Master's Dissertation

FolkSpace – 시멘틱 시스템을 위한 미들웨어
FolkSpace – A Middleware for Emergent Semantic Systems

Enkhbold Nyamsuren
Department of Information and Communications Engineering

KAIST

2009

FolkSpace – 시멘틱 시스템을 위한 미들웨어
FolkSpace – A Middleware for Emergent Semantic Systems

FolkSpace – A Middleware for Emergent Semantic Systems

Advisor: Professor Ho-Jin Choi

by

Enkhbold Nyamsuren
Department of Information and Communications Engineering
KAIST

A thesis submitted to the faculty of the KAIST in partial fulfillment of the requirements for the degree of Master of Information and Communications Engineering in the Department of Information and Communications Engineering.

Daejeon, Korea
June 15th , 2009
Approved of by

_____
Professor Ho-Jin Choi

FolkSpace – A Middleware for Emergent Semantic Systems

Enkhbold Nyamsuren

The present dissertation has been approved of
by the dissertation committee
as a master's dissertation
at the KAIST

June 15th, 2009

Committee head Ho-Jin Choi

Committee member In-Young Ko

Committee member Gwan-Su Yi

MICE Enkhbold Nyamsuren

20074308

FolkSpace – A Middleware for Emergent Semantic Systems.

FolkSpace – 시멘틱 시스템을 위한 미들웨어

Department of Information and Communications Engineering. 2009. 63 pages.

Advisor Prof. Ho-Jin Choi

Abstract

Although relatively new technology, folksonomies are proven to be useful. Folksonomy is a cheap and effective method for organizing and managing the content and resources. In some systems, like Delicious and Flickr, the folksonomy became the backbone around which all the services are delivered. The number of folksonomy-based applications is increasing rapidly.

Following this trend there is a vision of environment where human users and more importantly software agents can query, search and retrieve resources from disparate end-points in an easy and meaningful way. The environment with network of self-organizing communities generating, annotating and sharing the content dynamically, and it is believed that folksonomy-based systems are key technology for bringing this vision into reality. But the there is still no clear understanding of how to leverage from annotations produced by folksonomies in an efficient and productive way. Complicating this matter, the interoperability between folksonomy-based applications is still a major issue. At the same time folksonomy lack of structure which also limits its usefulness.

FolkSpace is a middleware or platform which is designed to solve the problems described above. FolkSpace not only provides unified access to folksonomies from different systems, but also tries to add much needed semantic structure to folksonomy. FolkSpace follows the modern standards of interoperability in Semantic Web by heavily

relying on platform independent formats such as XML. From other side FolkSpace provides easy access to folksonomies though standardized query language SPARQL.

Table Contents

Abstract i

Table Contents iv

List of Tables vi

List of Figures vii

List of Abbreviations viii

Chaper 1. Introduction 1
1.1 Motivation 1
1.2 Research Objectives 1
1.2.1. Ontology for formal representation and semantic of tagging 1
1.2.2. Support of software agents 1
1.2.3. Refined query, search and navigation 2
1.3 Success Criteria 3
1.4 Summary of Contribution 3

Chaper 2. Background 5
2.1 Folksonomy 5
2.2 Types of Folksonomy 6
2.3 Characteristic of Folksonomy 7
2.4 Semantic Value of Folksonomy 9
2.5 Folksonomy-based Systems 10

Chaper 3. Related Works 11
3.1 Ontology for Tagging 11
3.2 Tag filtering 12
3.3 Deriving semantics from folksonomies 12
3.4 Semantic Middleware 13

Chaper 4. FolkSpace – A Semantic Middleware for Folksonomy-based Applications 14
4.1 Definition of FolkSpace 14
4.2 Semantic power of FolkSpace 15
4.3 Overall Architecture 15
4.4 Semantic Model of Folksonomy 17
4.5 Tag Model 20
4.6 FOM Ontology 23
4.6.1 FOM Ontology schema for semantic model 27
4.6.2 Representing types of folksonomy in FOM Ontology 28
4.6.3 FOM Ontology schema for Tag Model 31
4.7 FOM Rules 33
4.8 FOM Processor 35
4.8.1 Creating semantic model 36
4.8.2 Updating the Tag Model 37

4.9 Folksonomy-to-Ontology Maturing Process 37
4.9.1 Tag filtering phase 38
4.9.2 Lexicographic analysis phase 40
4.9.3 Statistical analysis phase 41
4.9.4 Cosine Similarity based Double Clustering Algorithm 42
4.9.5 Probabilistic analysis method 43
4.9.6 Updating Tag Model 44

Chaper 5. Prototype Implementation 45
5.1 Prototype 45
5.2 Folksonomy datasets 45
5.3 Prototype Architecture 45
5.4 Sample results from semantic processing 46
5.5 User Interface for Querying 47

Chaper 6. Experimentation and Evaluation 49
6.1 Evaluation of FOM ontology 49
6.2 Evaluation of linguistic capabilities of the system 50
6.3 The taxonomical reasoning capability of the system 50
6.4 Overall Evaluation 52

Chaper 7. Conclusion 53

Summary 55

References 56

Acknowledgement 60

Curriculum Vitae 61

List of Tables

Table 1. Properties of the example folksonomy. 17
Table 2. Corresponding semantic model of folksonomy defined in Table 1. 18
Table 3. Example folksonomies and corresponding tag spaces. 21
Table 4. Descriptions contained within Tag Model given folksonomies from Table 3. 22
Table 5. FOM Ontology classed used in semantic model 26
Table 6. Set of classes used for defining the structure of Tag Model 32
Table 7. The rule defining the propagation of instance of *Tag* ownership from instance of *UserAnnotation* to *ResourceAnnotation* 34
Table 8. The rule defining the equivalence of two instances of Tag when they morphologically similar or synonymous 35
Table 9. Types of noisy tags together with examples 38
Table 10. Examples semantic extracted from tag collection 47
Table 11. Test data properties 49
Table 12. Comparison of reasoner enhanced and ordinary searches 51

List of Figures

Figure 1. The graphical representation of tripartite model of folksonomy. 5
Figure 2. Knowledge representation spectrum. 7
Figure 3. Ontological structure proposed for modeling the tagging. 11
Figure 4. The overall architecture of FolkSpace 17
Figure 5. The descriptions of tags from folksonomies are stored in a single model called Tag Model. 20
Figure 6. Tags from different folksonomies are collected in Tag Model and then processed for implicit semantics, revealed semantics are stored back in Tag Model 21
Figure 7. The structure of FOM Ontology; the part of ontology within dashed border defines the structure of semantic model, and the part highlighted with dotted line defines the structure of Tag Model. 25
Figure 8. The Venn diagram of tag space in folksonomy. 27
Figure 9. Difference between broad, narrow and personal folksonomies in terms of structures of their tag spaces 29
Figure 10. Modeling three types of folksonomies using *Folksonomy*, *ResourceAnnotation* and *UserAnnotation* classes in FOM Ontology. 30
Figure 11. The example scenario where rule is required to define relation between instance of *Tag* and *ResourcesAnnotation.* 33
Figure 12. Example of synonymous and morphological relationships among tags. 34
Figure 13. Inputs and outputs of FOM Processor. 36
Figure 14. Steps for creating the semantic model inside the FOM Processor. 37
Figure 15. Steps in tag filtering phase. 39
Figure 16. Process of identifying morphological groups. 40
Figure 17. Hierarchy analysis through combination of clustering and probabilistic analysis. 42
Figure 18. Basic steps in algorithm for deriving tag clusters 43
Figure 19. The prototype architecture. 46
Figure 20. User Interface for sending queries to prototype. 48

List of Abbreviations

KAIST	Korean Advanced Institute of Science and Technology
FOM	Folksonomy-to-Ontology Maturing
OWL	OWL Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SWRL	Semantic Web Rule Language
N3	Notation 3
XML	eXtensible Markup Language

Chaper 1. Introduction

1.1 Motivation

There is a vision of environment where human users and more importantly software agents can query, search and retrieve resources from disparate end-points in an easy and meaningful way. The environment with network of self-organizing communities generating, annotating and sharing the content dynamically, and it is believed that folksonomy-based systems are key technology for bringing this vision into reality. But the there is still no clear understanding of how to leverage from annotations produced by folksonomies in an efficient and productive way [13].

1.2 Research Objectives

1.2.1. Ontology for formal representation and semantic of tagging

One of the major issues today is the heterogeneity of formats for publishing the tagging data, since there is no semantic and formal agreement on the representation of tagging. Although most of the current services use RSS to publish tagging data, it is not suitable for this purpose. First of all there is a compatibility issue due to different version of RSS (RSS 1.0, 2.0, 0.91) that are not backward compatible. Second, since there is no standard RSS syntax for publishing tagging data, every service uses its own syntax to represent tagging within RSS which again results in compatibility issue. Therefore there is lack of accepted common agreement in formal definition of tagging and its semantics. This results in difficulties in interoperability and automated processing. Therefore the first objective is defined as

- mitigating the problem of heterogeneous representation of folksonomies in various systems by developing an ontology for describing the formal representation and the semantics of the tagging

1.2.2. Support of software agents

Another issue of interoperability arises from the unique characteristic folksonomy which is the lack of structure. Folksonomy itself is just collection of tags, and sharing simple collection of tags is as useless as sharing set of keywords. What makes the

folksonomy useful are relations among the tags which are expressed implicitly and derived based on semantic meaning of the tags and pattern of co-occurrence on basis of the same content and the same users. Because folksonomy was originally developed for human users, it is easy for them to understand the implicit semantics (relations between tags) within the folksonomy. In case of automated processing such kind of semantics should be expressed explicitly in a machine readable format. Only with a clear semantic structure the annotations in folksonomies can be useful not just to humans, but can be made available to software agents and applications on the semantic web **[9]**. Therefore extracting implicit semantics out of folksonomy and expressing it in machine readable way is another crucial requirement for achieving the successful interoperability. Also there is again the requirement that extracted semantics should be expressed in a format that is understandable by all systems. Therefore the following objectives should be identified:

- defining a common process of extracting implicit semantics from folksonomies
- defining a ontology-based common standard of representing extracted semantics in a machine readable way

1.2.3. Refined query, search and navigation

Due to lack of explicit and visible structure the folksonomy has serious navigation problems and reduced search capabilities. The searching result is restricted to the specific tags used in the annotation process, and search capability is reduced because of the linguistic and semantic limitations of tags. However, knowledge structured in ontologies can be processed in a more efficient way allowing more elaborated conclusions due to the use of reasoners. The formal definitions mainly serve for machine-machine interoperability purposes, such as semantic web services. Also user-machine interaction (e.g., searches) will be improved by applying taxonomic reasoning mechanisms. Therefore the following objective is defined:

- improve the search and navigation through folksonomy by introducing reasoning and structured query capability based on taxonomical relations

1.3 Success Criteria

The thesis goal will be considered to be accomplished if the success criteria defined in this section are met. Since this work can be considered unique and as far as we know there is no existing project to which results of this can be fairly compared, the success of this thesis will be evaluated by the results of the prototype implementation.

The first success criterion is to show the feasibility of proposed system architecture by implementing the working prototype of the proposed system with following characteristics and functionalities:

- implements the basic architecture of the system
- integrates folksonomies from two or more systems and converts them into common representation format
- provides query based access to those folksonomies using common query language

The second success criterion is to show the feasibility of the proposed process of extracting impicit semantics from folksonomy. This success criterion will be considered to be met if:

- the proposed process will be able to extract at least taxonomical relations from folksonomy
- process is implemented as a part of prototype system
- the extracted semantics can be queried in same manner as an original folksonomy

1.4 Summary of Contribution

The overall work done in this thesis contributes to the line of emerging studies for combining technologies from Web 2.0 and Semantic Web. From more technical side, this work proposes a FolkSpace a semantic middleware with architecture for supporting interoperability between folksonomy-based applications thereby promoting exchange of knowledge over the Internet. During the development of this architecture several new techniques and processes were developed including:

- FOM Ontology which provides the common description format for folksonomy structure

- Folksonomy-to-Ontology Maturing Process which describes how the implicit semantics can be extracted from folksonomy and stored in format easily understood by machine
- Hierarchy analysis algorithm based on combination of clustering and probabilistic approach
- Cosine-Similarity based Double Tag Clustering Algorithm which was developed as a part of Folksonomy-to-Ontology Maturing Process.

In addition to above contributions, the current research has showed the feasibility of the emergent semantics as an bottom-up approach for annotating and managing the content on semantic level. The current research showed that despite absence of explicit and visible structure, the folksonomy contains rich implicit knowledge that can be leveraged in combination with NLP and semantic web techniques. On working prototype this research has shown that such implicit knowledge can be used to improve folksonomy as a tool for better search and retrieval of annotate resources across disparate services.

Chaper 2. Background

2.1 Folksonomy

There are number of knowledge representation approaches which include taxonomies, thesaurus, conceptual models and logical theory, but the most recent approach is so-called Folksonomy. The term Folksonomy was originally coined by Thomas Wander Val [33] and is formed by combination of words “folk” and “taxonomy”. Wikipedia defines Folksonomy as a practice and a method of collaboratively creating and managing tags to annotate and classify the content or resource (further the terms “content” and “resource” are used interchangeably).

The folksonomy can be modeled by considering its three main elements which are user, tag and resource. Here the tag represents any keyword or phrase used by user to annotate certain resource, and each annotating action (combination of several annotating actions is commonly referred as collaborative tagging) can be represented in terms of user-tag-resource triple. Such representation of folksonomy is called tripartite model [18], [14]. The Figure 1 provides the graphical representation of tripartite model.

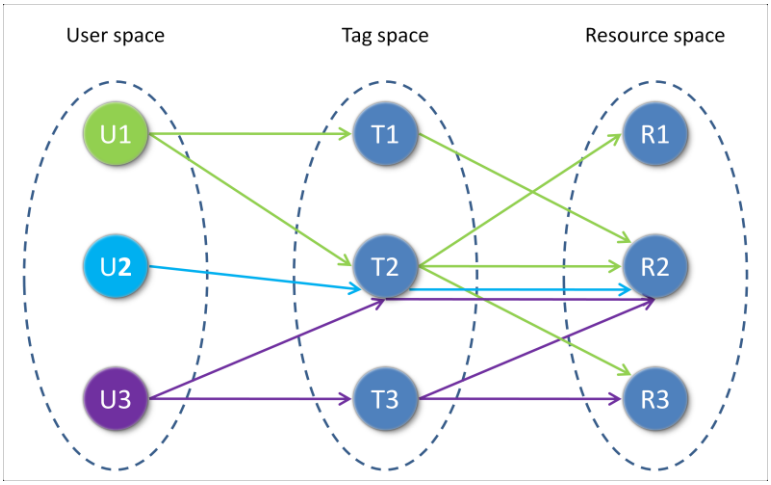


Figure 1. The graphical representation of tripartite model of folksonomy.

As you can observe from the Figure 1, the dynamics of collaborative tagging can be quite complex: the same user can annotate many resources, and the same tag can be used by different users to annotate different resources.

The tripartite model of folksonomy can be mathematically defined as a tuple $F := (U, T, R, Y)$ where

- U is finite set for Users $U=\{u_1,...,u_l\}$
- T is finite set for Tags $T=\{t_1,...,t_m\}$
- R is finite set for Resources $R=\{r_1,...,r_k\}$
- Y is ternary relation such as $Y \subseteq U \times T \times R$

The Y can be viewed as a set of all user-tag-resource triples and as collection of all annotation actions that were performed.

2.2 Types of Folksonomy

It is commonly agreed that there are three types of folksonomies depending on restrictiveness of the tagging: broad, narrow and personal folksonomies. This section defines each of the folksonomies and discuses the differences between them.

The broad folksonomy has the complete freedom of tagging comparing to other types, because it allows for any user to tag any resource with any tag of his choice [33]. The main characteristic of broad folksonomy is that it permits one user to tag resource with tag even if the resource was tagged with the same tag by another user before. doThus broad folksonomy has notion of “frequency of tag” for each resource. Also in broad folksonomy each individual annotating action is tracked, so it is possible to exactly know which user provided which tags. The typical example of broad folksonomy is Delicious[34].

The narrow folksonomy is more restrictive than broad folksonomy in a sense that it doesn’t allow providing the same tag again, which means every new tag assigned to resource must be unique, and should not be similar to previously assigned tags[33]. Another limitation in narrow folksonomy is impossibility to know which use provided which tag for particular resource. Most popular narrow folksonomy example is Flickr[35].
Finally there is personal folksonomy. The personal folksonomy has the most restrictions among the folksonomy types and allows user to tag only those resources that he has created [33]. Although the tags are visible to other users, those users cannot contribute

with their own tag. Example of such kind of personal folksonomy is Gmail where user can tag his e-mails.

Further in this thesis please notice that when we will refer to broad folksonomy simply as folksonomy, and the type of the folksonomy will be explicitly mentioned when we will refer to any other type of folksonomies.

2.3 Characteristic of Folksonomy

Folksonomy has several advantages and disadvantages comparing to other knowledge representation approaches. The Figure 2 provides the comparison of folksonomy to other knowledge representation techniques with respect to three main characteristics: expressiveness, cost of development and skills required for development.

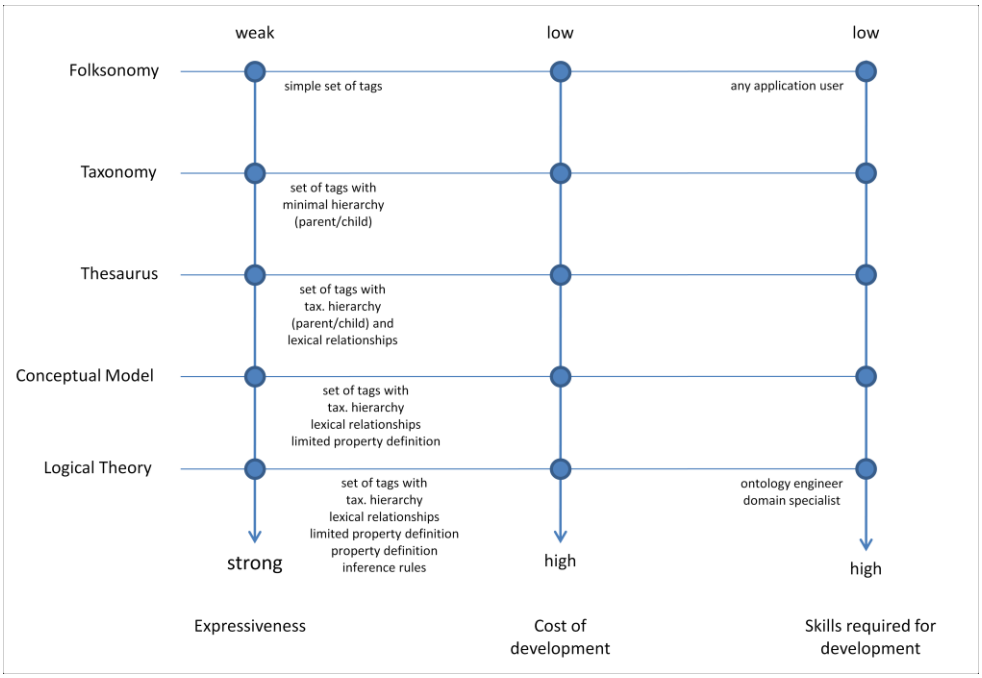


Figure 2. Knowledge representation spectrum.

As it can be observed from the above figure, the folksonomy requires almost no special skills for its development. Because the folksonomy based annotation is simply aggregation

of tags, it is easy to use and create the folksonomy. At the same time the development cost of folksonomy is also low comparing to development cost of other knowledge representation techniques. This is due to fact that any general user of the system can participate in the development.

Despite of those advantages folksonomy has one major disadvantage. The disadvantage derives from the fact that folksonomy was originally developed for human users. Although folksonomy has certain useful semantics that can be exploited to refine navigation, search and retrieval, these semantics are expressed implicitly. It is not that hard for human to reveal and exploit those implicit semantics in folksonomy, but in order to be processible by software agents the semantics should be expressed explicitly.

For example, taxonomy conveniently expresses the semantics in form of parent-child hierarchy which can be easily understood by machine, but folksonomy lacks this kind of infrastructure, therefore it has lowest level of semantic expressiveness comparing to other knowledge representation techniques.

The folksonomy has number of unique and special characteristics compared to ontology. The first and probably the most important of those characteristics is the existence of social dimension. The folksonomy extends the conventional two dimensional model of knowledge consisting of concept (tag) and instance (resource) with a social dimension[36], represented by community of agents (users). This social dimension forms what is known as implicit social network where users are indirectly linked with others by sharing the same tags and/or objects[9]. Since the study of dynamics of such social network is not directly related to the objective of this research, social dimension has some impact on dynamics and semantics of tagging which is need to be considered. The social dimension has several implications on the semantics of tagging:

- The social dimension makes it impossible to apply the traditional approach of ontology where the conceptual meaning of the tag is as precise as possible and grounded in the domain. Instead the conceptual or shared meaning of the tag is defined by negotiating between individual agent's subjective assignments of tags into resources[9]. The clear example of such negotiation is tags' usage frequencies which are considered as a reliable indicator of the usefulness and acceptance[10]. Therefore the conceptual meaning of the tag is relative depending on social dimension,

comparing to ontology where as it was mentioned before the meaning is precise and grounded into the domain.

- Despite having defined the conceptual meaning of tags, the social dimension also has influence on identifying relationships among tags. For example, the following association rule is often true in folksonomy: "users assigning certain tags to some resources often also assign another set of tags to those resources"[18]. These kinds of association rules make it possible to identify the existence of relationships among different tags based on social dimension.

Some tend to consider absence of structure in the classification of the information as an another unique characteristic of folksonomies which ensured its success on web[20]. Although it is true in some sense, it also can be viewed as a major disadvantage at the same time.

2.4 Semantic Value of Folksonomy

Many researchers agree that folksonomy can be beneficial in terms of semantic value it contains [1]. Folksonomy is a bottom-up classification method in a sense that classification of the content is done with respect to tags provided by each individual user who annotated that content. It was proven that collaborative tagging follows a power law curve, which means that after 100 or so the distribution of tags becomes stable [30]. This means that users are able to negotiate on common set of tags that best describes the annotated resource. The manual analysis of agreed set of tags revealed that there are certain relationships among the tags that annotate same or similar resources [14]. For example many tags can be synonymous to each other, or one tag can be just morphological variation of another tag, or some tag can have narrower or broader meaning than other tag. This kind of tag relationships can be obvious to human users, but not to software agents, this is why in this work those relationships are referred as implicit semantics.

Definition: *implicit semantics are the relationships among the tags which are based on semantic meaning of the tags and not stated explicitly.*

2.5 Folksonomy-based Systems

The folksonomy is widely used among web applications. It was first introduced in 2004 and since then its popularity and rate of adoption in other systems is increasing rapidly. Folksonomy used in applications mainly for purpose of organizing the content and enhancing its retrieval by using tag based searching.

One of the most famous applications which widely use folksonomy are Delicious and Flickr. Delicious is the largest collaborative tagging application on the Internet and provides same service as website bookmarking on internet browsers. Delicious is a social bookmarking applications which allows users to store their bookmarks on the Internet where it can be viewed by other users. Besides storing bookmark on the Internet, Delicious also allows users to annotate bookmarks with tags. Delicious uses broad folksonomy approach for annotating, this means that any user can annotate any bookmark no matter who created that bookmark.

Flickr is the largest Internet based application for organizing and sharing photos. By using Flickr users can upload their photos into the Internet where it can be seen by other anyone. Similarly to Delicious, Flickr also provides tagging service for uploaded photos, but, comparing to Delicious, Flickr uses narrow folksonomy approach which means that user can tag only own photos.

Other folksonomy-based applications include CiteULike – for organizing publications and papers, IMDB – movie tagging service, YouTube – video sharing and tagging service, Newsvine – social news bookmarking service. All those applications use folksonomy as a main technique to organize the content created by users. They provide a content search and retrieval based on user created annotations. This means that given a user query the search is done by matching query term with tags. This kind approach is very limited in terms content search and retrieval, as it will be discussed in next sections.

Chaper 3. Related Works

3.1 Ontology for Tagging

There are number of works that tried to create an ontology that would describe the major elements of folksonomy. Notably the author of [20] proposes to create ontology that will describe tagging. The author essentially tries to model it by defining class Tagging and relating it to user, tag, resource and source. Although author do not propose any particular ontology structure. It is one of the first tries to model tagging and corresponding source which provides the tagging. The paper [21] develops further the idea proposed in [20] and proposes the actual ontology shown in figure below.

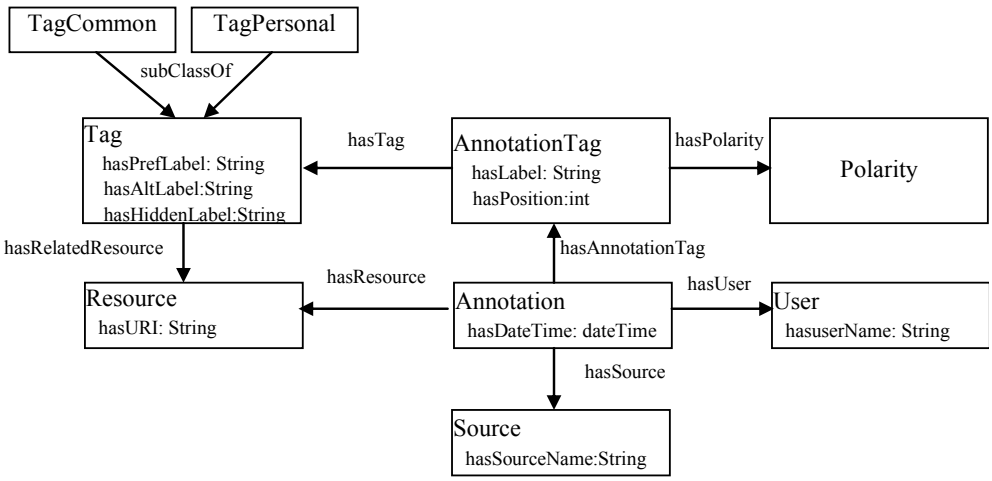


Figure 3. Ontological structure proposed for modeling the tagging.

The proposed ontology is very naïve and overly simplistic. Also the ontology tries to model number of features which are almost not used in folksonomies. For example ontology tries to model positive and negative tagging by defining class Polarity. Most of the folksonomy-based applications do not provide interface to negative tagging. Additionally authors try to model relationship between tag and annotation through intermediate class AnnotationTag. The rationale behind this design decision is not quite

obvious and seems to give no benefit. Also various authors try to model semantic equality between tags that have same meaning but different labels through defining the multiple datatype properties inside Tag class. This design decision is especially bad because it fails to model the user's tagging action in folksonomy. Using the folksonomy above it is impossible to know which user provided which tag.

Another attempt to model tagging is done in [22]. The basic approach is almost same as in [21]. There is main class Tagging to which all other classes are related. The ontology is also overly simplistic and fails to model fully model folksonomy. For example there is no class modeling the system which provides the tagging service. But the project in [21] is still in progress and much design of ontology might change.

3.2 Tag filtering

There is a well established research foundation on finding categories of noisy tags within folksonomy. For example [3], [30] and [31] provide a very good discussion about types of tags and their features within the folksonomy. But it seems that it is quite opposite case in terms of attempts to filter out those noisy tags. Many researchers seem to just ignore tag filtering step in their research. Despite those trends, there are few papers such as [3], [6] and [32] which describe good techniques for filtering noisy tags. Some of these papers even discuss about possibility of using online resources like Google for filtering purpose.

3.3 Deriving semantics from folksonomies

There are number of works that tried to extract semantics from folksonomies. Majority of those works are based on various clustering algorithms. For example works described in papers [2],[4] and [12] are all using clustering algorithms derive certain relationships among tags. But in all cases, although the clustering algorithms were able to group related tags together, they failed to identify specific relationships between specific instances of tags.

One notable approach to directly measure the degree of relationship between pair of tags was described in paper [13]. Here authors try to estimate the probability that one tags subsumes another one. Here the term subsumes mean that the meaning of one tag also

includes the meaning of another tag. Such probability is calculated by finding conditional probability of one tag depending on another tag. Then the subsumption is of one tag by another is decided by comparison of conditional probabilities of tags to certain threshold. As it is described in reference paper the tag X subsumes the tag Y if: $P(x|y >= 0.8)$ and $P(y|x < 0.8)$. Such kind of subsumption calculation allows to build hierarchy of parent child nodes in tag collection.

The one major problem is that this approach seems to have lost of noisy relations (false relations among the tags) when it is applied on whole tag collection. Thus the approach needs some way to decrease the amount of noisy relations.

3.4 Semantic Middleware

As far as author of this thesis is informed, there is no previous work done on building similar applications that serve as semantic middleware for folksonomy-based applications.

Chaper 4. FolkSpace – A Semantic Middleware for Folksonomy-based Applications

4.1 Definition of FolkSpace

FolkSpace is a middleware for emergent semantic systems. Let us disambiguate the meaning of semantic middleware based on following definitions.

Here is how Wikipedia defines the term middleware in computer science: "computer software that connects software components or applications. The software consists of a set of enabling services that allow multiple processes running on one or more machines to interact across a network". And here is how Wikipedia defines the term semantic: "study of the meaning of signs, and the study of relations between different linguistic units: homonymy, synonymy, antonymy, polysemy…".

The term "emergent semantics" is a widely accepted term referring to semantics that arise in a community of self-organizing, autonomous, networked and localized agents co-operating in dynamic, open environments, each organizing knowledge according to a self-established ontology, establishing connections and negotiating meaning only when it becomes necessary for co-operation[36]. It is believed that such emergent semantics can be obtained from collaborative tagging or otherwise folksonomies [37].

"The emergent semantics system is defined as information systems that combine informal and formal semantics approaches (i.e. folksologies and ontologies) to optimally serve the evolving requirements of communities of human and machine information providers and users" [10].

Based on previous definitions we can provide the definition of FolkSpace as the computer software that connects folksonomy-based applications, thereby promoting interoperability between them and enabling the exchange of machine processible semantics data extracted from folksonomies. FolkSpace basically supports following functionalities:

- connects multiple folksonomy-based services across the network
- extracts implicit semantics from folksonomy and represents it in explicit way

- provides querying service of multiple folksonomies that are from different services

4.2 Semantic power of FolkSpace

The power of FolkSpace comes not only from unified query interface of folksonomy and its backbone ontology, but also from its ability to work with implicit semantics. FolkSpace is a smart system which has ability to extract implicit semantics from folksonomy. That is why FolkSpace is referred to as semantic middleware. Currently the system is able to identify following implicit semantics from folksonomy:

- check if tags have same meaning based on their morphological similarity; for example tags “blog”, “blogging” and “blogs” have same meaning although morphologically they are different
- check if tags are synonymous
- build hierarchy of tags where one tag assumes another tag; for example if tag “vehicle” assumes tag “car” than “vehicle” is parent of “car”.

Despite ability to identify implicit semantics above, FolkSpace can also validate tags as valid or invalid words using common online and offline knowledge sources. This gives the FolkSpace the ability for not only checking for misspelled tags, but also check if meaning of the tag is commonly accepted. As result of those processing the final collection of tags produced by FolkSpace can be considered as a controlled vocabulary or very light weight ontology which can be used to effectively answer user queries.

4.3 Overall Architecture

The whole architecture of FolkSpace consists of four major components which are Ontology Repository, Model Repository, Model Creator and SPARQL query engine. This section briefly describes the roles of those components in overall architecture shown in Figure 4.

First, there is Ontology Repository which contains the description of FOM Ontology and set of FOM Rules. The FOM Ontology describes the common structure of folksonomy in a machine interpretable way. For example it defines basic elements of folksonomy as user, resource, tag and describes how they are related to each other. FOM rules are used to

express statements which cannot be described by ontology. Because most of the ontology languages have certain limitations in expressing conditional relations, the conditional relations are expressed separately in form of rules rather than ontological hierarchy.

The role of Model Creator is to obtain folksonomy from folksonomy-based application and to convert it into its equivalent semantic model. The folksonomy is obtained through RSS feeds, which are commonly provided by majority of the applications, and then the gathered folksonomy is processed using FOM Processor. FOM Processor is a core part of this component, it is the actual place where the folksonomy is processed and its semantic model is created. The semantic model of the folksonomy is created in form of RDF triples using the Jena library. The model describes system from which the folksonomy was obtained and folksonomy itself in terms of users, resources and tagging actions performed by each user. The description of tags used in tagging is stored separately from semantic model, in a model called Tag Model. Storing tag description separately in common model removes the problem of redundancy of same tag among different semantic models, and also provides the connection between folksonomies from different applications. Tag Model also contains the explicit description of implicit semantics extracted from folksonomy. The FOM Processor creates the semantic and tag models in accordance with the hierarchy defined in FOM Ontology and the rules defined in FOM Rules, so each semantic model can be treated as a collection of instances of classes defined in FOM Ontology.

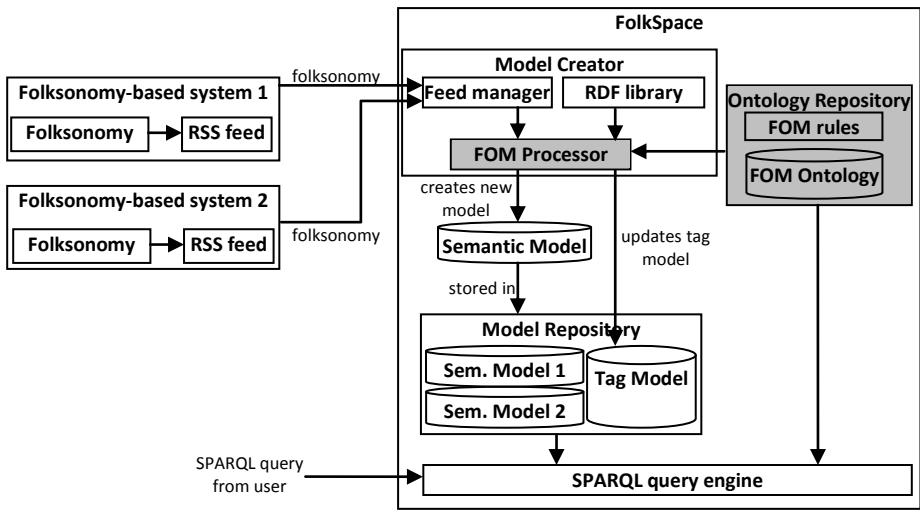


Figure 4. The overall architecture of FolkSpace

All models which are created by Model Creator, including Tag Model, are stored in Model Repository. Model Repository provides convenient interface of managing the created models.

Final component of FolkSpace is a query interface, which allows to execute structured query against the Model Repository and Ontology Repository. Because all models are stored in RDF format the SPARQL query language can be used for querying.

4.4 Semantic Model of Folksonomy

This section provides the definition and description of semantic model of folksonomy.

Definition: *The semantic model of folksonomy is a machine interpretable description of the folksonomy written in form of RDF triples.*

The following types of information are described in the semantic model:

- Service description – information about application from which the folksonomy was obtained. The example can be short description about Delicious or Flickr.
- User description – this is the information about all users who participated in creating particular folksonomy.
- Resource description – this is the information about all resources which were described using particular folksonomy.
- Annotating action description – this is the description of all annotating actions that were done by users over resources using certain tags.

Let's suppose there is a folksonomy which was obtained from Delicious with properties defined in table below.

Table 1. Properties of the example folksonomy.

Name of application	Delicious
Users	UserA and UserB
Tags	TagA, TagB and TagC
Resources	ResourceA and ResourceB

Annotating actions	Annotation1: UserA annotates ResourceA with TagA Annotation2: UserA annotates ResourceB with TagC, Tag B Annotation3: UserB annotates ResourceB with TagB

Then the corresponding semantic model will be created as shown in Table 2. Please notice that semantic model shown in Table 2 is not complete, and some statements that also should be included in semantic model are not shown in it. The Table 2 shows semantic model which was simplified and shortened for demonstration purpose.

From the table you can see that semantic model describes service with name *Delicious* and states that it has some folksonomy which is assigned id "*Delicious.Folksonomy*". Next the model states that there are two resources described by "*Delicious.Folksonomy*" and provides the properties of those resources such as type of resource and URL. Next follows the description of the users who uses *Delicious* for annotation purpose. Description of annotating actions is the main part of the semantic model which provides actual connection between description of resources, users, tags and folksonomy.

Table 2. Corresponding semantic model of folksonomy defined in Table 1.

Description of application and folksonomy	Description of users
:Delicious *a :Service* *: hasServiceName "Delicious";* *:hasServiceURL "http://www.delicious.com".* *: Delicious.Folksonomy* *a : Folksonomy* *:providedBy :Delicious .*	*: UserA* *a :User ;* *:hasUserID "UserA" ;* *:uses :Delicious .* *: UserB* *a :User ;* *:hasUserID "UserB" ;* *:uses :Delicious .*
Description of resources	**Description of annotating actions**
: ResourceA *a :Resource ;* *:describedBy : Annotation1;* *:hasType "multimedia" ;* *:hasURL "http://www.ResourceA.com" .* *: ResourceB* *a :Resource ;* *:describedBy : Annotation2, Annotation3;* *:hasType "multimedia" ;* *:hasURL "http://www. ResourceB.com" .*	*: Annotation1* *a :UserAnnotation ;* *:hasTag : TagA ;* *:hasUser : TagA ;* *:describes : ResourceA ;* *:belongTo: Delicious.Folksonomy.* *: Annotation2* *a :UserAnnotation ;* *:hasTag : TagC, :TagB ;* *:hasUser : UserA ;* *:describes : ResourceB;* *:belongTo: Delicious.Folksonomy.* *: Annotation3* *a :UserAnnotation ;* *:hasTag : TagB ;* *:hasUser : UserB ;* *:describes : ResourceB;* *:belongTo: Delicious.Folksonomy.*

As you can see from Table 2, the semantic model makes use of some set of predefined classes like *Resource* to define resource type; *UserAnnotation* to define annotating action; *User* to define user type; *Service* to define folksonomy-based application; and *Folksonomy* to define folksonomy. Those classes are defined within FOM Ontology, and semantic model is created according to the structure defined in FOM Ontology. This is the same case for the properties. For example, it can be observed that description of *Delicious* has properties such as *hasServiceName* and *hasServiceURL*. Those properties are also defined

within the FOM Ontology. For more detailed discussion about FOM Ontology please read section 4.6.

4.5 Tag Model

If you look at the Table 2 then there is no description of the tags within the semantic model. This is because description of tags is not stored within semantic model, but it is stored separately in a model called Tag Model, and each semantic model refers to Tag Model for tag description as it is shown in Figure 6.

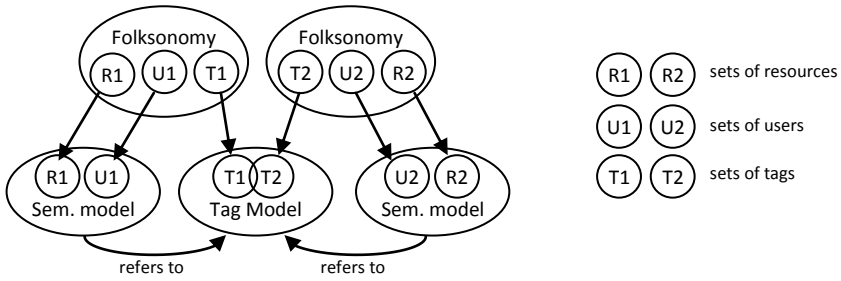


Figure 5. The descriptions of tags from folksonomies are stored in a single model called Tag Model.

Such design consideration was chosen due to several factors as discussed below:

1. First of all, it is very likely that there will be redundant sets of tags across different folksonomies. Storing description of tags in semantic models can results in redundant description of same tags in different semantic models. Instead storing tag description in Tag Model ensures that each distinct tag has only one description no matter how many semantic models use that tag.
2. Second, using Tag Model simplifies the process of finding semantic models that might be related to each other. For example if *UserAnnotation* in one semantic model refers to same set of tags from Tag Model as another *UserAnnotation* in another semantic model then it is very likely that annotated resource might related. Although it is completely possible to reveal such connection without using Tag Model, it would be much harder because it will require some way of measuring equivalence of two tags in different semantic models.

3. Finally, use of Tag Model significantly simplifies the process of automatically deriving implicit semantics from folksonomy, and storing that semantics in explicit machine interpretable way. Actually the derived relationships among tags are stored also in Tag Model.

As a result of such division the full description of folksonomy can be obtained through combination of corresponding semantic model and Tag Model.

Despite simply describing the collection of tags Tag Model also contains the descriptions of implicit semantics among those tags. Because Tag Model is a collection of all tags from all folksonomies it is a perfect place to start analysis of the implicit semantics. As shown in Figure 7, at first all tags are collected in Tag Model, then analysis of implicit semantics is performed on the collection and the description of extracted semantics is stored back in Tag Model.

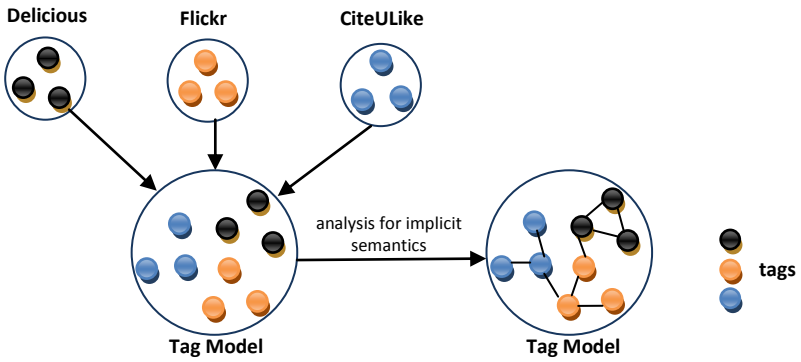


Figure 6. Tags from different folksonomies are collected in Tag Model and then processed for implicit semantics, revealed semantics are stored back in Tag Model

Let me provide example of Tag Model descriptions given three folksonomies with the set of tags as show in Table 3.

Table 3. Example folksonomies and corresponding tag spaces.

Folksonomy 1 tag space:	car, automobile
Folksonomy 2 tag space:	cars, vehicle
Folksonomy 3 tag space:	car, auto
Possible implicit semantics:	car and automobile are synonymous car and cars are same tags (morphological variations) auto and automobile are same tags (morphological variations) vehicle assumes car (vehicle is parent of car)

Table 4. Descriptions contained within Tag Model given folksonomies from Table 3.

Description of tags	**Description of uncovered implicit semantics.**
:ta_ car *a :Tag ;* *:tagName " car " .* *:ta_ automobile* *a :Tag ;* *:tagName " automobile " .* *:ta_ cars* *a :Tag ;* *:tagName " cars " .* *:ta_ vehicle* *a :Tag ;* *:tagName " vehicle" .* *:ta_ auto* *a :Tag ;* *:tagName " auto" .*	*:mg_ car* *a :MorphologicalGroup ;* *:hasMember :ta_ car , :ta_ cars ;* *:hasStem "car " .* *:mg_ auto* *a :MorphologicalGroup ;* *:hasMember :ta_ automobile, :ta_ auto;* *:hasStem "car " .* *:sy_000141* *a :SynonymousGroup ;* *:hasMember :ta_ automobile , : ta_ car .* *:ta_ vehicle* *:assumes :ta_car .* *:ta_ car* *:assumedBy :ta_vehicle .*

Also lets assume that tags were not only described within Tag Model, but also the analysis of implicit semantics was performed on the collection, and it was able to identify

the implicit relationship among the tags as it is shown in Table 3. Then the corresponding Tag Model will be as it is shown in Table 4.

As you can see from Table 4 the Tag Model will consist of two parts. First, there will be simple list of descriptions of tags, which is show in left column of Table 4. If you notice, each tag has only one description independently of number of folksonomies in which it occurs. Second, Tag Model has list of descriptions of implicit relations among the tags that were revealed during the analysis. This part is show in right column of Table 4. The first statement in right column of the table says that tags “car” and “cars” are same tags. The second statement says that tags “auto” and “automobile” are also same tags. The third statement says that tags “automobile” and “car” are synonymous.

As it was the case with semantic model, the Tag Model makes use of some set of predefined classes such as *Tag*, *MorphologicalGroup* and *SynonymousGroup*. Similarly to classes used in semantic model, those classes are defined within the FOM Ontology. The corresponding properties such as *tagName*, *hasMember* and *hasStem* are also defined within the FOM Ontology. The structure of the FOM Ontology will be described in next section.

4.6 FOM Ontology

FOM Ontology is the core part of FolkSpace. It is used almost everywhere within the FolkSpace. The list below provides the description of main functionalities of FOM Ontology within the system:

1. It defines in a machine interpretable format the common structure of folksonomy in terms of its elements and relationships between them.
2. It defines what kind of implicit relations can be contained and extracted from the tags.
3. FOM Ontology is also used as a schema for creating the semantic models and Tag Model.
4. Finally the structure defined in FOM Ontology is used for disambiguating the structured query sent to FolkSpace.

The Figure 8 provides the graphical view of FOM Ontology structure. As we have discussed in previous sections, description of folksonomy is stored in two separate models:

semantic model and Tag Model. In Figure 8 the part of the ontology highlighted by dashed line defines the structure of semantic model, and the part of ontology highlighted with dotted line defines the structure of Tag Model.

Thus FOM Ontology can be broken apart into two schemas. The next sections will describe those schemas separately in more details.

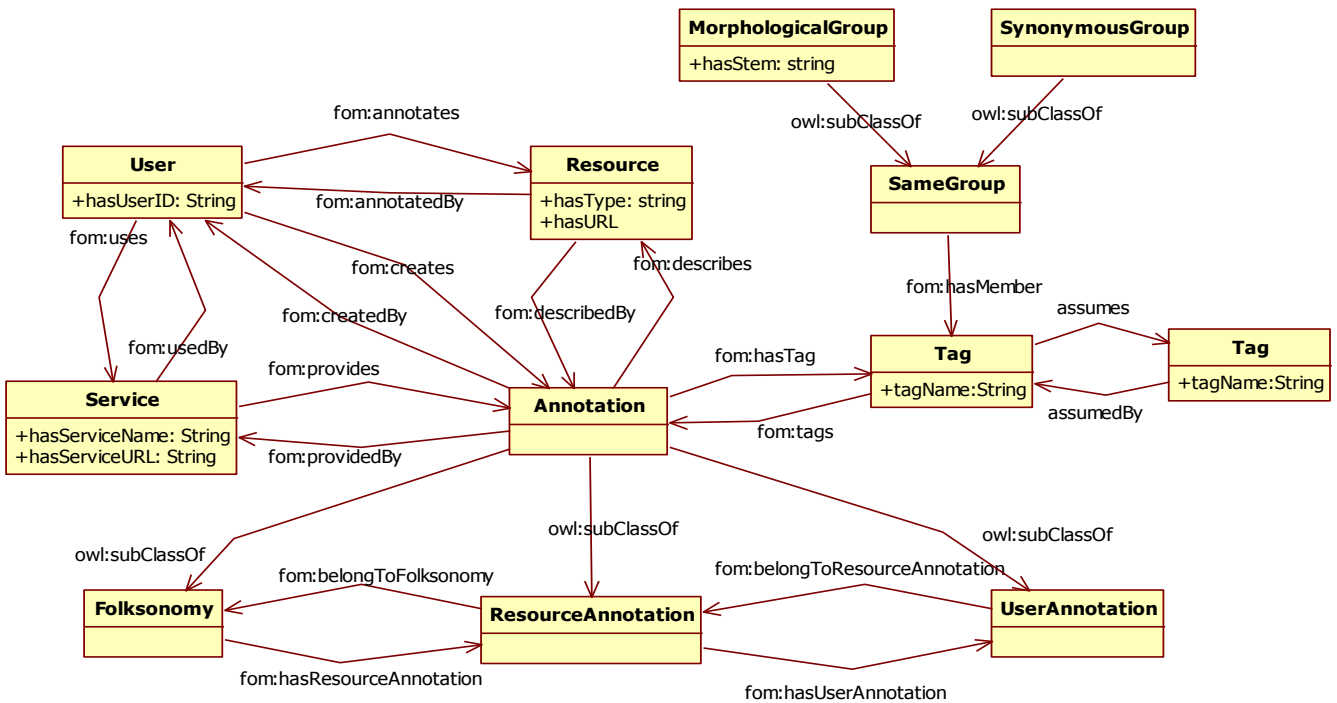


Figure 7. The structure of FOM Ontology; the part of ontology within dashed border defines the structure of semantic model, and the part highlighted with dotted line defines the structure of Tag Model.

Table 5. FOM Ontology classed used in semantic model

Classes	Class descriptions
Service	This class represents any folksonomy-based application. For example, instances of this class can be Delicious or Flickr.
Resource	This class represents any content which is being annotated. For example in Delicious it would be a bookmark of the website, but in Flickr it would be a photo image.
User	This class represents instances of all users that user certain *Service* to annotate certain *Resource*.
Annotation	This class represents all instances any type of annotation. This is the core class of FOM Ontology, because it binds together other classes such as User, Service and Resource. It has direct sub-classes of Folksonomy, ResourceAnnotation and UserAnnotation.
Folksonomy	This class represents the all instances of folksonomy. Folksonomy is a sub-class of Annotation, which means that Folksonomy also belongs to some service, has some users and describes some resources. But differently from Annotation class, Folksonomy has restriction that it can belong to only one Service. Folksonomy also can have several ResourceAnnotations.
ResourceAnnotation	This class represents the instances of all resource annotations. As it is case with Folksonomy class, it has restriction that it can belong to only one Service. In addition it has another restriction that it can describe only one resource. So ResourceAnnotation represents annotation for one particular Resource.
UserAnnotation	This class represents the instances of all user annotations. It has same set of restrictions as ResourceAnnotation, and one additional restriction that it can have only one User.

4.6.1 FOM Ontology schema for semantic model

This section will discuss about design of part of FOM Ontology which serves as a schema for the semantic model of folksonomy. As it was described before in section 4.3, the semantic model of folksonomy uses predefined set of classes to identify types of the elements in folksonomy. This group of classes together with their properties is highlighted by dashed line in Figure 8. Also you can refer to Table 5 provides the short description of those classes. Please notice that although class Tag is shown as a part of the schema, the actual instances of class Tag are not described within semantic model, but rather referenced from semantic model.

One thing to emphasize here is that classes Folksonomy, ResourceAnnotation and UserAnnotation are sub-classes of class Annotation. Therefore obviously those classes can have relation with any of the classes which has relation with class Annotation. How this design decision is exploited will be discussed later in subsection 4.6.2.

It might not be directly obvious the rationale behind the design decision made in FOM Ontology according to definition of three separate classes: *Folksonomy, ResourceAnnotation* and *UserAnnotation*. In order to understand this design choice the one should consider the structure of tag space in folksonomy. The Figure 8 provides the Venn diagram of tag space.

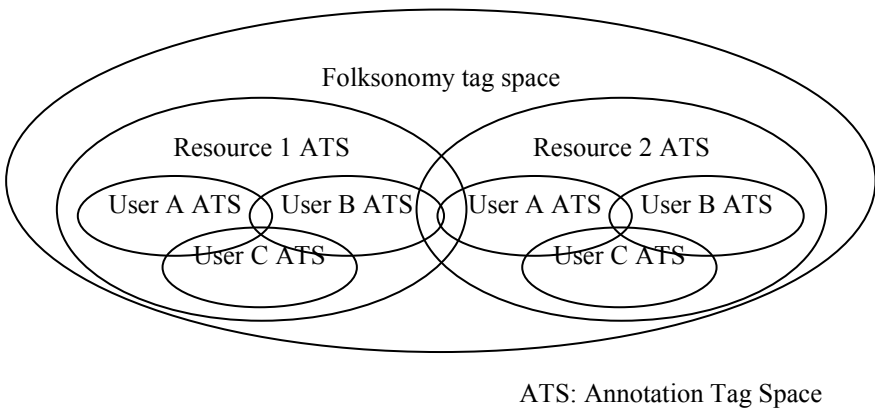


Figure 8. The Venn diagram of tag space in folksonomy.

The collection of all tags in folksonomy forms a tag space. But because folksonomy usually annotates several resources, and each resource has its own collection of tags that annotates it, the whole tag space can be divided into several intersecting parts. In this work those parts are referred as resource annotation tag space. For example in Figure 9 the folksonomy describes two resources, so it has two resource annotation tag spaces. Next, the same resource can be annotated by several users, in this case the tag space of the resource can be further divided into tag spaces of individual users who annotate that resource. So folksonomy can have quite complex structure of tag space depending on number of resources and number of users it has. The three classes *Folksonomy*, *ResourceAnnotation* and *UserAnnotation* are designed to model this division of tag spaces.

The class *UserAnnotation* models the user annotation tag spaces, the smallest tag space in folksonomy. As expected, class *UserAnnotation* can have only one user (the cardinality for *fom:createdBy* property is one) and can describe only one resource (the cardinality for *fom:decribes* property is one).

The class *ResourceAnnotation* models the resource annotation tag space. Correspondingly the instance of this class can describe only one resource (the cardinality for *fom:decribes* property is one). Because this resource annotation tag space can have several user annotations tag spaces within, the instance of *ResourceAnnotation* class can be related to several instances of corresponding *UserAnnotation* class. Also it inherits all the users from instances of *UserAnnotation* class to which it relates (thus cardinality for *fom:createdBy* property is *).

Finally the class Folksonomy models the whole tag space. Thus its instance can be related to several instance of *ResourceAnnotation* class, as a consequence it can describe several resources and can have several users (the cardinalities for both *fom:createdBy* and *fom:describes* properties are *).

4.6.2 Representing types of folksonomy in FOM Ontology

Previously in Background part of this thesis the three types of folksonomies were described as broad, narrow and personal folksonomies. It is essential for FOM Ontology to be able to describe all three types of folksonomies because all of them are commonly used in applications. Thus this section describes how FOM Ontology can represent all three

types of folksonomies, but at first lets analyze in details the difference between those types in terms of tag space.

The Figure 10 shows three Venn diagrams each describing one of the types of folksonomy in terms of its tag space structure. The upper diagram which shows the structure of broad folksonomy is similar to one which was already shown in Figure 8.

The middle diagram shows the structure of tag space for narrow folksonomy. Because narrow folksonomy doesn't track annotations of each individual user, the whole resource annotation tag space cannot be divided into user annotations.

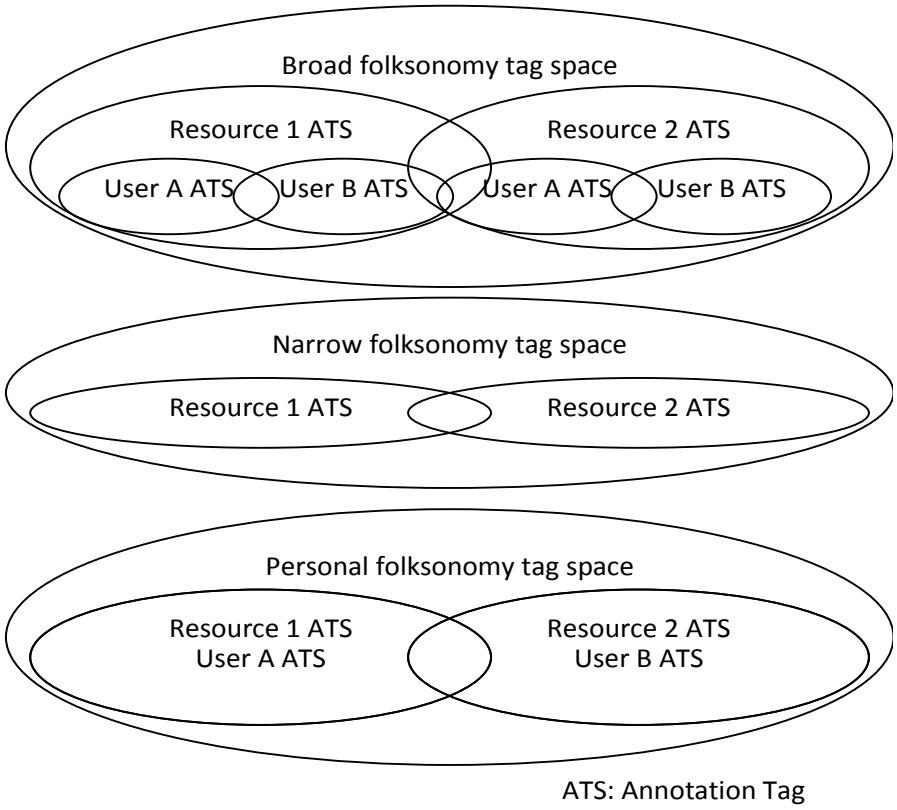
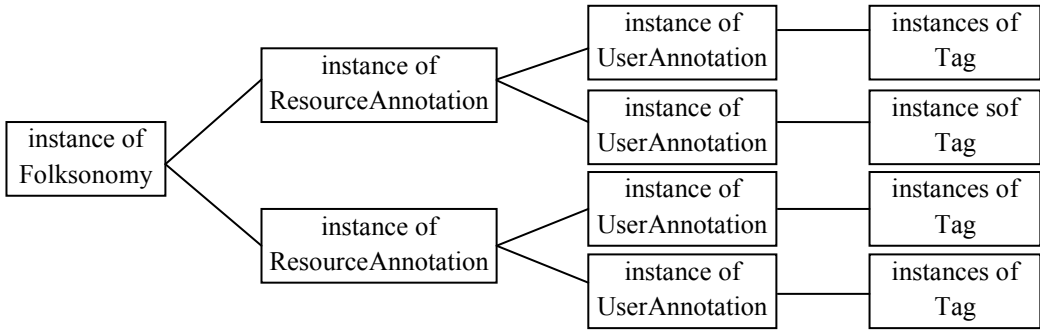


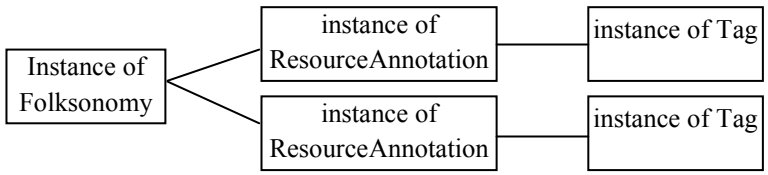
Figure 9. Difference between broad, narrow and personal folksonomies in terms of structures of their tag spaces

In case of personal folksonomy, user can tag only those resources which he has created, which means that no more than one user can tag any single resource. Therefore the resource annotation tag space will be exactly similar to user annotation tag space, as it is shown lowest diagram of Figure 9.

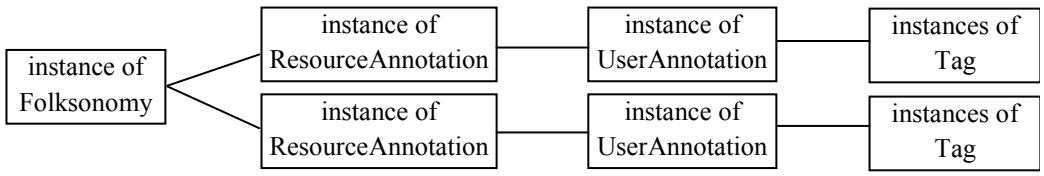
As you can see the main difference between three types of folksonomies derive from the structure of resource annotation tag space. So ability to model different resource structure of resource annotation will essentially give ability to model three types of folksonomies.



a) modeling the broad folksonomy



b) modeling the narrow folksonomy



c) modeling the personal folksonomy

Figure 10. Modeling three types of folksonomies using *Folksonomy*, *ResourceAnnotation* and *UserAnnotation* classes in FOM Ontology.

As it was discussed in previous section FOM Ontology uses three classes to effectively model tag spaces in folksonomy. Those classes also used to model different types of folksonomies. This is achieved through different combinations of instances of those classes as it is shown in figure below.

Careful analysis of the structure shown in Figure 11 will show that each of the modeling decisions basically replicates the corresponding diagram from Figure 10. For example, you can see that when modeling broad folksonomy the instance of *ResourceAnnotation* can have several instances of *UserAnnotation* related to it, and each instance of *UserAnnotation* can have several instance of *Tag* assigned to it.

In case of narrow folksonomy, the instance of *ResourceAnnoattion* don't have any instance of *UserAnnotation* related to it, therefore instances of *Tag* are directly related to instance of *ResourceAnnotation* (class *Tag* can be directly related to class *ResourceAnnotation* because *ResourceAnnotation* is also sub-class of *Annotation* class just like *UserAnnotation* class). This effectively models the situation in narrow folksonomy where resource has annotation, but it is not know exactly which user provided which tags.

Finally the graph c) in Figure 11 shows how personal folksonomy can be modeled in FOM Ontology. Each instance of *ResourceAnnotation* is related to only one instance of *UserAnnotation*, and each *UserAnnotation* is related to exactly one instance *ResourceAnnotation* forming inverse-functional relationship between those instances. This relation effectively models the situation where the resource can be annotated by only one user, which is the case in personal folksonomy.

4.6.3 FOM Ontology schema for Tag Model

The smaller part of FOM Ontology highlighted by dotted line in Figure 7 can be considered as a schema for Tag Model. This part of ontology defines what kind of structure the Tag Model should have. Let's quickly remind that *Tag* Model consist of description of tags themselves and also description of implicit semantic that were identified among those tags. First, the format of describing the tag is defined through single class *Tag*. The only one property value (*fom:tagName*) should be supplied to instance of Tag class which is label of that tag.

Table 6. Set of classes used for defining the structure of Tag Model

Class name	Class description
Tag	This class represents all instances of tags that are part of annotation.
SameGroup	This class is used to represent semantic equivalence of set of tags. It has direct sub-classes *MorphologicalGroup* and *SynonymousGroup*.
MorphologicalGroup	This class is used to represent semantic equivalence of set of tags that has same meaning, but different morphological forms.
SynonymousGroup	This class is used to represent semantic equivalence of set of tags that are synonymous to each other.

The format of describing the implicit semantics is more complex than format of describing the tags themselves. As it was discussed previously in section 4.2, FolkSpace is able to identify following implicit semantics in folksonomy:

- morphological variations of same tag
- synonymous tags
- assumption hierarchy among tags

The first semantic relation which morphological variation of same tag is defined through class *MorphologicalGroup*. When two instances of different tags are morphological variations of same tag, then those instances are assigned to same instance of *MorphologicalGroup* class. The Table 4 shows the usage of instance of *MorphologicalGroup* class.

The same approach is taken for tags that are synonymous. The synonymous relation between tags is defined through class *SynonymousGroup*. If two instances of different tags are synonymous then they are assigned to same instance of *SysnonymousGroup* class. You can also refer to Table 4 for example of usage of class *SynonymousGroup*.

The different approach is used for defining parent-child hierarchy. There is no class that represent the parent-child relation between two tags, but rather it is defined through two properties *fom:assumes* and *fom:assumedBy*. If one instance of tag is parent of another instance of tag then former instance links to latter one through *fom:assumes* property. The property *fom:assumedBy* is an inverse property of *fom:assumes* property. Finally whole

hierarchy is modeled by defining both properties as transitional properties. In Table 4 you can see example where tag “vehicle” assumes tag “car”.

4.7 FOM Rules

FOM rules play important role in establishing bridge between FOM Ontology and the reasoner. Rules allow reasoner to derive relationships among classes of FOM Ontology which otherwise cannot be expressed in ontology itself due to its limitations. All rules in FolkSpace are forward chaining rules and written using SWRL grammar. Let’s consider some scenarios where the rule is required.

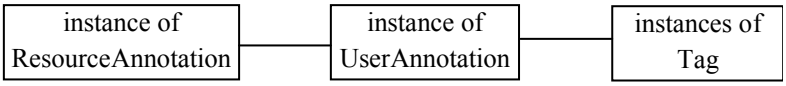


Figure 11. The example scenario where rule is required to define relation between instance of *Tag* and *ResourcesAnnotation.*

The Figure 12 shows the simple relation from FOM Ontology where instance of *ResourceAnnotation* has instance of *UserAnnotation*, and instance of *UserAnnotation* has instances of *Tag*. It would be natural to state that instance of *ResourceAnnotation* also has all instances of *Tag* from instance of *UserAnnotation*. The problem here is that such kind of transitive relationship is not state explicitly within the ontology so reasoner will not be able to derive the previous statement. The above problem is solved by defining the corresponding rule in Table 7. The rule is written in N3 format for readability purpose. Here the rule basically says that if some instance of *UserAnnotation* has instance of *Tag* and belong to instance of *ResourceAnnotation* then that instance of *ResourceAnnotation* has the same instance of *Tag*.

Table 7. The rule defining the propagation of instance of *Tag* ownership from instance of *UserAnnotation* to *ResourceAnnotation*

Rule: If instance of UserAnnotation has instance Tag and belong to instance of ResourceAnnotation then that instance ResourceAnnotation has the same instance of Tag
(?x fom:hasTag ?y) *(?x rdf:type fom:UserAnnotation)* *(?y rdf:type fom:Tag)* *(?x fom:belongToResourceAnnotation ?z)* *(?z rdf:type fom:ResourceAnnotation)* *-> (?z fom:hasTag ?y)*

Let's consider another more important case which is related to defining the semantic relationships among the tags. The example is shown in Figure 13.

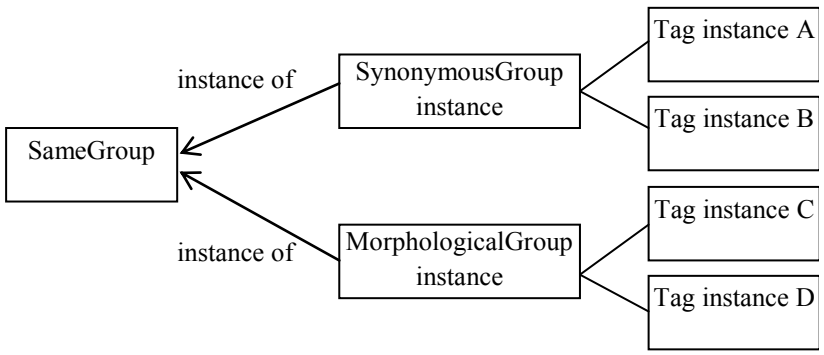


Figure 12. Example of synonymous and morphological relationships among tags.

Example in Figure 13 shows that there are two instances of *Tag* that are equivalent on morphological basis, and there are another two instances of Tag that are synonymous. If you notice, because *SynonymousGroup* and *MorphologicalGroup* are both sub-classes of *SameGroup*, the instances of those classes are also instances of *SameGroup* class. Therefore the equivalence of pair of tags is expressed by their membership in same instance of *SameGroup*. But that reasoner cannot derive the fact that instances of *Tag*

belonging to *SameGroup* are semantically equivalent just from the structure defined in Figure 13. Although ontology languages like OWL DL, allow expressing the equivalence of two instances, problem arises from the fact that equivalence in this case is conditional. Although most of the ontology languages are lack of expressiveness for conditional statement, such conditional equivalence can be expressed by rule.

Table 8. The rule defining the equivalence of two instances of Tag when they morphologically similar or synonymous

Rule: If different instances of *Tag* class are members of same instance of *SameGroup* class they are semantically equivalent.
(?x fom:memberOfSameGroup ?z) *(?y fom:memberOfSameGroup ?z)* *(?x rdf:type fom:Tag)* *(?y rdf:type fom:Tag)* *(?z rdf:type fom:SameGroup)* *-> (?x owl:sameAs ?y)*

The Table 8 describes such rule expressed in N3 format. The rule basically says that if two instances of *Tag* are members of same instance of *SameGroup*, then those tags are same. Please notice that equivalence of two tags is expressed through OWL property owl:sameAs. This is because FOM Ontology is described in OWL DL language.

The FOM Rules are supplied together with FOM Ontology for creating semantic model and Tag Model. FOM Rules are also used together with FOM Ontology for answering queries.

4.8 FOM Processor

FOM Processor is the second most important component of FolkSpace despite FOM Ontology. The main input to FOM Processor is folksonomy from Feed Manager. Feed Manager also supplies FOM Processor with information about application from which the folksonomy was obtained. Other supplementary inputs to FOM Processor are FOM Ontology and FOM Rules. Given this data FOM Processor creates new semantic model

and also updates Tag Model with description of new tags and semantic relations as it is shown Figure 14.

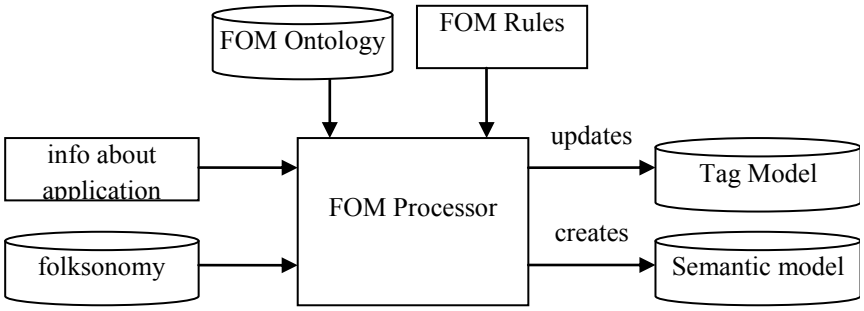


Figure 13. Inputs and outputs of FOM Processor.

4.8.1 Creating semantic model

The process of creating semantic model within the FOM Processor is pretty much straightforward. The Feed Manager supplies the FOM Processor with XML structured folksonomy, so FOM Processor matches the structure to the FOM Ontology classes and then creates the corresponding instances of those classes. The raw collection of instances can be considered as preliminary semantic model. This model is next supplied to a reasoner together with FOM Ontology and FOM Rules. The reasoner produces the final semantic model with additional inferred statements. The whole process is show in figure below.

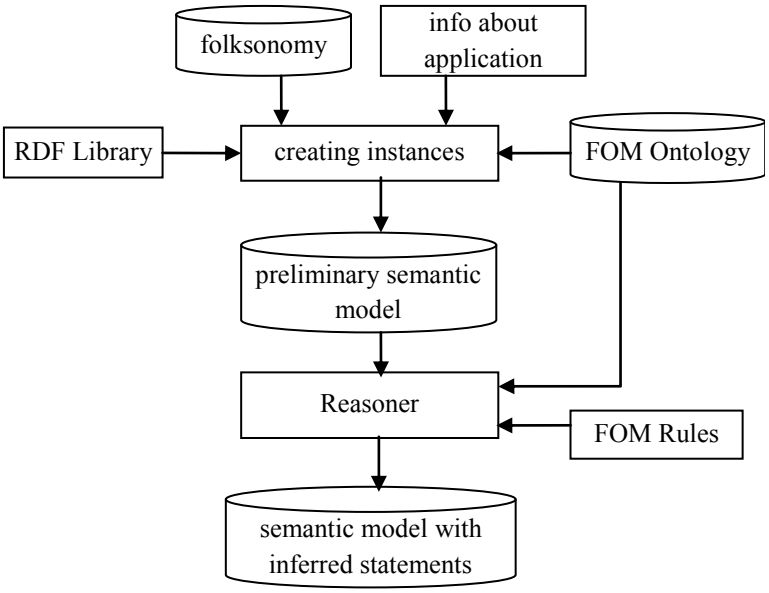


Figure 14. Steps for creating the semantic model inside the FOM Processor.

4.8.2 Updating the Tag Model

Previously it was explained that semantic model do not contain the description of tags, but only refers to Tag Model where all the description of tag instances are stored. With each new semantic model processed through FOM Processor the Tag Model is updated. This involves the complex process over the collection of the tags in folksonomy. The process is called Folksonomy-to-Ontology Maturing Process and it needs dedicated section for its explanation.

4.9 Folksonomy-to-Ontology Maturing Process

Folksonomy-to-Ontology Maturing Process (FOM Process) is a complex pipeline process which was specifically developed for FolkSpace. It consists of sequence of phase where each phase itself is also pipeline process. Given collection of tags from folksonomy the FOM Process consists of following four phases which follow each other:

- Tag filtering phase – removing irrelevant and noisy tags from the collection
- Lexicographical analysis phase – identification of morphological and variations and synonymous relations among the tags

- Statistical analysis phase – identification of hidden relationships among the tags through hierarchy analysis algorithm
- Model updating phase – building of RDF model from tag collection and identified relations, and updating Tag Model

The following subsections will discuss in details each of the phases.

4.9.1 Tag filtering phase

The main goal of this phase is to filter out the noisy tags from collection of tags. The Table 9 lists the types of noisy tags that are predominant in folksonomy.

Table 9. Types of noisy tags together with examples

Type	Description	Examples
Case sensitive	same tags written with different combinations of lowercase and uppercase letters	ADVERTISING, advertising, Advertising
Inappropriate beginning or ending	tags starting with characters other than letter or digit	.imported
Inappropriate length	tags consisting of only one character	D
Stop words	common stop words	an, the, on, of, me, I
Self-reference	represent relationship of user to content	mything, mycomment, mysociety, myp2p
Task-organizing	used by users for personal purpose	todo, toread
Multilingual	non-english words; words with special characters	Футбол, ΑΘΛΗΤΙΚΑ, Animação
Misspelled	Tags with spelling errors	busines
Compound tag	Tags consisting of two or more concatenated words	opencourseware, searchengine

If you notice the above table categorizes the multilingual (non-english words) tags as noisy tags. Although it is not completely correct, FolkSpace currently does not support other languages than English, therefore all non-english tags are filtered out.

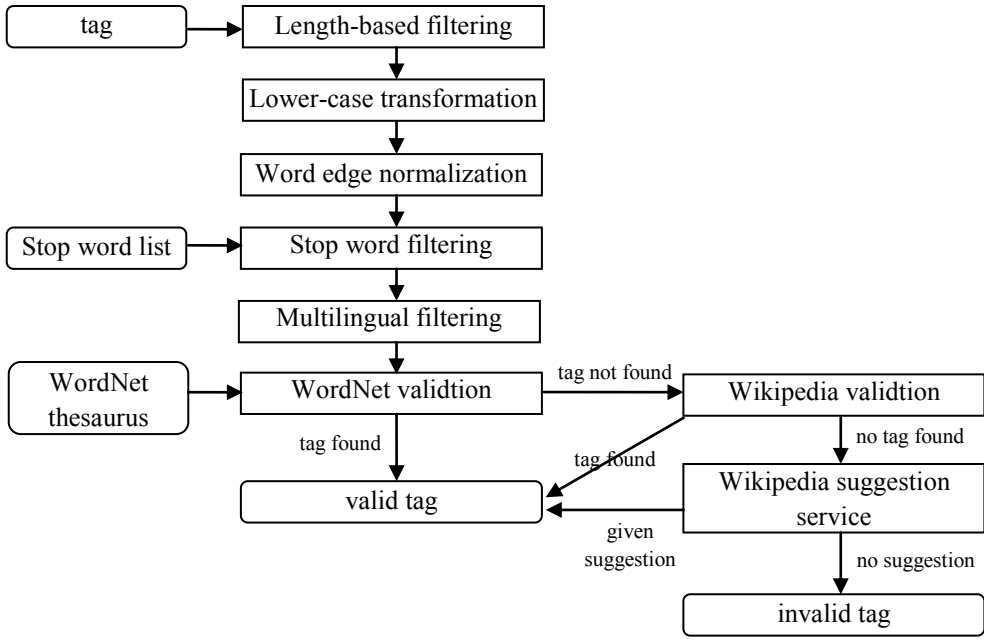


Figure 15. Steps in tag filtering phase.

In order to filter out the types of noisy tags mentioned in Table 9, the process consisting of number of sequential steps is applied on each tag in collection. The Figure 16 shows those steps in a sequence they should be done. In this process if tag is not multilingual then it is sent to be validated by WordNet. This is done in order to identify tags that may be misspelled or do not have any proper meaning. If tag is found in WordNet then it is considered as a valid tag, otherwise it is sent for further processing by Wikipedia. At this step the tag is searched among Wikipedia article names. If tag is found within article names then it is considered as a valid tag. In case if tag is not found within article names, Wikipedia still might give some suggestions which are usually corrections of misspelled word or tokens of compound words. If suggestion is given then the suggested words is returned as valid tag instead of original tag. If tag both not found within article names, and no suggestion was given then tag is deleted from collection as invalid tag.

The above algorithm is expected to filter out all types of noisy tags mentioned in Table 9, and it was proven to be efficient after its implementation was tested on actual dataset. Please, check Appendix for sample results from testing.

4.9.2 Lexicographic analysis phase

The goal of this phase is to identify implicit semantics among the tags, which are morphological variations of same tag and synonymous tags.

It is common that same word can have several morphological variations although the meaning of the word stays same. The most common example of such word is “blog”, which has two common morphological variations “blogs” and “blogging”. The words “blog, blogs and blogging” are very often used interchangeably, and it happens a lot in folksonomy that same resource annotation has all three variations of this word. Therefore identifying such morphological variations and using them for query extension can be advantageous.

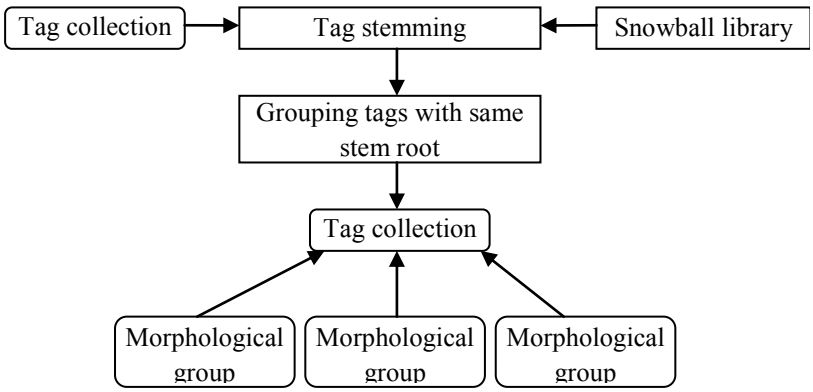


Figure 16. Process of identifying morphological groups.

At this phase word stemming algorithm is used to identify the tags with common stems, Figure 17. The assumption here is that tags are same if they have common stem. FolkSpace utilizes the well known Snowball library which provides various implemented stemming algorithms in Java programming language. The stemming is applied on whole collection of the tags, after stemming all tags with same stem are grouped into same

morphological group. For example tags “blog, blogs and blogging” have same stem “blog” therefore considered as morphologically similar.

The identification of synonymous tags heavily relies on the synsets of WordNet thesaurus. Given the collection of tags, the algorithm searches for synset in WordNet that contains each pair of tags in collection. If synset is found then synonymous group is created for corresponding pair of tags, and next algorithm tries to incrementally add new tags to group by checking if new tag is synonymous to all members of if synset.

Both algorithms for identifying morphological and synonymous groups have worked well when their implementations were tested on dataset. Please refer to Appendix for sample results from testing.

4.9.3 Statistical analysis phase

In this phase the implicit semantics in tag collection is further extracted by revealing the hidden relation among the tags. Given usual collection of tags as an input, this phase produces collection of tags with parent-child hierarchy in it. There is need to clarify when one tag becomes parent tag of another tag: “Tag A is a parent of Tag B if meaning of Tag A also includes the meaning of Tag B”. The example here can be parent-child relationship between tags “vehicle” and “car”. Because usually meaning of “vehicle” also assumes the meaning of “car”, these two tags can be modeled as “vehicle” being the parent of “car”.

Identification of such kind of hierarchy is difficult task and requires a complex algorithm. Therefore new approach based on combination of clustering and probabilistic analysis is proposed. Clustering algorithm applied here is also newly proposed and it is called “Cosine Similarity based Double Clustering Algorithm”. As it was explained in section of related works, directly applying the probabilistic analysis for building hierarchy in raw tag collection results in significant amount of noise (false relationship between two tags). Therefore in this work a new hierarchy analysis method is proposed which is aimed at minimizing the noise as much as possible. According to proposed approach the tag collection is first clustered to locate tags that are likely to be relevant to each other. Only then probabilistic analysis is applied on each cluster to build hierarchy within the cluster.

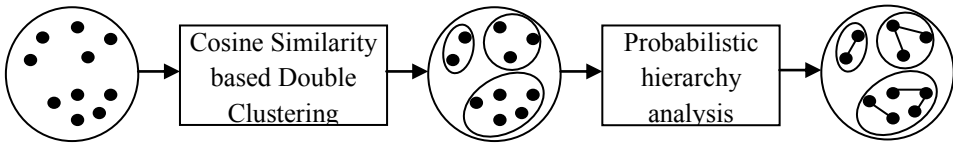


Figure 17. Hierarchy analysis through combination of clustering and probabilistic analysis.

The next sections will discuss details Cosine Similarity based Double Clustering Algorithm and probabilistic hierarchy analysis method.

4.9.4 Cosine Similarity based Double Clustering Algorithm

The idea behind the proposed algorithm is to apply cosine similarity based clustering two times, each time based on different criteria of similarity. The following types of similarity are identified between pair of tags:

- the similarity based on pattern of resources on which tags co-occur
- the similarity based on pattern of co-occurrence with other tags

The following assumptions are made on this approach:

1. Two tags with high similarity value are likely to be related to each other.
2. If two tags are tagging same set of resources then they are likely to be related to each
3. If two tags are co-occurring with same set of tags then they are likely to be related to each other.

The Figure 18 below shows the basic steps of algorithm. In Tag-Resource matrix each column represents the resource and each row represents the tag. Given any tag row, the values in that row are the frequencies of the tag in corresponding resources. If we consider each row as a vector representing the tag then it is possible to calculate the cosine similarity between tags using following formula:

$$\text{Cosine Similarity}_{ij} = \frac{\sum_{k=1}^{n} x_{ik} . x_{jk}}{\left(\sum_{k=1}^{n} x_{ik}^2 . \sum_{k=1}^{n} x_{jk}^2 \right)^{\frac{1}{2}}}$$

where x_i and x_j are vectors for different tags. By calculating cosine similarity between each pair of tags, it is possible to obtain the Tag-Tag matrix where each value represents the similarity between two pair of tags.

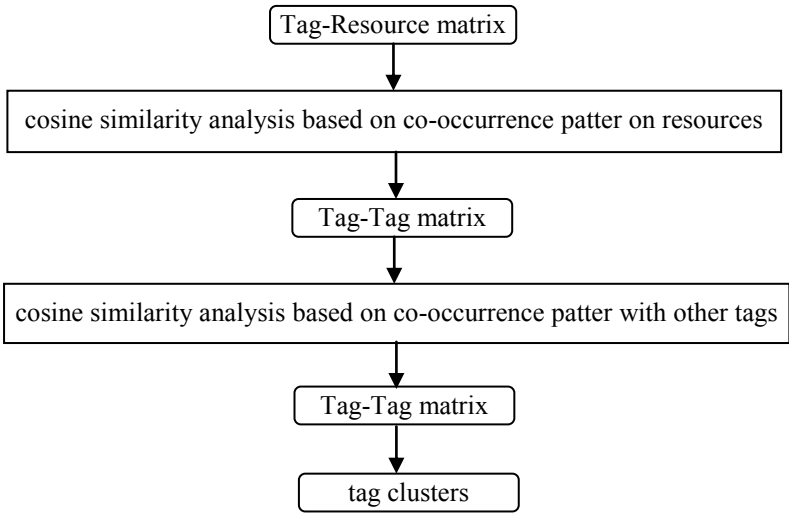


Figure 18. Basic steps in algorithm for deriving tag clusters

Now if we will consider the each row in Tag-Tag matrix then it will again represent the tag, but this time the tag is characterized in terms of similarity with other tags, not in terms of occurrence in resource annotations. As in previous time each row is treated as a vector and cosine similarity is calculated between each pair of tags. At the end it creates another Tag-Tag matrix with more precise similarity values than previous one. From similarity matrix all tags that have similarity value with each other above 0.8 are collected and formed into clusters.

4.9.5 Probabilistic analysis method

Now that tag model is divided into clusters, the same probabilistic analysis which was described in Related Works section is applied on each cluster separately. Applying probabilistic analysis on clusters rather than on whole tag collection decreases the chance of getting noisy relation, because tags in same cluster are very likely to have relation

between each other. As it was noted in Related Works, the optimal value of 0.8 was chosen as a threshold.

At the end of probabilistic analysis step each cluster has some elements of parent-child hierarchy.

4.9.6 Updating Tag Model

Hierarchy analysis step was one of the final steps within FOM Process. At the end of this step the tag collection from folksonomy contains not only description of tags but also identified semantics such as morphological and synonymous groups, and parent-child hierarchy.

The final steps within FOM Process involves to converting tag descriptions and identified semantics into RDF triples and storing those triples in Tag Model.

Chaper 5. Prototype Implementation

5.1 Prototype

To prove that proposed system and its design decisions are feasible, the prototype application was developed. The prototype include following functionalities and properties:

- Separate wrappers for two folksonomy-based applications which are Delicious and Flickr
- Complete FOM Ontology
- Subset of FOM Rules
- Complete FOM Processor with completely implemented FOM Process
- Two semantic models each describing 50 annotated resources
- Tag Model with about 2000 tag description
- SPARQL DL query interface

5.2 Folksonomy datasets

Two limited size folksonomy datasets were obtained from Delicious and Flickr applications using web wrappers. Delicious uses broad folksonomy, while Flickr uses narrow folksonomy. The folksonomy from each of the applications include 50 annotated resources and includes necessary information about users and tags. The combined dataset of two folksonomies contain around 2000 distinct tags. Both folksonomies were fed into prototype for building corresponding semantic models and Tag Model.

5.3 Prototype Architecture

The prototype application has architecture as it is define in Figure 19. It uses two wrappers to download small size folksonomies from Delicious and Flickr. Then those folksonomies are passed to FOM Processor. The FOM Processor utilizes Jena RDF library for managing and creating RDF triples. Also FOM Processor uses Pellet OWL DL Reasoner which is accessed through Jena interface.

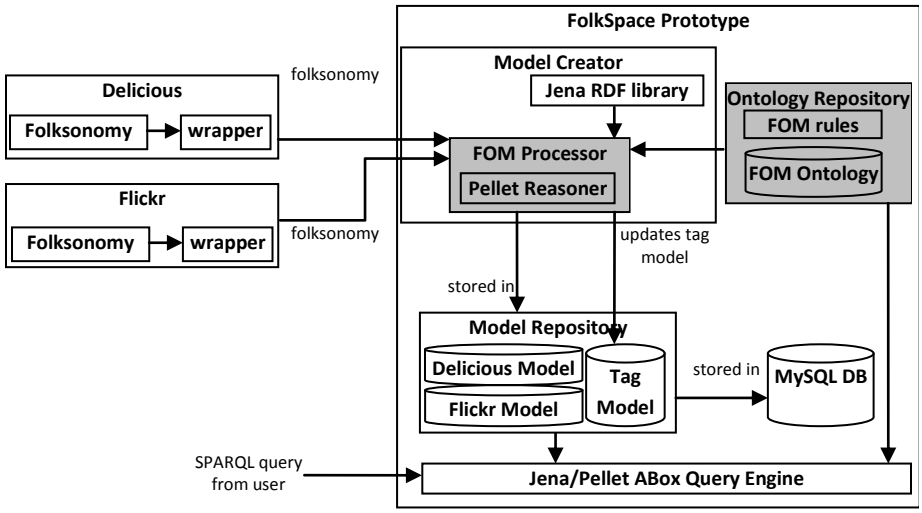


Figure 19. The prototype architecture.

FOM Ontology is created according to OWL DL grammar and serialized into XML structured file. Please refer to Appendix for complete XML description of FOM Ontology.

Although some FOM Rules are implemented, but not all of them are implemented. FOM Rules are created using SWRL/XML format following DL-safe grammar. This is due to fact that Pellet reasoner supports only DL-safe rules.

The whole Model Repository, including two folksonomy models and Tag Model, is serialized in MySQL database and accessed through Jena interface. The Model Repository is supported by Pellet ABox query engine, which is limited SPARQL query engine. Therefore it is possible to send SPARQL queries to Model Repository.

5.4 Sample results from semantic processing

During the processing of combined tag collection of two folksonomies the system was able to reveal several implicit semantics from the tags. Table 10 shows some example semantic relations that were identified.

Table 10. Examples semantic extracted from tag collection

Type of semantics	Tags involved
morphological group	education, educational, educate, educations, educator, educators, educated
morphological group	school, schools, schooling
morphological group	robotics, robot, robots
synonymous group	education, teaching, pedagogy, instruction, training
synonymous group	development, growth, evolution
synonymous group	studies, work, read, learn, report, study, discipline, subject, field, survey
same cluster	books, book, architecture, complexity, paradigm, e-text, download, cybernetics, libro, fractal, libros, booklist, pseudoscience
same cluster	resources, teaching, resource, general, curriculum, integration, lessons, plans, sites, plan
same cluster	noticias, old, news, health, echlin, archaeology, technet

As you can see from Table 10, the algorithms for finding morphological and synonymous groups worked quite well. The only problem with morphological grouping seems to be that it tends to group words of different parts of speech.

The clustering, although gives reasonable results, is not working as well as morphological and synonymous grouping. For example in first cluster the relationships between tags “book”, “archeology” and “cybernetics” are not quite obvious although they are in same cluster.

In overall the algorithms within the FOM Process for identifying the implicit semantics are working reasonably well.

5.5 User Interface for Querying

The user can send SPARQL queries to the prototype system using the UI shown in figure below.

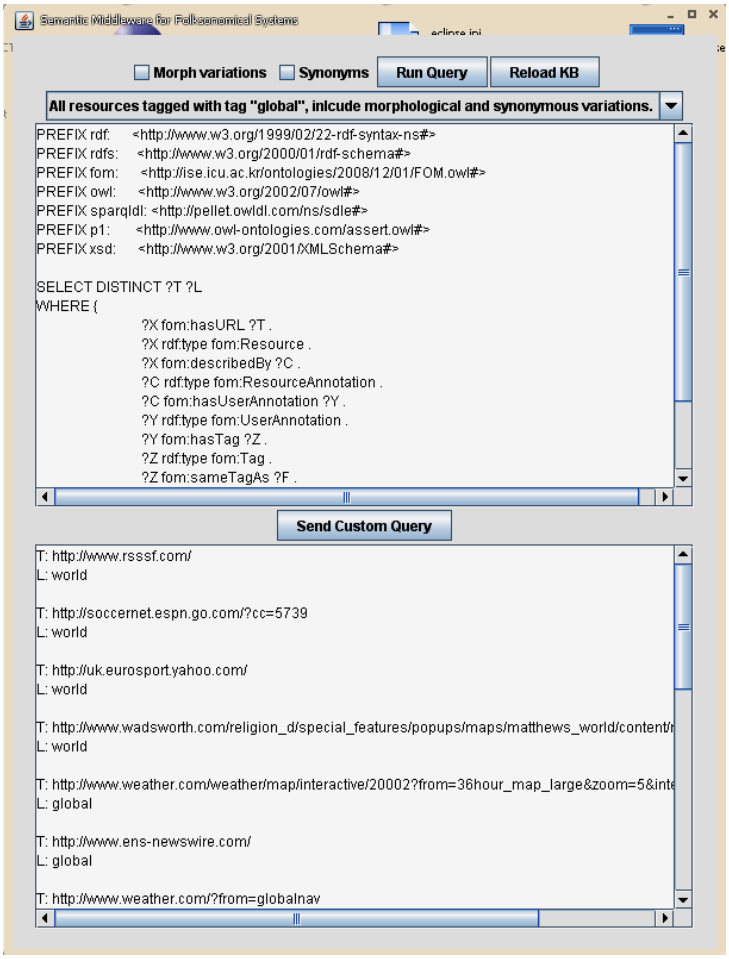


Figure 20. User Interface for sending queries to prototype.

In this UI user can send either some predefined set of queries or compose custom query and then send it. User can choose predefined queries from the combo box in upper side of interface. Or user can write custom queries in text area below the combo box. The query should be composed in valid SPARQL format and should be subject of restrictions of ABox query. The lowest text area shows the result returned to the query.

Chaper 6. Experimentation and Evaluation

The experimentation and evaluation was done on several levels: evaluation of FOM ontology, evaluation of the ability of the system to deal with ambiguity and synonymy, and evaluation of taxonomical reasoning abilities.

6.1 Evaluation of FOM ontology

The purpose of this evaluation is to prove that FOM ontology:

- can properly represent the semantics of real life tagging data
- can support interoperability among separate systems

For this purpose the data was gathered from two separate collaborative tagging services Flickr and Delicious. The tagging data from both services was formatted and stored according to FOM ontology model.

Table 11. Test data properties

Service	Flickr	Delicious
# of annotated resources	10	20
Average # of tags per annotation	7	16
Average # of users per annotation	1	23

The prototype agent was developed to query the data from both services using SPARQL queries. Below is the list of the queries:

- retrieve all instances of users belonging to only service X.
- retrieve all resources annotated within service X.
- retrieve all resources tagged with tag X and belonging to service Y.
- retrieve all resources tagged with tag X (no discrimination by service).
- retrieve all resources tagged with tag X by user Y and belonging to service Z.

All queries were successfully accomplished by the system. The system was able to successfully differentiate between different tagging services, but also allowed cross-service querying when it was needed. It proves the interoperability capabilities of FOM ontology.

Also the system was able to understand the semantics of tagging such as relationships between users, tags, resources and services.

6.2 Evaluation of linguistic capabilities of the system

The purpose of this evaluation is to assess the capability of the system to overcome linguistic limitation of the folksonomies during search and retrieval. The system is expected to recognize in utilize following linguistic features during the search and retrieval:

- synonymy
- morphological variations

In order to perform this evaluation the number of synsets was obtained from WordNet. The example of the synset obtained from WordNet: {car, auto, automobile, machine, motorcar, railcar, railway car, railroad car}. For each word contained in the synsets the annotations from Delicious and Flickr were obtained. The tagging data from both services was formatted and stored according to FOM ontology model, with lexicographical analysis done prior to it.

The prototype agent was developed to query the data from both services using SPARQL queries. Each word contained in the synset was send as a part of following query:

- retrieve all resources tagged with tag X.

The expected result of this query is to retrieve all resources tagged with tag X or with its synonyms or morphological variations. The experiment was a total success except some minor cases. The system was able to correctly identify all resources tagged with tag X and its semantic equivalence tags. In very few cases there were wrong resources returned due to incorrect identification as a morphological variations. In further analysis showed that system was confused due to similarity in the stems of the tags. Since such error happens very rare the implications are negligible.

6.3 The taxonomical reasoning capability of the system

The only way to assess the quality of the produced taxonomy is to compare results of search supported by taxonomy reasoner and usual tag-based search.

For this purpose the test data was obtained from Flickr containing annotation of 100 resources. To ensure the relation between resources, the top 25 annotations were chosen

from search results with five different query words: education, health, science, sport. The test data was formatted according to FOM ontology model, and statistical analysis was done over it to derive taxonomical hierarchy.

For the purpose of the experiment the prototype software agent was developed that can query the annotation in two modes: simple keyword based querying used in Flickr; taxonomical reasoner enhanced querying.

Three independent testers were asked to use the agent to provide random queries using both modes and then assess the relevance of retrieved resources to user query. Only restriction on the query was to be provided it within the context of education, economy, health, science or sport.

Table 12. Comparison of reasoner enhanced and ordinary searches

	average precision in usual search	average precision in reasoner supported search	average # of results in usual search	average # of results in reasoner supported search
User A	47%	54%	6	13
User B	51%	60%	7	12
User C	43%	44%	9	14

*The cases in which both modes gave 0 results are ignored.

If we look at the results provided in the table then we can see that reasoner supported search gives only slightly higher precision than usual search. But it also should be considered that reasoner retrieved more results because it considers more tags than was provided in the query due to subsumption hierarchy. Therefore the number of relevant results retrieved by it is quite higher than number of relevant results retrieved by usual search. Therefore we can see that taxonomical reasoner improves the quality of search by reasonable amount. From above result, it is possible to conclude that system was able to generate taxonomical hierarchy is with acceptable quality.

6.4 Overall Evaluation

In the beginning of this thesis, in Section 1.3 several success criteria were established against which the success of this work should be evaluated. Totally two success criteria were establish each of which had several requirements.

I claim that first success criteria was fully met by the implemented prototype. First of all, the prototype implemented all main components of the proposed systems. Second, prototype was provided with two folksonomies from two different applications and was able to successfully process those folksonomies into semantic models. Third, the implemented query interface was able to successfully send SPARQL queries to Model Repository and receive desirable results. One thing that should be discussed here is that because not all FOM Rules were implemented, the length of SPARQL query which user have to write increased significantly. Although it can be quite inconvenient for user to write lengthy query, the problem can be avoided in fully implemented application if all FOM Rules are defined.

In terms of second success criteria it is satisfied only partially. The hierarchy analysis algorithm worked not as good as expected. There was still noise both in created clusters and identified parent child relations. In other side, algorithms created for extracting morphological and synonymous groups worked quite well and gave satisfying results. The reasoner was able to consider identified semantic relationships between tags in answering the user query. For example when user requested all resources tagged with tag “globe”, the reasoner was able to expand the original result set with results which also had tags “world” and “globalization”, because those tags were described in Tag Model as synonymous.

Chaper 7. Conclusion

Although most of the objectives that were set at the beginning of this research were met, at current stage it cannot be considered as complete. The prototype have proven the proposed system architecture and processes to be feasible, but during the implementations several flaws were identified. Some of the proposed algorithms were difficult to implement because of restriction in development environment. Because FolkSpace is required to process and manage big amounts of data, there have been significant problems in finding optimal points between processing time and memory consumptions.

This was especially the case for hierarchy analysis algorithms. During the implementation of this algorithm in-memory based processing was proven to be impractical because of large amount of memory required for loading similarity matrixes. Instead storage based algorithm was implemented which significantly increased processing time due to frequent IO.

Other problems were identified with FOM Ontology. Current FOM Ontology results in large amount of RDF triples. The ultimate goal in FolkSpace should be decreasing number of RDF triples as much as possible, because it was proven that querying time of RDF triples increases significantly with the size of dataset. Therefore FOM Ontology classes and relations between those classes should be optimized in order to minimize number of RDF triples as much as possible.

FOM Rules have same problems as FOM Ontology. When rules were supplied to reasoner, it generated huge amount of statements. Again decreasing number of statements without losing semantics is a priority here. Currently two solutions are considered for solving this problem. First, localized reasoning can be implemented where reasoner considers only those statements which are related to query. Reasoner applies rules only on those statements instead of whole collection. Second, it is possible to implement rule-based query extension which produces extends original query with additional statements which are derived from application of rules on query. This approach also can solve the problem of requiring to write complex queries for sophisticated results.

Also revised system architecture should be considered as a new research priority, where semantic models of folksonomy are stored locally in folksonomy-based applications, but refer to common Tag Model in separate server. This architecture might be much more flexible than the one used in this research.

Overall results of this research are very promising, and future work should be done with respect to points mentioned above.

Summary

비록 비교적 새로운 기술, folksonomy 가 입증된 유용합니다. Folksonomy 가 콘텐츠 관리를위한 저렴하고 효과적인 방법입니다. Folksonomy 서비스를 제공하기위한 중요한 도구가 된 일부 시스템 있음. 애플 리케이션을 사용하는 folksonomy 의 수는 급격히 증가하고있다.

이 같은 추세가 환경의 비전은 인간 사용자와 더 중요한 것은 소프트웨어 요원 쿼리, 검색 및 엔드에서 이질적인 자원을 검색할 수있습니다 - 포인트를 간단하고 의미있는 방식으로된다. 자신의 네트워크 환경이 - 커뮤니티를 생성하는 조직, 주석과 동적 콘텐츠를 공유하고있다는이 비전을 현실로 folksonomy - 기반 시스템을위한 핵심 기술. 하지만 아직 어떻게 주석 folksonomies 하여 효율적이고 생산적인 방식으로 생산에서 지렛대로의 명확한 이해를합니다. 이 문제를 복잡, folksonomy - 기반의 애플 리케이션 사이의 상호는 여전히 큰 문제입니다. 같은 시간에 folksonomy 구조의 부족은 또한 유용성에 한계가있다.

FolkSpace a 미들웨어 또는 위에서 설명한 문제를 해결하기 위해 설계는 플랫폼입니다. FolkSpace 뿐만 아니라 의미 론적 구조에 필요한 많은 folksonomy 에 추가하려고 서로 다른 시스템에서 folksonomies 을 통합 액세스를 제공합니다. FolkSpace 무겁게 플랫폼에 의존하여 '시맨틱웹'에 상호의 현대적인 기준은 다음과 같은 독자는 XML 형식. 반대편 FolkSpace 에서 표준화된 쿼리 언어 folksonomies 에 쉽게 액세스할 수 있지만 SPARQL 을 제공합니다.

References

[1] "An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval", Castells P., Fernandez M., Vallet D., IEEE Transactions on Knowledge and Data Engineering, Vol. 19, No. 2, February 2007

[2] "Automated Tag Clustering: Improving Search and Exploration in the Tag Space", Begelman G., Keller P., Smadja F., WWW2006, May 22–26, 2006, Edinburgh, UK

[3] "Bridging the Gap Between Folksonomies and the Semantic Web: An Experience Report", Angeletou S., Sabou M., Specia L., Motta E., *Bridging the Gap between Semantic Web and Web 2.0, SemNet*, pp. 30-43., 2007

[4] "Clustering Tags in Enterprise and Web Folksonomies", Edwin Simpson, International Conference on Weblogs & Social Media, Seattle, March 31st, 2008

[5] "Collaborative Creation of Communal Hierarchical Taxonomies in Social Tagging", Heymann P., Garcia-Molina H., Stanford InfoLab Technical Report 2006-10

[6] "Enriching Ontological User Profiles with Tagging History for Multi-Domain Recommendations", Cantador I., Szomszor M., Alani H., Fernandez M., Castells P., *1st International Workshop on Collective Semantics: Collective Intelligence & the Semantic Web (CISWeb 2008)*, 2 June, 2008, Tenerife, Spain.

[7] "FolkRank: A Ranking Algorithm for Folksonomies", Hotho A., Jaschke R., Schmitz C., Stumme G., In Proc. FGIR 2006, 2006

[8] "Folksonomies – Cooperative Classification and Communication Through Shared Metadata", Adam Mathes, http://www.adammathes.com/academic/computer-mediatedcommunication/folksonomies.html, 2004

[9] "FolksOntology: An Integrated Approach for Turning Folksonomies into Ontologies", Damme C.V., Hepp M., Siorpaes K., In Proceedings of the ESWC Workshop "Bridging the Gap between Semantic Web and Web 2.0", 2007, http://www.heppnetz.de/files/vandammeheppsiorpaes-folksontology-semnet2007-crc.pdf

[10] "From Folksologies to Ontologies: How the Twain Meet", Spyns P., Moor A., Vandenbussche J., Meersman R., On the Move to Meaningful Internet Systems 2006: CoopIS, DOA and ODBASE (OTM2006), Montpellier, France: Springer, 2006.

[11] "Harvesting Social Knowledge from Folksonomies", Wu H., Zubair M., Maly K., In Proc. of HYPERTEXT '06, August 22-25, 2006, Odense, Denmark

[12] "Improving Tag-Clouds as Visual Information Retrieval Interface", Hassan-Montero Y., Herrero-Solana V., International Conference on Multidisciplinary Information Sciences and Technologies, InSciT2006. Mйrida, Spain. October 25-28, 2006

[13] "Inducing Ontology from Flickr", Schmitz P., In Proc. of the Collaborative Web Tagging Workshop at WWW 2006, May 22–26, 2006, Edinburgh, UK

[14] "Information Retrieval in Folksonomies: Search and Ranking", Hotho A., Jaschke R., Schmitz C., Stumme G., In Proc. of ESWC'06, 2006

[15] "Web Page Recommender System based on Folksonomy Mining", Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on (2006), pp. 388-393

[16] "Measuring the Semantic Value of Folksonomies", Al-Khalifa S. H., Davis C. H., In: *the Second International IEEE Conference on Innovations in Information Technology*, November 17-21, Dubai, UAE, 2006

[17] "Metadata Mechanisms: From Ontology to Folksonomy … and back", Stijn Christiaens, On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops (2006), pp. 199-207

[18] "Mining Association Rules in Folksonomies", Schmitz C., Hotho A., Jaschke R., Stumme G., Data Science and Classification: Proc. of the 10th IFCS Conf., Ljubljana, Slovenia, July, Springer, 2006

[19] "Ontologies are us: A Unified Model of Social Networks and Semantics", Peter Mika, In Proc. of ISWC'05, 2005 and Journal of Web Semantics, Elsevier, Volume 5, Number 1, p.5--15 (2007)

[20] "Ontology of Folksonomy: A Mash-up of Apples and Oranges", Thomas Gruber, Published in Int'l Journal on Semantic Web & Information Systems, 3(2), 2007.Originally published to the web in 2005.

[21] "Ontology of Folksonomy: A New Modeling Method", Echarte F., Astrain J. J., Cordoba A., Villadangos J., In: Semantic Authoring, Annotation and Knowledge Markup (SAAKM 2007), Whistler, British Columbia, Canada, October 28-31 (2007)

[22] "Tag Ontology design", http://www.holygoat.co.uk/projects/tags/

[23] "P-TAG: Large Scale Automatic Generation of Personalized Annotation TAGs for the Web", WWW 2007 *: Proceedings of the 16th International Conference on World Wide Web,* pp 845-854, May 8–12, 2007, Banff, Alberta, Canada

[24] "RelExt: A Tool for Relation Extraction from Text in Ontology Extension", Schutz A., Buitelaar P., In Proceedings of the 4th International Semantic Web Conference (ISWC), Galway (2005), pp 593-606

[25] "Semi-Automatic Ontology extension Using Spreading Activation", Liu W., Weichselbraun A., In Journal of Universal Knowledge Management, vol. 0, no. 1 (2005),pp 50-58

[26] "Semiotic Dynamics and Collaborative Tagging", Cattuto C., Loreto V., Pietrinero L., Proceedings of the National Academy of Sciences 104(1461), 2007

[27] "SNS Environmental Vocabulary – from Terms to Ontology", Ruther M., Bandholtz T., Menger M., Semantics 2006. Wien, 28.-30.11.2006

[28] "Extreme Tagging: Emergent Semantics through the Tagging of Tags", Tanasescu V., Streibel O., *Proceedings of the International Workshop on Emergent Semantics and Ontology Evolution (ESOE2007) at ISWC/ASWC2007* (2007)

[29] "The Complex Dynamics of Collaborative Tagging", Halpin H., Robu V., Shepherd H., WWW 2007, May 8.12, 2007, Banff, Alberta, Canada

[30] "The Structure of Collaborative Tagging System", Golder S.A., Huberman B.A., HPL Technical Report, 2005

[31] "Towards the Semantic Web: Collaborative Tag Suggestions", Xu Z., Fu Y., Mao J., Su D., WWW2006, Edinburgh, UK.

[32] "Using the Semantic Web as Background Knowledge for Ontology Mapping", Sabou M., d'Aquin M., Motta E., In Proc. of the Int. Workshop on Ontology Matching (OM-2006), 2006

[33] "Folksonomy", Thomas Vander Wal, Online Information 2005, London, UK, Dec 30th 2005

[34] www.delicious.com

[35] www.flickr.com

[36] “Ontologies are us: emergent semantics in folksonomy systems”, Semantic Web and Beyond, Social Networks and the Semantic Web Vol. 5, pp.193-207, Springer US, 978-0-387-71000-6, 2007

[37] “Emergent Semantics from Folksonomies: A Quantitative Study”, Zhang L., Wu X., Yu Y., Journal on Data Semantics VI (2006), pp. 168-186.

Acknowledgement

I would like to express my sincere appreciation to my advisor Prof. Ho-Jin Choi for his advices and support. I would also like to thank my thesis committee members, Prof Ho-Jin Choi, Prof. Gwan-Su Yi, and Prof. In-Young Ko for their support and their comments.

Curriculum Vitae

Name: Enkhbold Nyamsuren
Date of Birth: September 1st 1985

Education

2007-2009	KAIST (M.S.)
2002-2006	Huree University of ICT (B.S.)

Career

2007~	Research Assistant, GRID Middleware Center (GMC), Intelligent Software Engineering and Robotics Lab, KAIST, Korea
2006-2007	IT Professional, Khan Bank of Mongolia, Ulaanbaatar, Mongolia
2003-2004	Project Assistant, Member of JMITA/UN ICT Students Volunteer team, Japan Mongolian IT Association, Mongolia

Academic Activities

List of publications:

1. Enkhbold Nyamsuren and Ho-Jin Choi, “Building a Semantic Model of a Textual Document for Efficient Search and Retrieval”, The 12th International Conference On Advanced Communication Technology (ICACT 2009), 15-18 February 2009, Phoenix Park, South Korea, ISBN 978-89-5519-139-4, pp.298-302
2. Enkhbold Nyamsuren and Ho-Jin Choi, "Building Domain Independent Ontology For Web 2.0", IEEE 8th International Conference on Computer and Information

Technology (CIT 2008), 8-11 July 2008, Sydney, Australia, ISBN: 978-0-7695-3242-4, pp.655-660.

3. Enkhbold Nyamsuren and Ho-Jin Choi, "User Feedback Based Web Software Engineering", APIS 2008: The 7th International Conference on Applications and Principles of Information Science, Jan 28-30, 2008, Auckland, New Zealand, ISSN: 1976-7587, pp.558-562.
4. Enkhbold Nyamsuren and Ho-Jin Choi, "A Model of Code Reusability for Mobile Robots", APIS 2008: The 7th International Conference on Applications and Principles of Information Science, Jan 28-30, 2008, Auckland, New Zealand, ISSN: 1976-7587, pp.73-76.
5. Enkhbold Nyamsuren and Ho-Jin Choi, "Preventing Social Engineering in Ubiquitous Environment", Future Generation Communication and Networking (FGCN), December 6th ~ 8th, 2007, Jeju, Korea, ISBN: 0-7695-3048-6, Vol 2, pp. 576-580.
6. Enkhbold Nyamsuren and Ho-Jin Choi, "Addressing Privacy issue in Pervasive Environment Through the Use of Composite Approach", International Conference on Mobile Computing, Communications, and Applications (ICMOCCA `07), Sep 25~29, 2007, Tashkent, Uzbekistan, ISBN 978-89-958425-1-5, pp. 223-227.