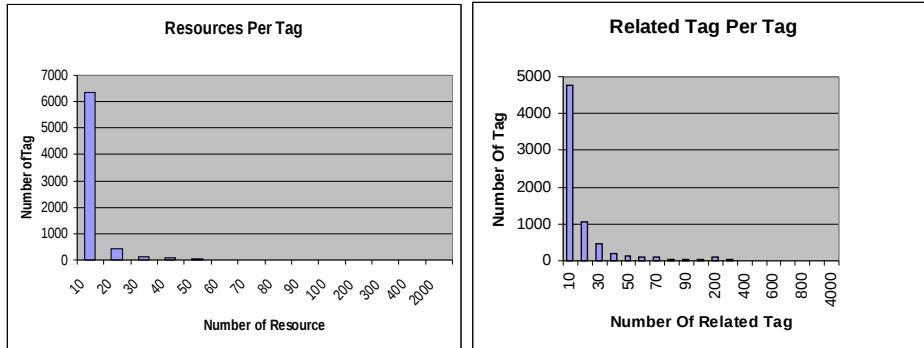


FIG 2. - *Statistic of personal tagging data downloaded from Delicious*

The chart A in FIG 2 presents statistics on the number of tag created by a person. Each column represents an interval of the tag number and the number of persons whose tag number are in that interval. The first column shows that there are 11 persons who own from 1 to 100 tags. The range of tag number owned by a person is very large: from 2 tags to 4590 tags. The chart B shows a statistic on the number of resource tagged by a person. Most of the person has from 100 to 200 resources. There are five persons that have more than 2000 resources. The chart C displays the number of resources assigned to a tag. Most tags have about 10 resources. The chart D shows that most of the tags have from 10 to 20 related tags. There are 171 tags having more than 100 related tags. Especially, there are 3 tags that have more than 2000 related tags.

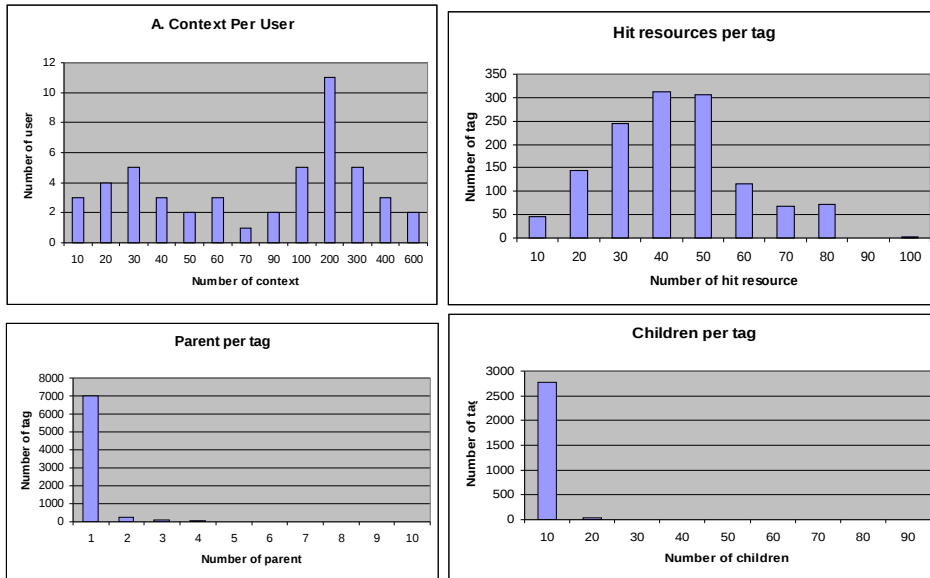


FIG 2. - *Statistics of DAGoTs created from tagging data downloaded from Delicious*

We then used the above algorithm for creating DAGoT and created 46 DAGoT using the above data. We make some statistic on the created DAGoT to have a view on the general DAGoT.

<TO BE COMPLETED SOON! >

7 Conclusion and Future Works

We have proposed to enhance personal file retrieval with context-based file retrieval. We support that each user has each own personal vocabulary of tags that they are semantically grouped into different contexts. The set of tags associated to the same file creates a context. A context can contain many files. A tag may participate in many contexts. For a user, the meaning or semantics given by a context is more complete than the one given by each member tag. So instead of providing tag-based file retrieval, our system provides context-based file retrieval. Hit files in searching result are more relevant because they hit not only a given tag but also its contexts. We proposed an algorithm for creating automatically a Directed Acyclic Graph of Tags (DAGoT) based on tag popularity and relationships of tag. This DAGoT is used to identify automatically the hit context for a given tag. User can browse the DAGoT to change from one searching context to others to retrieve his files in an efficient manner.

For the future works, we continue on optimizing the running time of the algorithm. We implement the algorithm in Java, use MySQL as database and JPOX1.2 as Persistence Data Object. The cost for updating DAGoT as a new post arriving is in average from 200 ms to 800 ms. There is some case, for the tag with more than 1000 related tags, it augments to 10.000 ms. 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

Abbasi R.,S. Staab, and P. Cimiano (2007). *Organizing Resources on Tagging Systems using T-ORG*. Bridging the Gap between Semantic Web and Web 2.0, workshop at ESWC. <http://www.uni-koblenz.de/~abbasi/publications/Abbasi2007ORO.pdf>.

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

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/>

Flickr. <http://www.flickr.com/>

Titre court de votre article en 10 mots maximum

Marchetti, A., M. Tesconi, F. Ronzano, M. Rosella and S. Minutoli (2007). *SemKey: A Semantic Collaborative Tagging System*. WWW2007, May 8–12, 2007, Banff, Canada.

Ngo, H.B, C. Bac, and F. Silber-Chaussumier (2007). *Toward ontology based semantic file systems*. Proceeding 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*. These de doctorat, Université de Rennes 1.