TECHNISCHE UNIVERSITÄT MÜNCHEN
FAKULTÜT FÜR INFORMATIK
Forschungs- und Lehreinheit XI
Angewandte Informatik / Kooperative Systeme

A Framework for Working with Cross-Application Social Tagging Data

Walter Christian Kammergruber

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Helmut Krcmar
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. Johann Schlichter
	2.	Univ.-Prof. Dr. Florian Matthes

Die Dissertation wurde am 26.06.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 26.11.2014 angenommen.

Zusammenfassung

Mit dem zunehmenden Erfolg des Web 2.0 wurde und wird Social-Tagging immer beliebter, und es wurde zu einem wichtigen Puzzle-Stück dieses Phänomens. Im Unterschied zu ausgefeilteren Methoden um Ressourcen zu organisieren, wie beispielsweise Taxonomien und Ontologien, ist Social-Tagging einfach einzusetzen und zu verstehen. Bedingt durch die Einfachheit finden sich keine expliziten und formalen Strukturen vor. Das Fehlen von Struktur führt zu Problemen beim Wiederaufinden von Informationen, da beispielsweise Mehrdeutigkeiten in Suchanfragen nicht aufgelöst werden können. Zum Beispiel kann ein Tag „dog“ (im Englischen) für des Menschen bester Freund stehen, aber auch für das Lieblingsessen mancher Personen, einem Hot Dog. Ein Bild einer Katze kann mit„angora cat“, „cat“, „mammal“, „animal“oder „creature“getagged sein. Die Art der Tags hängt sehr stark vom individuellen Nutzer ab. Weiterhin sind Social-Tagging-Daten auf verschiedene Applikationen verteilt. Ein gemeinsamer Mediator ist nicht vorhanden. Beispielsweise kann ein Nutzer auf vielen verschiedenen Applikationen Entitäten taggen. Für das Internet kann das Flickr, Delicious, Twitter, Facebook and viele mehr sein. Innerhalb großer Firmen, wie Siemens, werden viele verschiedene Social-Software-Applikationen verwendet, wobei die meisten eine Form des Tagging verwenden. Ein gemeinsamer Mediator fehlt. Social-Tagging-Daten werden nicht wiederverwendet.

In dieser Arbeit wird ein Framework beschrieben, mit dem sich Social-Tagging-Daten von verschiedenen Applikationen aggregieren lassen. Zudem werden die Daten mit Relationen angereichert. Ein einfaches und generisches Datenaustauschformat wurde definiert, ein Aggregationsmechanismus entwickelt. Das Tagging-Framework speichert die Tag-Daten zwischen und stellt den angeschlossenen Applikationen Services zur Verfügung. Das Verwalten der Daten und die Kontrolle verbleibt bei den jeweiligen Applikationen.

Drei verschiedene Ansätze, die zu leichtgewichtigen Strukturen führen, wurden gestaltet: Statistische Analysen von Social-Tagging-Daten, Abbildung von Tags auf vorhandene Ontologien oder anderen strukturierten Quellen und ein kollaborative Thesauruserzeugung. Weitere Data-Mining-Anwendungen wurden ausgewählt und angepasst. Dies beinhaltet das Clustern von Nutzern anhand ihrer Social-Tagging-Praxis und die verbesserte Verteilung von Fragen in einer Question-And-Answer-Plattform.

Teile der Implementierung des beschriebenen Frameworks befinden sich seit geraumen im Produktiveinsatz im Intranet der Siemens AG. Als ein Ergebnis konnte die Anzahl der internen Email-Notifikationen für ein Siemens Social-Media-Werkzeug um ein Faktor von mehreren Hunderttausend pro Monat reduziert werden.

abstract

Social tagging has become very popular with the rise of Web 2.0, making it an important piece of the puzzle that forms this phenomenon. In contrast to more elaborate ways of organizing resources, such as taxonomies or ontologies, tagging is very easy to use and understand. Because of its simplicity tagging does not create explicit, formalized structures. The lack of structure leads to problems in information retrieval, since for example ambiguities in queries cannot be resolved. For example, a tag “dog” can stand for a man’s best friend or for some people’s favorite food, a frankfurter. A picture of an angora cat can be tagged e.g. with “angora cat,” “cat,” “mammal,” “animal,” or “creature”. The choice of tag strongly depends on the individual user. Further, social tagging data is spread across many different applications. For example, a user can tag entities in many different applications. In the case of the Internet this can be Flickr, Delicious, Twitter, Facebook, and many other. Inside a larger corporation, such as Siemens, there are many social software tools in use. Most of them use some kind of tagging mechanism. A common mediator is here missing. Social tagging data is not reused across application boundaries.

This thesis describes a framework for aggregating social tagging data from different applications. An approach for enriching its social tagging data with semantic relations has been developed. For fetching social tagging data a simple and generic data exchange format has been created. For retrieving social tagging data from different sources, a mechanism has been designed. The tagging framework caches social tagging data and only acts as a service for associated applications. The actual storage and control over the tagging data stays with the individual application.

Three different methods that lead to lightweight structures have been developed: statistical analysis of social tagging data, mapping tags to existing ontologies or other structured input and collaborative tag thesaurus creation. Further data mining applications have been selected and adopted. This includes clustering of users based on their social tagging practice and improved channeling of requests in a question and answer platform.

Parts of the implementation of the described framework are in productive use within the Intranet of Siemens AG. As one result, internal email notifications for a Siemens social media tool could be reduced by the factor of hundreds of thousands emails per month.

Acknowledgements

Many people have helped make this research possible; adequately acknowledging them has been one of the most challenging writing tasks of this dissertation.

Above all, I would like to thank to Prof. Dr. Johann Schlichter, my supervisor at the Technical University of Munich. Prof. Schlichter was a guiding mentor that helped me a lot in leading this work in the right and fruitful direction. Additionally, the fellow PhD students and assistant teachers (Dr. Wolfgang Wörndl and Dr. Georg Groh) at the Technical University of Munich where very helpful with their constructive feedback. Especially, the doctoral seminars were highlights during the time working on this research.

This thesis has been done in cooperation with Siemens Corporate Technologies in Munich. I would like to acknowledge the support provided by my colleagues at Siemens Corporate Technologies, especially Dr. Manfred Langen and Dr. Karsten Ehms but also Werner Zucker, Dr. Maximilian Viermetz, Bernd Lindner and Dr. Albert Eckert for their support and inspiration.

I would like to express my gratitude towards Mike Burgold, who has helped a lot in transforming a prototypical implementation of a tagging framework developed within this thesis into a production ready implementation.

Furthermore, I wish to thank to thank Prof. Dr. Jiří Panyr, who departed from us too soon, for some interesting discussions and literature recommendations. Feedback given by Dr. Isabella Peters on information science topics has been a great help in refining the corresponding chapter. I would like to express my gratitude to Dr. Axel Rauschmayer for his advice on structural aspects of my dissertation and tips on writing styles.

Additionally, I would like to thank the team related to the Siemens TechnoWeb for their agile, friendly, and professional cooperation: Dr. Michael Heiss, Thomas Mayerdorfer, Clemens Wiener, and Dr. Thomas Lackner.

This PhD thesis has been created within the framework of the Theseus project, more precisely the Use Case Alexandria. The project was funded by the German Federal Ministry of Economy and Technology under the promotional reference “01MQ07012”.

Contents

CHAPTER 1

Introduction

Order is heaven's first law.

— Alexander Pope (1688 – 1744)

Contents

1.1. About Social Tagging

Over the last few years, social tagging has become a very popular tool for categorizing knowledge items [MNBD06], [Mat04]. Basically, a tag (also referred to as keyword or label) is a textual annotation that can be attached to any kind of resource. The resource can be a picture, a bookmark, a blog post, a wiki page and in general everything that can be referenced in form of some kind of identifier, in the web context typically an URL. For example, figure 1.1 shows a picture that has been taken in San Francisco (CA). In a photo community site tags, such as *usa*, *san-francisco*, *morning*, *bw*, *mission-street*, are used to annotate that picture. Based on these tags, the owner and other members of the community can search for and access the picture. By allowing individuals to categorize and mark their own data as well as data of others, an overview of content with some personal relevance can be won – hence the "social," otherwise one speaks of manual, plain keyword indexing.

Figure 1.1: *Picture taken in San Francisco. Tagged with:* ***usa***, ***san-francisco***, ***morning***, ***bw***, ***mission-street***. *(Source: Author)*

Social tagging has been hyped since about 2005 and has now become some kind of de facto standard for categorizing resources on the Internet. Delicious[1] has been one of the first real platforms where social tagging has been used for categorization. It is a very simplistic bookmarking which has evolved out of a hobby project of Joshua Schachter. At first, Schachter created a simple service which allowed him to share his bookmarks with his friends. Additionally, he wanted to have access to his bookmarks from different computers. The idea behind it was that anyone could create an account and bookmark a page via a bookmarklet[2] and categorize it via tags.

Schachter mainly has chosen tags for categorization since tagging is very simple to use. Additionally, there is no need to predefine categories. The service became very popular and Delicious was bought by Yahoo! in 2006. There has been a re-design of the user interface and plugins for all popular browsers have been published. In 2011, Delicious has been sold to Avos – a startup supported by the founder of YouTube – during the cleanup of Yahoo!s services. The user interface has been modernized again and Avos has integrated new features, such as profile pictures and new overview pages.

There are many other Internet services using tagging for organizing their information

[1] `http://www.delicious.com/`

[2] A bookmarklet is a JavaScript snippet that is stored as bookmark in a browser. By clicking the bookmark the code is executed in the context of the current page. This is for example useful in the case for Delicious to get the URL of the current page.

items[3]. An example is Flickr[4], which is a very popular photo platform with millions of users. Tags can be applied to photos by the owner of it.

Last.fm[5] is a music community where tags are used to categorize music and artists. The terminology in the music domain is very subjective. Having the user community apply tags leads to a consensus for the terminology to use for classifying musicians or songs. 43things[6], a community around life goals, has found its niche in the cloud of community and social software platforms. Tags are used to categorize texts that describe tasks that people want to achieve in their life, such as learn to play guitar, travel to Paris or learn Spanish. CiteULike and Bibsonomy [7] focus on researchers. Both provide a service for organizing scholarly references. Literature collections can be exported to various formats, such as BibTex or EndNote. Tags are used to categorize literature items.

With the advent of social software as knowledge management tools in companies – sometimes referred to as enterprise 2.0 [McA06] – organizing tagged information on a larger scale and also in a professional working environment has become an important issue. Organizing in this context means creating some *structures* which provide a certain degree of stability to tagging activities. Technically, this translates into the need to find (hidden) structures in sets of tag in order to be able to work with large sets of tagged entities more efficiently. In this introductory chapter only short introduction of social tagging is provided. An in depth investigation of the characteristics of social tagging is elaborated in chapter 2.

1.2. Three Major Social Software Applications at Siemens

Inside Siemens, there are many platforms that employ tags for annotating resources. At the time of this research there are three major social software applications that use tags inside the Siemens intranet. The banners of these three social software applications are shown in figure 1.2. Theoretically, every employee of Siemens can access these applications, which is more than three hundred thousand people. Of course the number of people participating on these platforms is much less. This is due to various reasons, such as the lack of an Internet connection, because somebody is working in production or persons that are not aware of these tools.

Siemens has a blogging platform (called Siemens Blogosphere) where each Siemens employee can maintain his or her own personal blog or contribute to group blogs [Ehm10]. Additionally, Siemens has an open editable Wiki platform based on Atlassian Confluence named Wikisphere [Lin08]. This platform can be used for collaborative knowledge exchange. There are no access restrictions. Every Siemens employee can edit and create pages. A recently introduced respectively modernized social software application is Tech-

[3] An information item is used in this work to refer to an entity or object that has a certain inherent information that can be accessed and processed in some way by a machine or a human being. This can be nearly anything, for example a text snippet as well as a picture or a music file.

[4] `http://www.flickr.com` – owned by Yahoo!

[5] `http://www.Last.fm`

[6] `http://www.43things.com`

[7] `http://www.bibsonomy.org/`, `http://www.citeulike.org/`

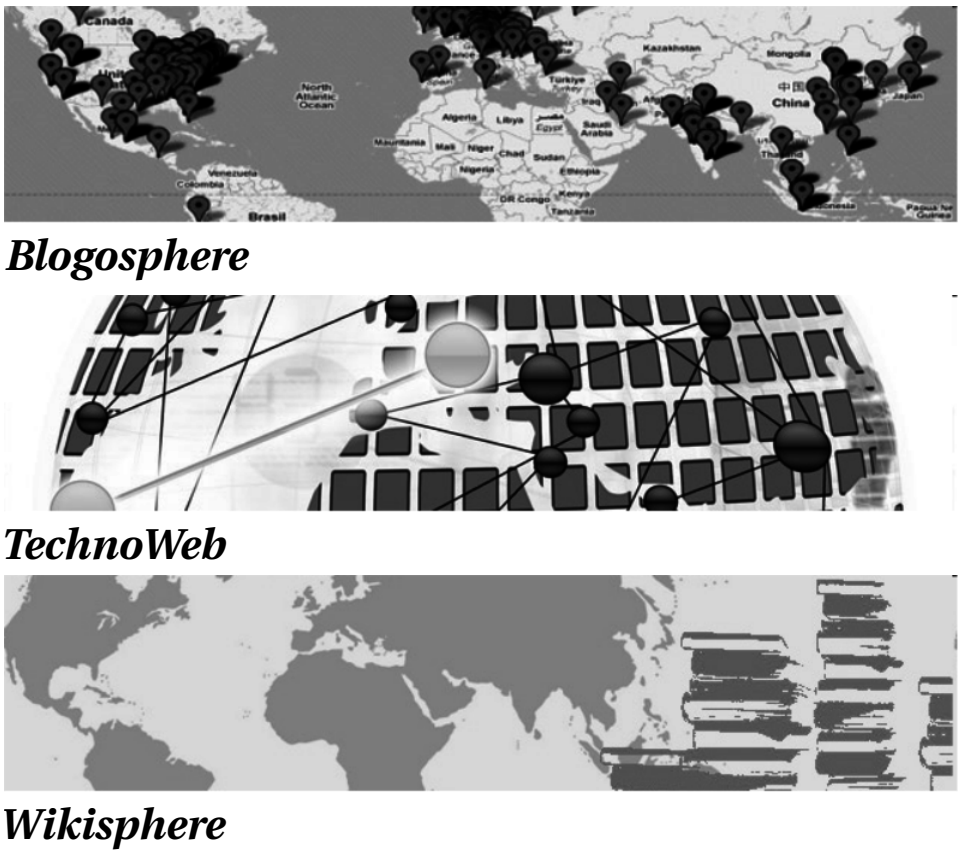


Figure 1.2: *Banners of three major social software applications at Siemens: Blogosphere, TechnoWeb, Wikisphere.*

noWeb [DPH10]. The aim of TechnoWeb is to create networks of people and topics in order to enhance knowledge exchange between people that are organizationally or geographically separated inside the company.

Blogosphere

Siemens Blogosphere, a corporate blogging platform, was introduced in 2006 as part of the Siemens intranet portal (the start page for the Siemens intranet) [Ehm10]. Every employee is allowed to have his or her own weblog (or short blog). In general, the blogging platform has been introduced to foster knowledge building and sharing of information between employees and extend dialog oriented internal communication practices. Blogs are a rather new form of communication targeting at creating a simple form of knowledge base. Complex processes are avoided and every employee can contribute easily. For example simple day to day problems, such as having problems with certain Microsoft products or internal Siemens tools can be discussed and a solution shared.

Every published blog post is accessibly inside the Siemens intranet without any special permission. For contributions a user has to be logged in. Blog post can only be tagged by

its author. The Siemens Blogosphere is therefore a narrow folksonomy (see the definition by Vander Wal in 2.2.2).

Ehms investigates blog usage patterns [Ehm10]. He identifies the following intended purposes of personal blogs:

- Advocating a topic that an individual blogger thinks is important and increase the general awareness for it.
- Receiving feedback on personal ideas.
- Receiving feedback on personal opinions.
- Convey tips and tricks.
- Exchange of experience and networking.
- Creating a personal knowledge base.
- Creating a knowledge base in order to target certain groups of people.
- Solely an experimental approach in order to find a purpose.

Considering the different and heterogeneous usage patterns for personal blogs, a rather heterogeneous tagging practice can be assumed. Incentives and motivations for users to annotate resources with tags as described by Marlow et al. [MNBD06] (see 2.2.2) are highly depending on the purpose of a personal blog. If someone wants to create attention for his or her post, she or he uses more general tags and also provide alternative tags that are used synonymously (for example "sustainable_city," "green_city," "eco-city") and spelling variants. In contrast, if someone is only blogging for personal usage, such as note taking, he or she typically uses less tags. Furthermore, only tags reflecting a personal vocabulary are used.

Additionally, to the personal blogs, there are several group weblogs. For example, some are posting announcements for certain project groups, some provide IT tips and some discuss limited topics such as knowledge management. The assigned tags in group blogs are rather general and contain various synonyms.

Wikisphere

The Siemens Wikisphere is an enterprise wiki platform (based on Atlassian Confluence[8]) that can be accessed and modified by every Siemens employee inside the Siemens intranet [Lin08]. Having a common wiki for all sectors of Siemens enables the creation of a common knowledge base – a Wikipedia for topics concerning Siemens and its employees.

Although Confluence supports the hierarchical categorization of wiki pages into spaces Wikisphere makes no use of this feature. The number of spaces needed for a serious usage of this feature would have been too large. More spaces lead to more administrational

[8] `http://www.atlassian.com/software/confluence/overview`

overhead and is orthogonal to the idea behind a wiki as an open platform. There are only two spaces for German and English context. The Wikisphere strongly relies on tagging and cross-linking of pages for navigating its content. Having tagging as a way for organizing information objects leads to all the advantages discussed in chapter 1.

In order to provide some kind of view on pages belonging to a certain topic, a concept named topic portals has been developed for the Wikisphere. A topic portal is analogues to portals known from Wikipedia[9]. In contrast, to Wikipedia where a page is attributed to a portal through wiki mark up, topic portals in the Wikisphere are defined by tags. A standard page becomes a topic portal when a user assigns the tag "topicportal" to the page.

For each topic portal amongst other meta-data tags that define a topic are set. For example, there is a topic portal around social media. "social" and "media" have be defined as discerning tags for it. Each page having "social" and "media" as tags (amongst potentially other tags) belong to the topic portal social media. By using tags to group pages belonging to a certain topic instead of moving a page to a space, multiple categorizations are enabled. A page can belong to different topic portals and therefore its content is reused in different contexts.

Since each user can assign tags to a wiki page, but a tag can only be assigned to a single page once, the type of the folksonomy is a mixture between narrow and broad folksonomy (see the definition of Vander Wal in 2.2.2).

TechnoWeb

TechnoWeb (or TechnoWeb 2.0) is a social application comparable to sites such as the Stack Exchange Network[10] or Yahoo! answers[11] featuring question and answers relevant for Siemens employees. Typically, this type of application is referred to as a Q&A system (question and answer).

TechnoWeb is a customized Life Ray Portal[12] installation. This tool is available for all Siemens employees since 2010 and supports the general networking of technology experts [KH09]. Each user can join and create networks discussing different topics defined through their tags. Additionally, each user can follow certain tags. Items having these tags assigned occur in the news feed for a user. In general there are several types of information items in TechnoWeb: news, networks and urgent requests (UR). News are announcements of users that want to distribute certain pieces of information, e.g. a certain upcoming event or a success story.

Urgent requests are the very central part of TechnoWeb. The requesters publish questions to a community of potential experts who are not definable by name in advance. The topic range is rather wide. Questions vary between where someone asks, where to get spare parts for a certain product, to what is the best tool for problem X. Some more information about TechnoWeb is provided in section 5.4.4 – in the motivation for an algorithm that has been developed for the distribution of questions in a Q&A system.

[9] see `http://en.wikipedia.org/wiki/Wikipedia:Portal`
[10] `http://stackexchange.com/`
[11] `http://answers.yahoo.com/`
[12] `http://www.liferay.com/products/liferay-portal/overview`

1.3. Research Issues

Three main research issues (challenges) have been selected to be addressed in this work. In a shortened form they can be formulated this way:

- **Semantic challenge:**
How can tags be enriched with relations in order to form more valuable structures such as thesauri or ontologies?

- **Hidden Structure challenge:**
How can other than semantic relations between general entities (not only tags, but also users and information items) be found out of social tagging data?

- **Orchestration challenge:**
How can tags from different applications be aggregated?

The *semantic challenge* deals with diminishing the disadvantages of social tagging in comparison to more controlled forms of annotating resources (see chapter 2 for more explanations on this issue). This is typically addressed by the field of information science. In the tradition of knowledge discovery in databases (or data mining) an aim of this work is to find information implicitly contained in social tagging data. This is formulated via the *hidden structure challenge*. The *orchestration challenge* falls into the area of software engineering and system design. An integration of the various facets of the research issues into a coherent solution is subject of the proposed approach.

1.4. Proposed Approach

The above-mentioned challenges are targeted in two blocks: (i) development and adoption of algorithms that can be applied on social tagging data, (ii) implementation of a tagging framework for integrating and testing the algorithms in an existing IT landscape.

(i) Development and adoption of algorithms This block deals with the semantic and hidden structure challenge. First, several approaches for automatically deriving relations between tags via statistical analysis are proposed. Well established data mining algorithm have been adopted for this purpose.

In order to utilize existing structured information sources such as thesauri (for example WordNet[Mil95]), ontologies or simpler hierarchical structures present in web directories (such the open directory project[13] (alias DMOZ)) or the category system of Wikipedia, a simple algorithm for mapping tags to these type of sources is implemented.

Both of these described approaches function as input for suggestions incorporated in a web technology based thesaurus editor in order to achieve better results. The desired

[13] `http://www.dmoz.org/`

target of these efforts is to achieve a modest shift from social tagging to a term-based thesaurus. The terminology for the categorization of meta-data and indexing approaches is explained in chapter 2. In figure 2.1 (chapter 2) an overview of vocabulary approaches is given – arranged in the power of expressiveness from left to right. In some papers, a desired and possible extension of folksonomies is proposed, in order to achieve a shift from folksonomies to ontologies [BSWZ07]. This approach seems rather unrealistic and magically. Furthermore, in the application scenarios of this work, ontologies are simply an overkill. It is assumed that for basic information retrieval problems a term based thesaurus structure comparable to an association thesaurus [JC94] is the right way.

Additionally to the implicit relations between tags, further hidden structures have to be investigated. First a method for discovering communities of interest in social tagging data is presented. Second, an algorithm for channeling questions in a Q&A system by utilizing the information contained in social tagging data is described.

(ii) Implementation of a tagging framework Embedded into a "real world" corporate environment, a tagging framework is developed in order to show the usefulness of a shared service. Within this service, the social tagging data of different applications is aggregated. The actual management of the tagging data stays within the power of the individual application. The tagging framework offers an API where applications can access provided methods, such as "recommend experts" on a certain topic. Having the social tagging data collected from many different applications enables the tagging framework to compute better results than a single application could, based on its own data.

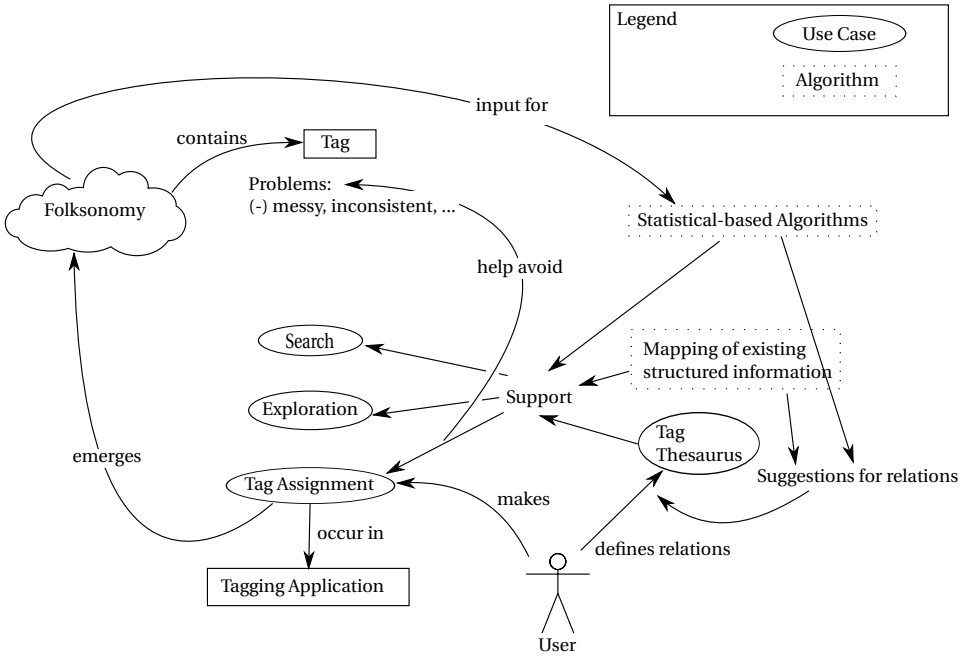


Figure 1.3: *Overview of the approach in combination with use cases.*

Both blocks are used to support several use cases. Figure 1.3 shows the approach in con-

text of the practical use cases (displayed as ellipses). On the bottom there is a user. A user makes tag assignments. These tag assignments happen in a tagging application. The collection of all tag assignments forms a folksonomy (all social tagging data aggregated – see section 2.2.2 for a more in depth description). In the middle, there are three central use cases: search (search for a specific artifact), exploration (trying to get an overview of a collection of information items) and tag assignment (the act of assigning a tag to an entity). Each of these use cases are influenced by the disadvantages or disadvantages tangled with social tagging. The search use case has the typical information retrieval problems. Exploration needs some kind of structure to browse a set of information items.

Tag assignment needs some kind of support in order to establish a consistent tagging practice (avoiding spelling mistakes, assign enough and appropriate tags). On the right there are two algorithmic approaches displayed: Statistical based algorithm (algorithm that try to find patterns in folksonomies), mapping of existing structured information (trying to map tags to existing structured information sources in order to utilize the contained structure). These two kind of algorithmic approaches rely on the data contained in the folksonomy.

As it turns out, these algorithms are far away from a perfect solution. A hybrid design in the form of a thesaurus editor has been developed (displayed as use case since user interaction is required). In the thesaurus editor, a user can state that two terms are related in a certain way (broader, narrower, related or synonym term). Both algorithmic approaches function as input for the thesaurus editor. The suggested relations help the user with defining relations. These two algorithmic approaches and the thesauri relation can be used to support the practical use cases. For example (alternative) tags can be presented to a user in the search use case or a user can be supported during tag assignment. Having a support in the tag assignment process the quality of the folksonomy is supposed to increase and the problems related to social tagging should be reduced. Some kind of feedback circle emerges.

For addressing the *orchestration challenge*, the thesaurus editor is embedded into a framework where the social tagging data of several applications are aggregated. The framework in return provides services that these applications can use in order to improve cross application user experience.

Choosing the right scientific method for dealing with identified research questions is not a trivial task. Sometimes the *right* choice of method is inherent with the specific problem. In mathematics, a deductive approach is the method of choice. A theorem is shown to be true via a mathematical proof. If testing a hypothesis can only be achieved (by generating and) by examining empirical data, an inductive[14] (or sometimes even abductive[15]) approach is the remaining alternative.

A Design Science approach is followed in this research. Figure 1.4 depicts the general methodology of Design Science. At first, an existing problem has to be identified and elaborated. In a second step, a solution has to be suggested and implemented in a third step.

[14] Not to be confused with mathematical induction or structural induction.

[15] This is popular for example in archeology, where scientists – in absence of a time machine – have to check theories on the remains of past times.

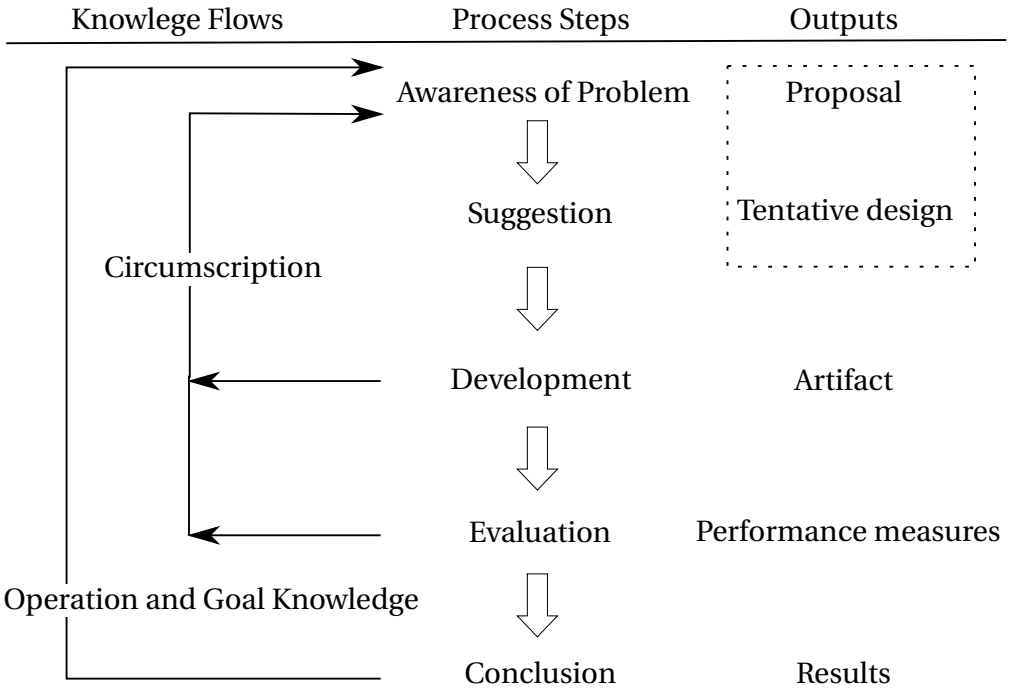


Figure 1.4: *General Methodology of Design Science – see [BHRS07]. The process steps from the awareness of a problem to a conclusion are shown.*

The proposed solution has then to be evaluated and conclusion has to be drawn out of the results of the evaluation. Typically there are several iterations necessary in which the proposed solution and the implementation are refined in order to achieve the desired result. The details of the iterations are not described in this work. Only the resulting solution is presented.

Hevner et al. [HMPR04] define 7 guidelines in order to confirm with the methodology of design research – see table 1.1. The framework described in the paper is the methodical basis for this thesis.

Based on an implementation of a tagging framework (guideline 1), several algorithms are implemented, improved or new ones designed (guideline 3). Several modules of the implemented framework are productively used inside Siemens since November 2010 (guideline 2). The resulting implementation is called STAGS (Siemens Tagging Service) and described in chapter 6. The results of the work have been published at various occasions, such as conferences[16] (guideline 5). STAGS is still under improvement and new modules that are not mentioned in this thesis are created in order to reflect new use cases and requirements (guideline 6). Interviews conducted during the evaluation of STAGS (see chapter 6.7) show its value for technical as well as non-technical audiences (guideline 7).

[16] see chapter 5 for references.

Guideline	Description
Guideline 1: Design as an Artifact	Design-science research must produce a viable artifact in the form of a construct, a model, a method, or an instantiation.
Guideline 2: Problem Relevance	The objective of design-science research is to develop technology-based solutions to important and relevant business problems.
Guideline 3: Design Evaluation	The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.
Guideline 4: Research Contributions	Effective design-science research must provide clear and verifiable contributions in the areas of the design artifact, design foundations, and/or design methodologies.
Guideline 5: Research Rigor	Design-science research relies upon the application of rigorous methods in both the construction and evaluation of the design artifact.
Guideline 6: Design as a Search Process	The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.
Guideline 7: Communication of Research	Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.

Table 1.1: *Design-Science Research Guidelines [HMPR04]*

1.5. Thesis Outline

The remainder of the thesis has been organized into the following chapters.

Chapter 2: An overview of annotation and knowledge modeling approaches is given. Investigating other methods helps understanding the advantages and disadvantages of social tagging. A special emphasis in this chapter is on the characteristics of social tagging.

Chapter 3: Related and Existing Work is presented. Included are data mining approaches, tags to structured input mapping as well as thesaurus editor interfaces.

Chapter 4: Use cases and requirements are elaborated. Three fine granular use case (tag suggestions, information navigation and semantically enhanced search) are described. Functional and non-functional requirements based on the use cases and further reflections are listed. This builds a foundation for the design decisions and the implementation in the following chapters.

Chapter 5: It targets the first two research issues in this thesis: a) *Semantic challenge* and b) *Hidden Structure challenge*. a) How can tags be enriched with relations? Three approaches are presented: i) Statistical Data Analysis, ii) Mapping tags to existing information structures and iii) a thesaurus editor where users can define relations manually. b) The dis-

covery of communities based on tags, as examples for other hidden structures functions. An approach for channeling of "urgent requests"[17] is provided. Additionally, a simple algorithm for suggesting tags based on a full text is described. The evaluation of the different algorithms is included after their description – where applicable. The algorithms are rather heterogeneous in their nature. Hence providing an evaluation "in place" increases the readability of the presented argumentation. In software testing, this is comparable to unit tests ensuring the functionality of individual components.

Chapter 6: STAGS, an implementation of the proposed tagging framework is described. An overview of the architecture and the design decisions are given. The requirements elaborated in chapter 5 are discussed based on the chosen implementation. The evaluation of the implemented system, STAGS, is content of the last section of this chapter. This is analogous to system testing in software testing. Unlike unit tests via integration tests the overall functionality of a system (including the combination of the individual components) is verified[18]. Log files from the third of May in 2012 to the twenty-eighth of January in 2013 were analyzed. In the nine months covered, there were 57,186 different users in total. By the answers of a questionnaire with five different expert users aspects of the Updated DeLone and McLean IS success model (D&M model) [DM03] are discussed. These expert users are project managers or responsible for a platform that makes use of STAGS.

Chapter 7: A concluding chapter contains an overview of the results and mentions future work. A summary of the contributions of this research is provided.

[17] A form of request for help, comparable to question on `stackoverflow.com`. For details see chapter 5.

[18] Typically in software testing there is also a phase of integration testing. This phase is not applicable here because it conducted implicitly.

CHAPTER 2

Meta-data or Indexing Approaches

But, as we consider the totality of similarly broad and fundamental aspects of life, we cannot defend division by two as a natural principle of objective order. Indeed, the "stuff" of the universe often strikes our senses as complex and shaded continua, admittedly with faster and slower moments, and bigger and smaller steps, along the way. Nature does not dictate dualities, trinities, quarterings, or any "objective" basis for human taxonomies; most of our chosen schemes, and our designated numbers of categories, record human choices from a cornucopia of possibilities offered by natural variation from place to place, and permitted by the flexibility of our mental capacities. How many seasons (if we wish to divide by seasons at all) does a year contain? How many stages shall we recognize in a human life?

— Stephen Jay Gould (1941 – 2002)

Contents

This chapter provides an overview in meta-data or indexing approaches in general. Information organization is an important problem especially in the information age. Investigating other approaches leads to an insight in the problems each individual method tries to solve and what new problems emerge with each approach.

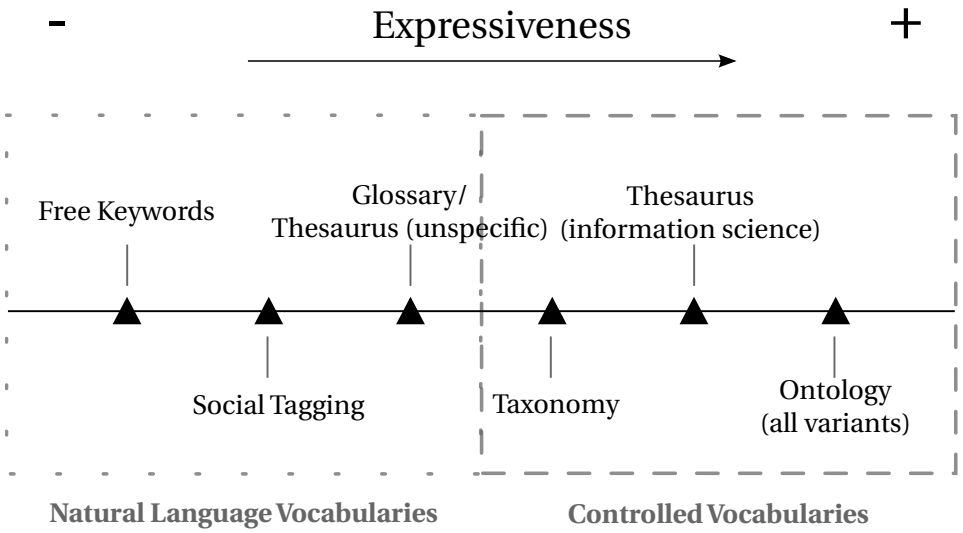


Figure 2.1: *Expressiveness of vocabulary approaches (derived from [Wel07]).*

Figure 2.1 contains a schematic representation of vocabulary approaches arranged by their "expressiveness" – *free keywords* with the least structure, *ontologies* with the most. It reflects the composition of this chapter and is a simplified version of [Wel07]. Each concept is explained in detail in the following sections. A focus is on Social Tagging – see section 2.2.2.

In the first part *natural language vocabularies* are presented (figure 2.1 on the left). These vocabularies have few or no restrictions on the terms one can apply in order to categorize an information item. The second part describes *controlled vocabularies* (figure 2.1 on the right). For these vocabularies, there are, depending on the complexity, few or many restrictions on the way information items can be organized.

An *exhaustive* description of *all* knowledge representation approaches is not desired in this work, because it does not deliver any significant surplus value. For some more details on the presented and omitted concepts see [SS08], [Gau05], [PB06], or [IOF07].

2.1. Introduction

The representation and organization of information or knowledge[1] has been a major challenge since immemorial. Especially nowadays with the beginning of the digital age, information overload[2] or flooding has become a serious problem. As Rutherford Rogers is

[1] The definition of both terms, information and knowledge, is subject of discussion in various scientific fields.
[2] A term said to be popularized by Alvin Toffler.

quoted – unsourced: "*We're drowning in information and starving for knowledge*". From a user perspective finding relevant information has to be fast, focused and simple.

Weinberger [Wei07] discusses in a very popular book under which circumstances it makes sense to use different kinds of approaches to organize (information) items. Sometimes it does not make any sense to apply some kind of explicit organization mechanism at all. For example, while Weinberger was a student, he and his roommates used to put spoons, knives, and forks unsorted into their drawer, after doing the dishes. In their opinion, it is a bigger effort to sort the cutlery into the right slots than to simply look for the needed cutlery when setting up a table. Weinberger also states that typically there is no *right* way to organize entities. Arranging items in supermarkets is a science of its own. Actually for each customer the various products should be placed at a different location, depending on the items a customer wants to buy in the current transaction.

Individualized optimizations can be seen from different point of views as well: a marketing department wants customers to buy as much as possible. Being a customer, a person in for most cases wants to spend as less time as possible on shopping in order to have more time for other things. In the real world one is restricted by given space and time constraints. In a virtual world such as the Internet, with computer support, it is possible to supply a view on products tailored for each customer individually.

In modern Internet shops, one can take many paths in order to find the desired items. This type of navigation is normally implemented via facets. *Facets* are classes of characteristics of an object. In faceted classification, an object can be assigned to multiple facets. The idea is that facets are rather assigned to objects then the other way round. *Faceted browsing* is a user interface pattern for navigating through a huge set of objects by narrowing down the potential results based on meta-data characteristics.

Unlike in general classification, an object is assigned to multiple classes. This is a central aspect of facets: items can be in several categories or need to have diverse types of properties. For example, trousers have a size, a color, can be for men, women or unisex, can have certain price range, and can be made of denim, cotton, and other materials.

By using facets, a customer can start with for example blue trousers then select men and then only denim. Alternatively one can start with trousers for men then select denim and blue. Each path leads to the same results. One main advantage here is that the user is not limited to a certain path through which an item is findable. This is normally much easier for a person. If adequate filters have been applied, in only one step (or only a few steps), items are narrowed down to a selection of items, a customer is looking for.

One very important project around this concept is the flamenco project[3] lead by Marti Hearst. In contrast to navigating through a predefined path of decisions, for example based on a decision tree, a user can choose which restrictions he or she wants to apply. The number of results for a potential additional restriction is typically presented to the user. Especially the choice of order in which different filter criteria (facets) are applied can significantly vary from user to user. In context of a desktop search, a system called Feldspar [CMF08] addresses this issue. It is easier to guide a user through a search by his or her individual associations.

[3] `http://flamenco.berkeley.edu`

Organizing *information* objects or other entities is not an easy task. Depending on the use cases — for example a taxonomy for biological classification, organizing the books in a library or even use cases affecting daily life – different approaches have been developed or adopted. In this chapter, an overview of approaches for organizing and indexing information objects is presented. The used terminology in this field is fuzzy in the usage and sometimes leads to confusion [Pan06]. A selection of the most important representatives is shortly presented. The author attempts to stay with the most common usage, which reflects the terminology in information science. A strict separation between the different terms is not always possible. A taxonomy can for example be seen as a special case of an ontology.

2.2. Natural Language Vocabularies

Natural language vocabularies schemes do not restrict the used terms to annotate resources. An user can decide which terms are the best to describe an entity. This is in contrast to controlled vocabularies that are discussed in a later section.

2.2.1. Free Keywords

Indexing entities with *free keywords* is the simplest form of text based indexing methods. In general there is no restriction on the used terminology and anyone can specify terms under which, the corresponding items can be found. These keywords may be contained in the corresponding text but there is no need for that. This is in contrast to most basic algorithms for automatic indexing of texts used in information retrieval, such as algorithms based on Tf-idf[4] computations [MRS08]. Classical automatic indexing algorithm are only able to extract words that are contained in a text[5]. Free keywords are applied by a person and can therefore be terms that are important to describe a document, but are not part of the document itself. Selecting the best terms for indexing a document is a task where a real person is in advantage of an automatic algorithm. At least if one considers the current state of the art in natural language processing and information retrieval.

2.2.2. Social Tagging

Social tagging became very popular with the rise of the Web 2.0. It is an important piece of the puzzle that makes up this phenomenon. In contrast to traditional categorization systems, users can use keywords freely without having to select a term from an existing vocabulary. This leads to less cognitive efforts and therefore a faster and less challenging process of classifying objects. While in certain domains such as library science or classification of

[4] *Term Frequency–Inverse Document Frequency*: A popular weight that relates the frequency of a certain term for a current document to the absolute frequency of this term in a whole document corpus. When a term frequently occurs in a certain document, but is a less frequent term in the document collection, it is supposed to be – with limitations – characteristic for the regarding document.

[5] More sophisticated algorithm also consider synonym or related terms for indexing.

medical artifacts (such as X-ray images), social tagging may be not an adequate replacement for more expressive annotation approaches. For the case of Internet applications, it has superseded alternative approaches.

Social tagging is a special variant of free keyword indexing. The significant difference is the social component: a user makes a tag assignment, it means he or she gives a vote that a tag is related (in an unspecified way) to a resource. As in all Web 2.0 applications interactions between users are important. Networks of links between users, tags and resources emerge. This additional information provides a decisive advantage over traditional free keyword indexing.

Characteristics of Social Tagging

In contrast to alternative approaches for indexing or annotating entities social tagging has some unique features. The following section provides a short summary of characteristics of Social Tagging.

Ease of Use Mathes [Mat04] cites Stewart Butterfield (co-founder of Flickr): "*Aside: I think the lack of hierarchy, synonym control and semantic precision are precisely why it works. Free typing loose associations is just a lot easier than making a decision about the degree of match to a pre-defined category (especially hierarchical ones). It's like 90% of the value of a proper taxonomy but 10 times simpler.*"

Although the actual numbers are a subject to discuss, Butterfield addresses the point correctly. Tagging is simpler and costs less effort than categorizing objects based on a more formal system. To quote Hendler [HG08]: "*tags are trivially easy to use.*"

Sinha [Sin05] tries to find an explanation for that: The cognitive process behind classical categorization consists of 2 stages (see fig. 2.2 (a) – stage 0 is not considered as an extra stage). In a first stage a user is confronted with multiple concepts that come to his or her mind related to a certain entity. In a second stage the user has to select *one* of these concepts for categorization. This selection can be very demanding since in many cases one has to ponder which exact concept to choose. When tagging an object, a user writes down all concepts that he or she thinks fits for it (see fig. 2.2 (b)). Hence, social tagging is faster for annotating information and has a higher user acceptance.

Broad and Narrow Folksonomies The term folksonomy was introduced by Thomas Vander Wal in a discussion on a mailing list [Van07b]. A folksonomy is a portmanteau consisting of the words taxonomy and folk. It refers to the implicit structures (therefore taxonomy) between entities emerging from the individual tagging behavior (therefore folk). "Taxonomy" can actually be considered as the wrong expression since there are no real hierarchical structures present. Some authors hence avoid the term folksonomy and prefer to simply use *social tagging* instead.

In this work both terms are used synonymously. Vander Wal provides following definition for the term folksonomy [Van07b]: "*Folksonomy is the result of personal free tagging of information and objects (anything with an URL) for one's own retrieval. The tagging is done*

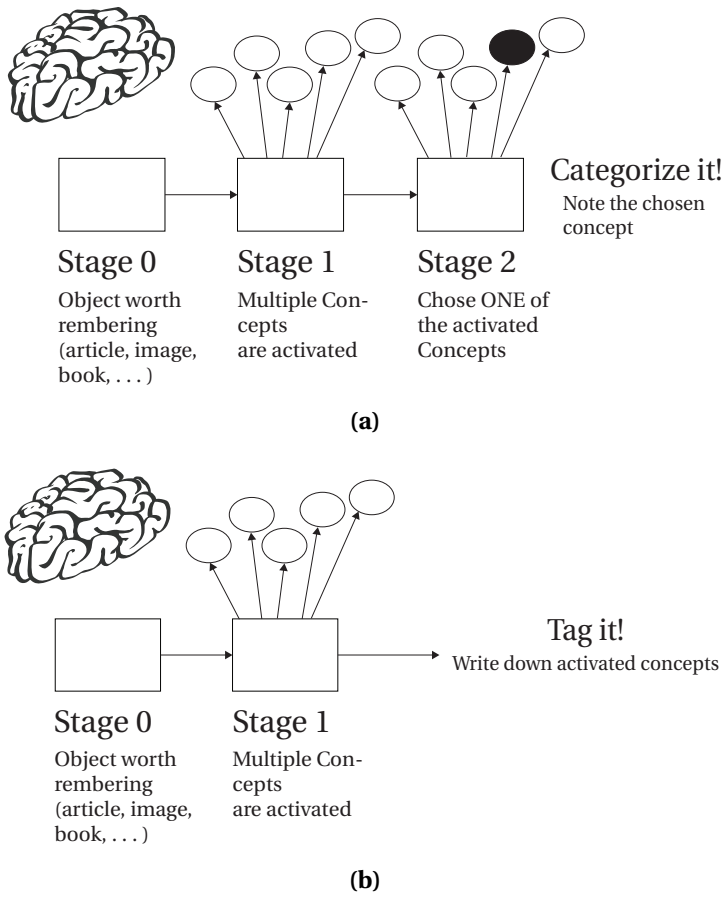


Figure 2.2: *Cognitive process behind categorization (a) and tagging (b) (see [Sin05])*

in a social environment (usually shared and open to others). Folksonomy is created from the act of tagging by the person consuming the information."

In another blog post, Vander Wal introduces two categories of social tagging: *Broad* and *narrow folksonomies* [Van07a] (see fig. 2.3). Both categories are distinguished by the type of users that are allowed to tag a resource, for example either the author or any user. Additionally, a resource can be tagged with the same keyword more than once – by different users.

Broad folksonomies means that anyone can tag a resource and a resource can have the identical tag originated by different users. This leads to some kind of vote what tags fit best for a given resource. Typically one can observe a power-law distribution of possible terms. A few tags are extremely popular, but the majority of tags are used only infrequently [Ton06]. The applied tags for a resource converge to some most frequently used and therefore important phrases. An example for a broad folksonomy is Delicious. Every bookmark (identified by its URL) is typically stored and tagged by several different users. For example the bookmark for the URL `http://java.sun.com` can be found un-

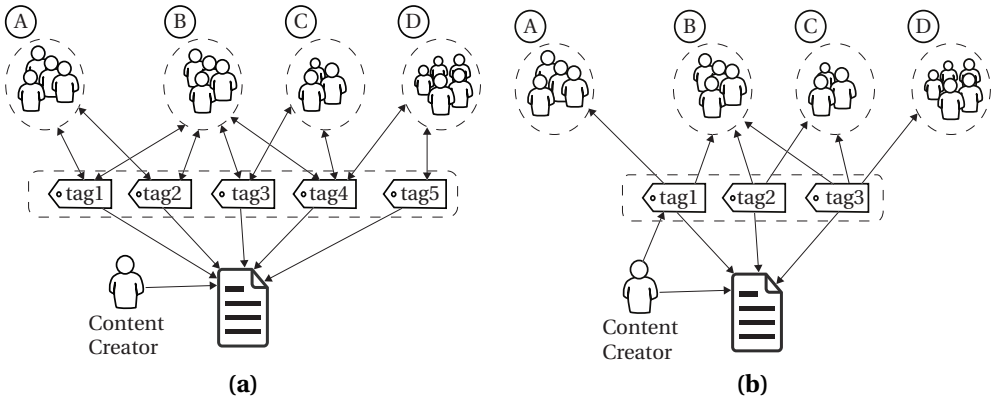


Figure 2.3: *Broad (a) and Narrow Folksonomy (b) (see [Van07a])*

der `http://www.delicious.com/url/0c657db11cb416e125446ee10eae99a3`[6].

A *narrow folksonomy* is given when only the creator of a resource is allowed to tag it. This is the case for Flickr. Within Flickr only the owner of a picture can tag it[7]. If only the creator of a resource applies tags to a resource, it is assured that the creator keeps in control over the tags his or her resources can be found with. Hence, tags that he or she considers inappropriate can be excluded. The disadvantage of a narrow folksonomy is that in some cases it is more difficult to find a resource, because the creator has assigned unusual tags.

Functions of Tags Different types of tagging applications with different purposes and the tagging gusto of individual users lead to different tagging patterns. While investigating tags used in Delicious Golder et al. [GH06] identify several functions of tags. They analyzed two data samples collected between twenty-third and twenty-seventh of June 2005. The first set containing 212 URLs and 19,422 bookmarks consisted of bookmarks with popular (most frequently used) tags during that period. The second set contained 68,668 bookmarks collected from 229 randomly selected users.

They work out seven possible functions of tags:

1. *Identifying what (or who) a document is about:* These are tags that denote an entity. This includes general substantives of the everyday life as well as more specific terms referring to persons or organizations. Examples are “programming,” “w3c,” or “berners-lee”.

2. *Identifying what the document itself is:* Tags are also used to describe an artifact, e.g. if a document is a “book” or “article”.

3. *Identifying who owns the document:* For the case of Delicious there are often blog posts bookmarked. Then sometimes the owner of the blog or the author of a blog post is used as

[6]The last part of the URL is the MD5 sum for the URL of a bookmark.
[7]This restriction might change over time.

tag, e.g. “peter_brantley”[8]

4. *Refining documents or other tags:* Sometimes additionally to more general tags more specific tags are used. An Example is “java” being a less general word than “programming”.

5. *Identifying qualities or characteristics:* “funny” or “inspirational” can serve as examples for this kind of usage. This is a subjective view of an entity.

6. *Self reference:* Artifacts having a specific relation to the person tagging it are typically tagged with “myown” or “mystuff”. For example, in Bibsonomy lists of publications can be dynamically created by filtering content with a certain tag restricted to a certain user. This filtered list can be used in to embed a publication list into a personal homepage.

7. *Task organizing:* While working on a specific project or in order to collect interesting information artifacts, tags such as “toRead,” “todo” or “job_search” are used.

User Incentives Tagging is usually not only used, because a certain software offers that feature by default, but because people are using it for a certain purpose. This includes both personal and social aspects. Additionally, as a side effect, by investigating the motivation for the usage of a tag, one can make conclusions about the function of a tag – as mentioned in the section before. If somebody is doing research regarding a certain topic, he or she will typically apply tags such as “toRead” or “semanticWebPaper2010”.

Marlow et al. [MNBD06] describe incentives and motivations for users to annotate resources with tags:

1. *Future retrieval:* In the context of social bookmarking tags, such as “todo” or “toRead,” are utilized by an user to come back to a resource at a later point in time.

2. *Contribution and sharing:* In Flickr, tags are used to refer for example to a certain event. This can be pictures of a vacation trip or another event that somebody wants to share with his or her friends or even potentially strangers, such as it is the case for big events, for example “loveparade” or “burning_man”.

3. *Attract Attention:* Sometimes (popular) tags are applied to a resource in order to make more people find it. In extreme cases this behavior can be interpreted as SPAM.

4. *Play and Competition:* The target of ESP games[9] is typically to apply the same tags to a resource as another user. For matches users are rewarded with points.

5. *Self Presentation:* Sometimes people want to state a certain personal relation to an entity. For example in Last.fm some people use the tag “seen_live”.

6. *Opinion Expression:* Tags can be used to state a certain attitude to or opinion about a resource. Sometimes people use asterisks for rating objects, e.g. “*****” expresses an excellent rating for an item.

[8] Peter Brantley is an author writing for the popular “O’Reilly Radar” blog `http://radar.oreilly.com/peter/`.

[9] `http://www.espgame.org/gwap/`

Roles Users Take in Social Tagging In contrast to free keyword indexing or more elaborated ways of annotating information items, *social* tagging benefits from interactions between users applying tags. Tags can be used as signals stating a certain relation to a topic.

Thom-Santelli et al. [TSMM08] identify five major roles users adopt in social tagging applications. They interviewed thirty-three people. The tagging activities did take part in several enterprise systems available for internal use within a large corporation.

Based on the usage of tags they identify following roles:

1. *Community-seeker:* These are people trying to find like-minded people or communities regarding a certain topic.

2. *Community-builder:* If a community around a topic does not exist, these users try to establish one.

3. *Evangelist:* If someone is recognized as an expert for a topic in a community he or she is referred to as evangelist.

4. *Publisher:* Production and dissemination of content to a variety of targets is the job of a publisher.

5. *Small Team Leader:* These users are less frequent taggers using tags inside a certain terminology with a narrower audience.

Assets and Drawbacks of Social Tagging

Social tagging is (ad hoc) a flat approach for organizing information items. In general – not only in the context of social tagging – this is a quite modern trend which can be subsumed by following two statements:

- **Trees versus Leaves:** "*The old way creates a tree. The new rakes leaves together.*" [Wei05]
- **Ontology is Overrated:** "*The idea of a perfect scheme is simply a Platonic ideal*" [Shi05]

Both quoted articles reason that having a sophisticated — but more complex — way of annotating resources is not needed and also an utopia — at least for a perfect representation of the world or even a simple domain.

No Hierarchies If there are enough links between entities out of the box present there is no need for an artificial hierarchical structure. Figure 2.4 illustrates this argument. There are three stages of nodes with links: *(a)* A traditional folder hierarchy. This is a strict hierarchy. Nodes *cannot* be freely interlinked. *(b)* A folder hierarchy with some free links between nodes. *(c)* Just interlinked nodes without a given hierarchy. When there are enough references between information items there is no need for an hierarchical system.

This trend is noticeable in modern file managers such as the one in newer versions of Ubuntu utilizing zeitgeist[10]. Zeitgeist is a service that keeps track of the history of user

[10] `http://live.gnome.org/Zeitgeist`

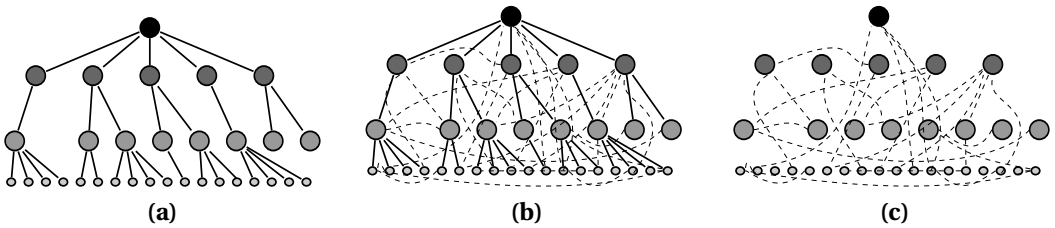


Figure 2.4: *File Systems and Hierarchy (see [Shi05]): (a) A traditional folder hierarchy. (b) Folder hierarchy with links between nodes. (c) Interlinked nodes without a given hierarchy.*

actions, such as which emails have been written and what documents have been edited or created. By having this kind of activity stream integrated into a file manager, the time metaphor can function as a central criterion to find the desired information. In combination with tagging and other meta-data there is no need for a folder structure.

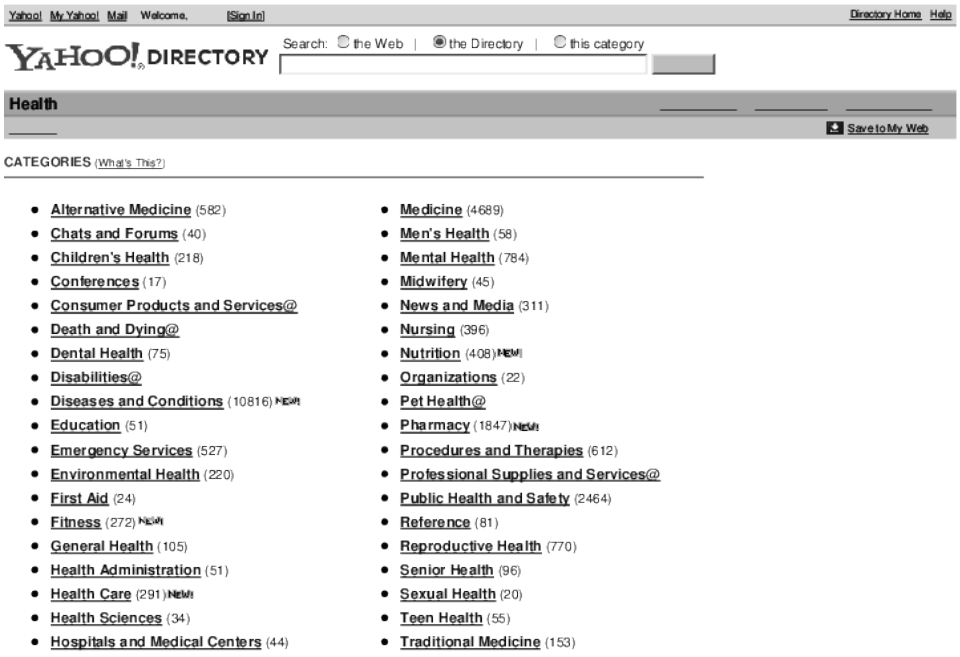


Figure 2.5: *Yahoo! Directory for the category Health – screenshot September 2009.*

In context of the Internet, having a manually created hierarchical structure is very hard to maintain. Yahoo! (Yet Another Hierarchical Officious Oracle), started in 1994, is one of the oldest directory of web sites on the Internet. Yahoo being very successful in the late nineties lost its poll position in the search engine market to Google. Google had a better ranking strategy and provided access to its index via a plain search field. Google did not

need a manually created web directory. A search index and a smart ranking approach[11] turned out to be a better solution.

Figure 2.5 depicts an excerpt of Yahoo! directory for the category health. Categorizations can be incomplete and error prone. Sometimes a category contains terms that should be in a different category, because some users expect a term to be a sub-category of that category. For example in figure 2.5, "Death and Dying" is in the category "health" although it should belong to another category. In Yahoo! directory such terms are marked with an "@". The success of full text search (Google) and other approaches shows that such directories have a limited usage.

Lack of Structure Although Shirky [Shi05] and Weinberger [Wei05] argue that hierarchies or other structures are an unnecessary overhead – as described in the section above. There are still some people who are used to that kind of organizing information items. An article with the title "Don't Take My Folders Away! Organizing Personal Information to Get Things Done" [JPGB05] discusses this issue.

The main advantage of (social) tagging over traditional annotation methods (e.g. modeling and instantiating entities in ontology- or taxonomy-based systems) is the ease of use. Not only experts but also untrained users can utilize tagging for their needs. In tagging, there is no restriction concerning the permitted terminology. Social tagging, as concept within Web 2.0, supports the interaction of users on the social web because tags are not only intended for personal use, but are also intended for others to give them the opportunity to quickly estimate semantic aspects of given information items.

Because of its simplicity, tagging in its basic variant lacks any form of explicit structure that comes with other more formal categorization methods, such as ontologies or thesauri.

In general, one can distinguish between two categories of problems. The *first category* contains the following very common problems that come with free text annotations in general:

• *Typos, spelling mistakes, or different spellings:* This is the simplest case where tags are susceptible. A user might type "instuments" instead of "instruments". Also spelling variations are a problem, for example differences between American and British English such as "color" and "colour".

• *Special chars for word combinations ("_," "-," ".," "/") or camelCase*: Depending on the individual taste of a user or sometimes related to restrictions given by the tagging application (Delicious does not allow white spaces in between tags; Atlassian confluence[12]) "open source," "open_source," "openSource" might be a user's choice to tag a specific item associated with open source software.

• *Meta-Noise*: In Internet tagging applications, there are also effects related to SPAM (e.g. in Delicious user accounts are abused to link to dubious sites in order to gain more traffic), trolls (e.g. some users try to categorize items incorrectly for fun or destructive reasons) and pseudo "experts" (some people overestimate their expertise).

[11] The famous PageRank algorithm developed by Larry Page, Sergey Brin and other [PBMW99].
[12] A wiki software `http://www.atlassian.com/software/confluence/`

• *Different languages*: In tagging applications with international users, variants of the same term may occur in different languages, e.g. one might find pictures somehow related to "luck" under "Glück" (German) as well as "suerte" (Spanish) or "bonheur"(French).

These problems are also targeted in classical information retrieval [MRS08]. Syntactic issues can be dealt with by using spell checkers, stemming algorithms (e.g. Porter stemming [Por80]) or comparing string distance metrics (e.g. Levenshtein distance [BHS07]). SPAM detection in folksonomies for example is discussed in [KSHS08]. Language detection might be done e.g. by matching tags against several dictionaries.

The *second major category of problems* is the lack of (semantic) structure:

• *Synonyms*: Two words are synonyms when they have the same (or nearly the same) meaning. Examples are "buy" and "purchase" which can be nearly interchangeable used. "Dog" and "Canis familiaris" are synonym as well, but are normally used in a different context – the first one as widely used term, the second one mostly used in scientific articles.

• *Homonyms/ Polysemy*: Homonym means that two words are spelled (homograph) or pronounced (homophone) in the same way [Gau05]. Regarding tags homographs are the only relevant case. A typical example for a homograph is "bow" which can have several different meaning such as the weapon or to bend forward.

• *Acronyms*: Acronyms are abbreviations of longer terms. For instance "GIS" can stand for "Geographic Information System," "Greenland Ice Sheet," or "Gruppo di Intervento Speciale"[13].

• *Level of abstraction – hyponyms or hypernyms*: Depending on the expertise of an user or other circumstances (e.g. who is tagging for whom) different levels or abstractions for the chosen tags can be used. A picture of an angora cat can be tagged e.g. with "angora cat," "cat," "mammal," "animal," or "creature".

If a user uses a tag to search for information items, the returned result set can contain entities that he or she did not look for or it can miss some relevant items. For ambiguous tags such as ones with homonyms or tags that are acronyms, there are very likely entities in the result set that the user was not looking for. If it is hard for a user to choose the right tag to find the desired resources, then the search results may not contain the desired items. This can be the case with tags that have synonyms. Then the user simply did not pick the right one. If there are intuitively different possible levels of abstractions, then the tagger of an object might have chosen more (or less) general tags to describe an entity than the user has chosen to search for an entity. In general: The lack of (semantic) structure in social tagging leads to problems with identifying and finding information items.

Information Retrieval Aspects of Tagging The main purpose of tags is to find desired resources. For the use case of social bookmarking an user assigns tags in order "to keep

[13] For other examples see `http://en.wikipedia.org/wiki/GIS_(disambiguation)`.

found things found". Voß [Vos07] even argues that tagging might be seen as "Renaissance of Manual Indexing".

Chi et al. [CM08] investigate the efficiency of social tagging by applying criteria typically used in Information Theory. By testing their method on an example set of Delicious they conclude that information theory provides adequate methods to determine the efficiency of social tagging systems.

In context of multimedia retrieval Kierkels et al. [KSP09] describe how tags can affect queries.

In large systems, regarding the number of users and resources, there is a lack of precision for general terms. From [HG08]: "*common tags on Flickr include terms like "dad" (80,000+ photos), "Fred" (90,000+ photos) and "My (something)" (over 8,000,000 photos)*"

In order to discuss these aspects first, some basic measures used in information retrieval are defined (from [MRS08]):

Definition 2.1 (Precision) *Precision is the fraction of the documents retrieved that are relevant to the user's information need.*

$$Precision = \frac{\#(relevant\ items\ received)}{\#(retrieved\ items)} = P(relevant|retrieved)$$

Definition 2.2 (Recall) *Recall is the fraction of the documents that are relevant to the query that are successfully retrieved.*

$$Recall = \frac{\#(relevant\ items\ received)}{\#(relevant\ items)} = P(retrieved|relevant)$$

For very general tags such as "cat," "web," "sun_set" or "design," the value for Recall is typically very high, but Precision is low. General tags tend to be used to tag many items. Depending on the needs of a user having a high Recall is no disadvantage. For example: someone is looking for a picture of a sunset to use as a background image in a presentation. He or she is not actually looking for a sunset at a certain location or having other very specific requirements. Thus nearly any decent picture of a sunset is suitable. In order to ensure a certain quality or license, one can use additional filter mechanism. In Flickr for example, it is possible to restrict the result set to a certain license and to sort the list of images by descending popularity.

Looking for a special kind of image, such as a German Shepherd, filtering images by using "dog" may not return a relevant picture on the first few pages of the returned search results. Using "german_shepherd" may lead to a higher precision, but to a lower recall since potentially matching pictures only tagged with "dog" are not present in the search results.

At some point of maturity of a platform, a user may learn the importance of recall and precision for tagging and change his or her tagging practice [GT06]. Some approaches for cleaning and normalizing (such as merge similar or synonym tags) folksonomies use stemming algorithms or string distance measures [VDHS07, Mul07, WD08]. Similar to classical tokenizing in information retrieval [CMS09] there might occur false positive or false negatives when searching for objects, i.e. retrieval failures.

Tagging Applications as Islands on their Own Today's knowledge work [Hub05] is characterized by a multitude of larger information systems, smaller ICT tools, and underlying file formats. Most creative and therefore weakly structured workflows stretch across systems and tools. Therefore, tool supported knowledge work is often more a hassle than an efficient flow [Csi02] of activities. With the advent of Web 2.0 tools in organizations (discussed as Enterprise 2.0 [McA06]) at least granular hyperlinks, and the capability to embed those into content, supports minimal integration allowing to switch from one application to another. In rare cases, the hyperlink can be complemented by dynamic linked information, e.g. through RSS or ATOM feeds. Still, cross application integration is far from being efficient. These problems are referred to as (personal) *orchestration challenge* [Ehm10]. The term orchestration alludes to the requirement of composing and possibly configuring the tools needed for a certain task.

While this turns aforementioned workflows into "switch flows" between applications, challenges related to the organization of knowledge are not addressed by the mechanisms described so far. Typical Web 2.0 applications and augmented (client sided) desktop tools inspired by the web provide *tagging* as the smallest common denominator for content organization.

Having many different heterogeneous applications, tags have to be re-entered and user assistance, such as auto completion, cannot benefit from tags stored in other systems. The same holds for search, navigation and tag gardening [WP08] scenarios.

For example, at Siemens there are several intranet applications that support tagging (such as a blogging platform [Ehm08] and a wiki [Lin08]). If a user leaves the blogging platform and works on articles in the wiki, the information available in blogging platform is ad hoc not accessible by the wiki. The same is valid for the other way round.

For further information on social tagging, there is an an excellent book written by Isabella Peters [Pet09].

2.2.3. Glossary

In books, a glossary is typically placed at the last pages. It is used to provide an explanation for words that are assumed to need one. Sometimes, for the case of foreign words, translations are provided. Additionally, a glossary can be used to find occurrences of a certain term in the corresponding book. This is not to be mixed up with the term *index*. An index simple lists the occurrences of a word in a document. A word in a glossary is called a *descriptor*. Other used terms for descriptors are *subject headings*, *controlled terms*, or *preferred terms*. In figure 2, *Thesaurus (unspecific definition)* is basically equivalent to a glossary. A glossary can contain synonyms or other relations and might be called some kind of thesaurus.

Glossaries are further often used in documents that have to be very exact and unambiguous such as technical documentations, patents or use case descriptions in software development. A glossary helps in these cases to provide a more or less exact definition (and if needed/ desired alternative terms) for the used terminology.

Thesaurus (unspecific definition) can be treated as equivalent to a *glossary* for the sake of simplicity – but technically speaking, they are not identical. The term glossary is preferred

later in this chapter, because of its historic importance and because the term thesaurus (in the information science sense) is used later with a more specific meaning.

2.3. Controlled Vocabularies

Controlled vocabulary scheme, in contrast to natural language vocabularies, restrict the vocabulary that is used to describe an object. If there are synonym terms, only the one defined in the vocabulary can be used for annotation. Terms not included in the vocabulary are not permitted. The main distinction between the different types of controlled vocabulary schemes is the kind of potential relations between terms. More types of relations lead to a higher complexity simultaneously the expressiveness of a scheme increases.

2.3.1. Taxonomy

A *taxonomy* is a strictly hierarchical structure that is used for classification. There are typically two types of relations between entities (called *taxonomic unit* or *taxa* — singular *taxon*) in a taxonomy: supertype and subtype or alternatively generalization and specialization.

In biology, one speaks of biological classification (or scientific classification) which is used to group organisms by biological type (such as genus or species). This taxonomic system is named after Linnaeus Linnaean system. The biological classification is arranged in the following way:
Life → Domain → Kingdom → Phylum → Class → Order → Family → Genus → Species.

The taxonomy classifies species into subspecies. Figure 2.6 gives an example for the taxonomy of Primates of the animal kingdom tree[14]. By walking the tree, one can get an insight on the origins of the Homo Sapiens Sapiens (the modern human). It is worth to mentioning that there are still discussions whether a correct and adequate taxonomic classification of life is possible at all [Rao48].

In general, standardized names for organisms are crucial for communication among scientists. Therefore having a maybe not perfect classification system is better than having nothing to rely on. Biological classification is revised from time to time. For example when there are species that were placed by previous authors in different genera, but turn out to be closely related, they are typically reclassified in the same genus. Also when new species are discovered, the taxonomy is updated accordingly – see [Fut05] for more examples.

The full classification of the Homo Sapiens Sapiens is gone through in table 2.1[15]. For the sake of simplicity, the example is shortened by very specific categories such as *Subphylum*. This example is chosen here to describe a use case where a taxonomy is very useful. It helps gaining a deeper understanding of connections between entities in a taxonomy — in this case connections between (living) beings. The evolutionary processes[16] can be illustrated

[14] For more information about biological classification see [CR07], [CD04], and [Fut05].
[15] see `http://www.itis.gov/servlet/SingleRpt/SingleRpt?search_topic=TSN&search_value=180092`
[16] As discovered by Charles Darwin.

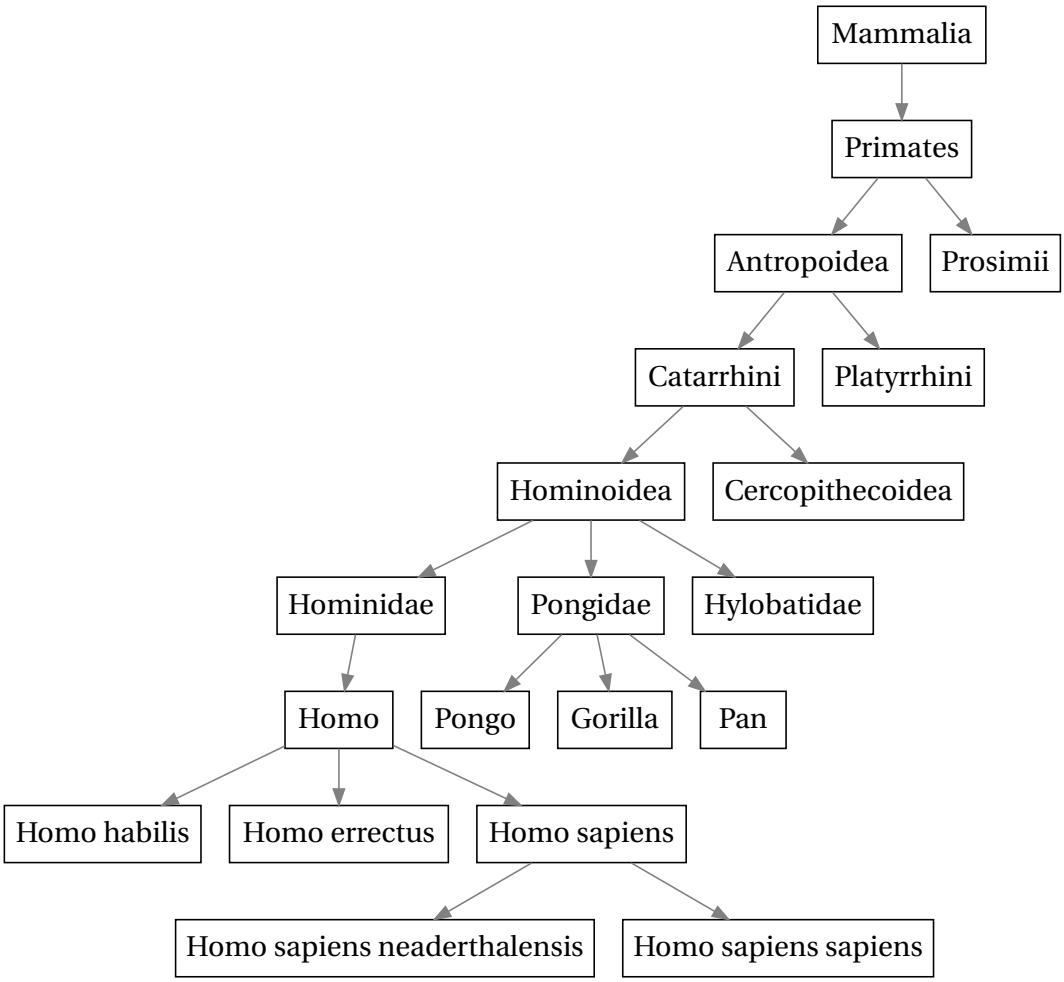


Figure 2.6: *Example taxonomy: biological classification, subtree primates*

through biological classification in a straight forward manner.

Another example for a taxonomy in the context of library science is the Dewey Decimal System (DDS). It was created by Melvil Dewey in 1873 and published for the first time in 1876. It is widely used especially in the Anglo-American language area for organizing books on library shelves. Also internationally, it is often used. In Germany there is a translation available since 2005[17]. The books of several German institutions can be navigated along the DDS with a web browser[18].

Clay Shirky uses the DDS in an article as a bad example for a categorization system [Shi05]. Figure 2.7 shows an excerpt of the category 200, the top category for books related to religion. The subcategories reflect the number of books in a library in the USA in the end of the 19th century. At this time, the chosen categories seemed to be reasonable. Because there were many books with topics around the Christian religion as a subject, hav-

[17] `http://www.ddc-deutsch.de/`
[18] `http://melvil.d-nb.de/melvilsearch`

Kingdom	Animalia – Animal, animals, animaux
Phylum	Chordata – chordates, cordado, cordés
Class	Mammalia Linnaeus, 1758 – mamífero, mammals, mammifères
Order	Primates Linnaeus, 1758 – homem, macaco, primata, primates, primates, sagui
Family	Hominidae Gray, 1825 – man-like primates
Genus	Homo Linnaeus, 1758 – hominoids
Species	Homo sapiens Linnaeus, 1758 – human, man

Table 2.1: *Taxonomic Classification: Homo Sapiens*

```
  Dewey, 200: Religion
210: Natural theology
220: Bible
230: Christian theology
240: Christian moral & devotional theology
250: Christian orders & local church
260: Christian social theology
270: Christian church history
280: Christian sects & denominations
290: Other religions
```

Figure 2.7: *Dewey Decimal System: 200 Religion*

ing a fine granular distinction between different categories of the Christian theology on a high level has been a natural choice. At present, the category of other religions (290) seems ill-designed. More books with topics around non-Christian religions, such as Islam or Buddhism have been published since the late 19th century. Most likely, these Religions would have a more prominent position if the DDS was designed a hundred years later.

It is hard to predict what system would be an adequate one in a couple of hundred years. Hence, having a static categorization system may lead to very strange side effects over time. With the restriction to a fixed number (ten) of categories per level adding additional categories is not possible. The fixed number of categories is an artificial restriction that does not make sense for modern information systems.

Ted Nelson states in a paper [Nel74] about his idea of hypertexts that "*hierarchies are typically spurious*". For many domains there is no *true* way to determine a clean taxonomy. Therefore, strictly hierarchical systems should be abandoned in favor of a network structure. This is of course a philosophical and idealistic view. For some use cases especially when it comes to a technical implementation - such as data bases - trees and hierarchies still have their application. Databases such as MySQL, Oracle, MSSQL, and PostgreSQL are used in many systems and therefore they have proven their practical application. It is the responsibility of a developer to find the most appropriate scheme for his or her specific use case.

2.3.2. Thesaurus

A *thesaurus* is a controlled vocabulary[19] of terms that can be used as keywords. There are several variants of thesauri depending on the area they are used in. Peter Mark Roget's famous Thesaurus of English Words and Phrases (1852) initiated the concept of a linguistic thesaurus. A *linguistic thesaurus* is some kind of dictionary where words are arranged systematically. This type of thesaurus is the most widely used and a known one since it is included in popular word processors such as Microsoft Word or Open Office. The use case of a linguistic thesaurus is hereby assisting a user in finding alternative words in order to avoid repetitions of phrases when writing texts.

Alternatively, a linguistic thesaurus can be utilized to determine the meaning of a word when in doubt. One very popular and freely available linguistic thesaurus is WordNet [Mil95] which is frequently referred to in a later chapter of this work. A screenshot of a WordNet Browser is shown in figure 2.8. The used search term is "know". The displayed results contain several senses with example sentences. Doing a query with a rather new term such as "web 2.0" or a term from a narrow domain such as "tarsorrhaphy"[20] returns no results. Hence, WordNet can be useful as a general purpose thesaurus. For a very specific domain or in cases where new terms emerge quickly WordNet is less useful.

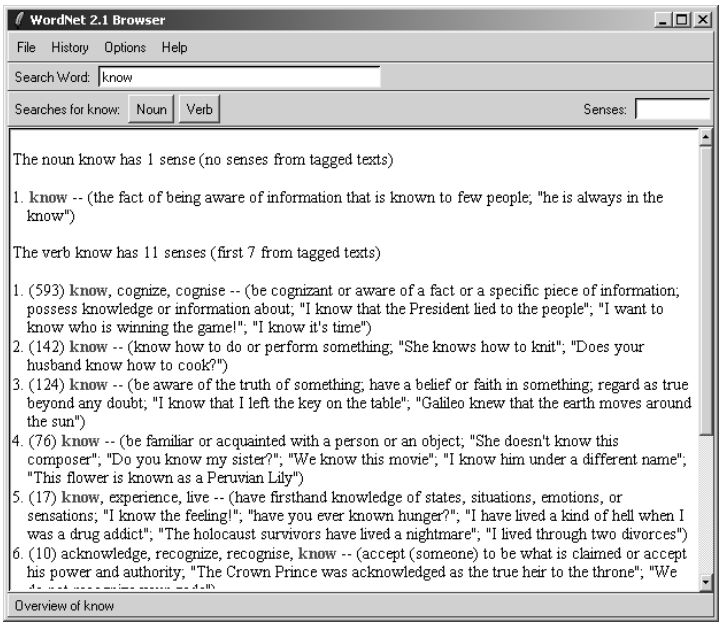


Figure 2.8: *WordNet Browser with results for "know".*

In *information science* or in the context of libraries a thesaurus is used to categorize information objects. A thesaurus is some kind of classification system backed by a controlled vocabulary with several kinds of relations between terms contained in the vocabulary. Here,

[19] Not to be confused with a glossary that is also sometimes simply called a controlled vocabulary.

[20] "Tarsorrhaphy is a surgical procedure in which the eyelids are partially sewn together to narrow the opening (i.e. palpebral fissure)." from `http://en.wikipedia.org/wiki/Tarsorrhaphy`

a thesaurus helps conducting research to a certain topic, e.g. by allowing a user to enter a classification system through different terms having the same meaning.

Furthermore, Thesauri are used in several scientific fields such as biology or medicine. These thesauri typically have are very narrow domain and a well-defined and restrictive terminology. Sometimes these thesauri are a preliminary stage to an ontology and also referred to as one. The Radlex ontology[21] is an example for such an ontology, taxonomy or thesaurus – depending on the point of view. It is a controlled vocabulary to classify information items, such as x-ray images or medical reports, in the area of radiology.

In context of *information retrieval*, typically a thesaurus is used to alleviate problems resulting of trivial search variation or of term ambiguity by offering terminology control [TDB+06]. Some more details about thesauri can be found in [Wer85].

2.3.3. Ontology

"Ontology" is a rather fuzzy term – at least sometimes in its usage. To quote a project web page on the topic of the Semantic Web by Aaron Swartz[22] from 2001: "*It doesn't seem anyone is really sure what an ontology is.*"[23] Originally in philosophy (meta physics), the term Ontology was dealing with the concept of "being". This meaning is only slightly related to the one used in the context of of computer and information science. Here, the most popular definition comes from Gruber in 1993 [Gru93]:

Definition 2.3 (Ontology short) *An ontology is an explicit specification of a conceptualization.*

There is also a newer and longer definition by Gruber in 2009 [Gru09]:

Definition 2.4 (Ontology long) *In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application. [...]*

Figure 2.9 displays an attempt to arrange the different kind of approaches that can be considered as some kind of ontology. From bottom left to top right the expressiveness of semantics, a model can contain increases. For example, taxonomies contain only hierarchical relations whereas description logic allows modeling more sophisticated relations. Below, the black circles are important types of semantic relations that are introduced with each concept of formalization (with bigger font size). Some methods are more frequently used than others. For example, entity relationship (ER) models are popular for developing a data base design. Unified Modeling Language (UML) is typically used to describe

[21] `http://www.radlex.org`

[22] A Internet activist who tragically passed away in January 11, 2013. He was a member of the RDFCore working group at W3C in 2001.

[23] `http://logicerror.com/ontology` visited November 2013.

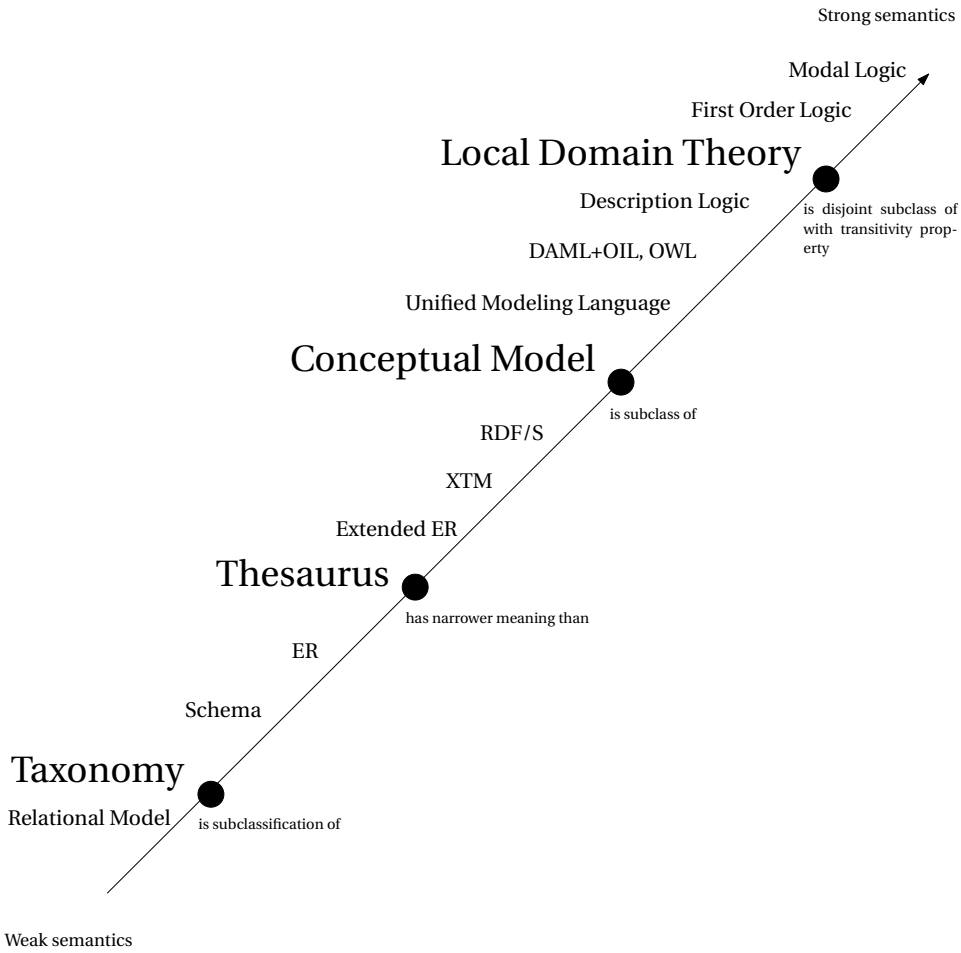


Figure 2.9: *The ontology spectrum: Weak to strong semantics [DOS03].*

and discuss software architectures. Systems that incorporate (higher order) logics are less frequently used, but are niche players in areas where there is a need to proof that the implementation of a system is consistent with given rules and constraints. For some companies, such as banks or insurance companies, this is an important requirement. In general, one may say: The more expressiveness a formalization concept allows, the less frequently it can be found outside an academic context.

Using modal logic to describe systems can be very useful in some situations. The idea behind $\Box$ for necessary conditions and $\Diamond$ for possible conditions might confuse some people. An example for the usage of these unary operators is: $\Box$ a circle is round, $\Diamond$ there are green zebras or $\neg\Diamond$ a circle is not round. Modal logics used to be popular in the middle of the last century.

Other concepts of higher logics have their renaissance in certain languages for the semantic web. For example, the Web Ontology Language (OWL) is a family of languages that allow knowledge representation with formal semantics. The possible semantic relations depends on the individual family member. OWL lite is a rather restricted language that targets the modeling of hierarchical classifications. OWL DL tries to keep computational

completeness and decidability while allowing to use some higher logic. OWL full is a compromise between OWL DL and OWL lite. It has been created to ensure some compatibility with RDF Schema (abbreviated with RDF/S in figure 2.9; RDFS, RDF(S) or RDF-S can also found as abbreviations in publications).

RDFS and the variants of OWL are used to model an ontology. Depending on the type of (potential) semantic relations an ontology can contain more or less information. When there are only hierarchical relations included then the model can be referred to as taxonomy. With other (linguistic) semantic relations added, such as "narrower term" or "synonym term," a model becomes a thesaurus and so forth. What the type of model is called depends on the specific scenario it is used in. Sometimes, ontology is used to avoid restrictions on a model. A model can be subject to change. If it turns out that a formalization is referred to as taxonomy, but needs more complex relations, such as cardinality relations, then it might lead to confusion when it is called a taxonomy. Speaking initially of an ontology, when a knowledge representation is chosen, can therefore be more sensitive.

Braun et al. [BSWZ07] describe a typical process for the creation of an ontology. Ontologies are shared understandings of a particular domain. Two main perspectives on ontology engineering exist:

i) Ontologies are developed by *ontology engineering* experts. This is an expensive process since the costs for experts are typically high. For example, estimated costs for the Gene Ontology (GO) [ABB+00] have been at upwards of $16M [GTT+06]. The created ontologies express a shared understanding of experts, not of standard users that very likely have to work with them. The ontologies can still be error-prone if modeling experts are not at the same time *domain* experts.

ii) Ontologies are developed by *experienced users* who become ontology modeling experts right away. This seems to be an unrealistic assumption that users learn ontology modeling fast. Traditional, ontology modeling methodologies are treated as an overhead to their work process – time lag between the emergence of concepts and their inclusion in ontologies are far too big for ontologies to be useful. The ontology engineering process in the area of social semantic web is described in a book by Katrin Weller [Wel10].

Ontologies are therefore used in narrow domains such as medical science (e.g. RadLex[24]), biology, or some industrial applications. Other examples can be found at "The Open Biological and Biomedical Ontologies" portal [25].

In general, whether to use an ontology or an alternative knowledge modeling method, one has to consider the return of investment (ROI). A complex ontology is expensive to create and maintain and the benefits in comparison to something simpler might not be worth the effort.

Especially using ontologies at web-scale is a difficult task – in the scenarios of the Semantic Web. In the context of the semantic web, there are several languages that extend the simple graph format RDF with ways of expressing semantic relations: RDFS, RDFS-Plus and OWL Web Ontology Language (with variants). With these languages, one can form systems with different orders of logic.

[24] Can also be referred to as taxonomy – see section before.
[25] `http://www.obofoundry.org/`

Within the scope of an article Halevy, Norvig and Pereira argue about semantic web services that even the required technology is well understood, there are significant hurdles to deal with [HNP09]:

• *Ontology writing.* Simple ones are already created, such as Dublin Core. The long tail is expensive to create with current technologies.

• *Difficulty of implementation.* A simple web page is easy to create. For more complex ones where a service is compliant with Semantic Web protocols, an expert is needed.

• *Competition.* In some domains, it is hard to achieve a consensus over the used ontology. There are too many competitors who want to see their ontology become a standard.

• *Inaccuracy and deception.* Based on true premises, true conclusions can be inferred. With current algorithms, this is a basic task. Dealing with imprecise or wrong information (for example from criminals, such as spammers) at the moment is nearly impossible at least regarding large scale systems.

It is arguable that for most use cases a huge elaborated ontology is some kind of overkill. An ontology is expensive to create and maintain. The gained surplus value does not justify the costs. In most cases either a light weight ontology with a few relations, such as the ones contained in RDFs, or a simple thesaurus is sufficient.

Worth mentioning: If some kind of 'perfect' ontology – even for a narrow domain – is possible to create (or discover) is *very* unlikely. A formalization needs to deal with the problems of incompleteness and undecidability (see [Hin05]). Hence, only some limited models are possible. It is doubtful that there exists a general abstraction of things.

The term Ontology in information science is derived from its metaphysical brother, but both terms share a common notion. The idea that there are abstract forms and concepts of entities (in the real world) goes back to Plato with his theory of forms. "The (medieval) problem of universals" is one of many still unsolved enigmas in philosophy – see [Kli13]. This bone of contention is strongly related to this kind of discussion. An in depth investigation of its main issues lies beyond the scope of this thesis.

2.4. Others

A *topic map* represents topics (concepts), associations between the topics, and occurrences. Occurrences represent information resources relevant to a particular topic. There is an ISO-Standard ISO/IEC 13250 for topic maps first published in 1999. For serialization there exists a XML format called XML topic maps (XTM)[26]. For some background information about topic maps and examples see [PH02]. An example of a topic map version of the CIA world fact book can be found online[27]. Topic Maps are very similar to graphs in RDF and in most applications, topic maps are replaced with RDF versions.

[26] `http://www.topicmaps.org/xtm/`

[27] `http://www.ontopia.net/omnigator/models/topicmap_complete.jsp?tm=factbook.hytm`

Left out are other attempts to formalize relations between entities (and up to a certain point to capture semantics): DB Schema, XML-Schema, Entity Relationship (ER) Models and UML [IOF07] – see also figure 2.9. These information structures contain explicit (semantic) relations between entities (e.g. in ER model there is an entity *customer* with a n:m relation to another entity *products*) but these relations are seldom used for other purposes than modeling a system in order to be used inside an application. Still there are attempts to make tables of a database system available in a semantic web style. D2R developed at Freie Universität Berlin[28] is an example for a data base to RDF mapper.

Some variants of ontologies[29] have "real world" applications. This can be seen by the example of Google's knowledge graph[30]. This technology is built on top of freebase, a company acquired by Google in 2010. Freebase does not speak of an ontology.

The knowledge representation form is called a schema[31]. The creative commons version of freebase is available following linked data principles[32]. The facts contained in freebase are merged together with other sources to Google's knowledge graph. The knowledge graph is used to display overview information for certain search queries additional to found results in Google search. This feature offered for search queries where Google recognizes known entities. For example, a search for a movie returns informations about the movie, such as its cast and other details.

[28] `http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/`

[29] Or more one might call it structured data with typed links between instances.

[30] `http://googleblog.blogspot.co.uk/2012/05/introducing-knowledge-graph-things-not.html`

[31] It can be found at `http://www.freebase.com/schema`.

[32] See `http://www.w3.org/DesignIssues/LinkedData.html`

CHAPTER 3

Related and Existing Work

If your mind is empty, it is always ready for anything; it is open to everything. In the beginner's mind there are many possibilities; in the expert's mind there are few.

— Shunryu Suzuki (1904 – 1971), Zen Mind, Beginner's Mind

Contents

The following chapter provides an overview of the state of the art in the current research in the area of social tagging. The first section describes the published work around statistical analysis of tagging data. In a following section strategies for mapping tags to existing structured sources are presented.

3.1. Data Mining and Statistical Algorithm

Social tagging is un- or semi-structured. At least in the context of the Internet a huge amount of data is available. Therefore it is a natural approach to use data mining techniques to derive implicitly contained information. An increasing amount of literature is devoted to data mining on social tagging data.

3.1.1. Co-Occurrence Analysis and Clustering

One very popular approach for finding relations between tags is using the co-occurrence of tags, meaning two tags have been used together to annotate an object. This method is simplistic and can only deliver some kind of unspecific relation. The exact kind of semantic relation between two tags is very hard to determine and depends on the actual tagging practices of the single user in a tagging application.

In general, co-occurrence analysis is an old idea used in information retrieval. For example, Mandala et al. [MTT98] describe an approach where they used a Dice Coefficient to compute semantic relations between two terms. The computation is based on the number of occurrence of the individual word, respectively, the co-occurrence in a document collection. Having this statistical measurement for semantic similarity, they derive an automatically constructed thesauri.

Grahl et al. [GHS07] use a standard clustering algorithm (KMeans - see [ES00]) to find hierarchies in sets of tags. They use an iterating approach where first a coarse clustering with 300 clusters is computed. In a second step, each resulting cluster is split again into 20 clusters. In a last step, the two most representative tags from the clustering in the second step are merged into another clustering. Other examples for clustering tag sets can be found in [Ton06],[Ten06],[HJSS06a].

Cattuto et al. investigate several methods for automatically discovering relations between tags in social tagging applications [CBHS08b]. They test three different approaches: tag co-occurrence, cosine similarity of co-occurrence distributions, and FolkRank [HJSS06b]. In order to provide a semantic grounding of their folksonomy based measures, they try to map tags to synsets of WordNet. They compare the semantic similarity computed from WordNet with the ones determined through folksonomy based similarity measures. Not mentioned in this paper is how the actual mapping between words in WordNet and tags from their test data set has been conducted. This can be a very challenging task and depending on the chosen method the results may vary significantly.

A newer work by Markines et al. evaluates different similarity measures for emergent semantics of social tagging data [MCM+09]. First they introduce a mapping/ projection of the multinary relations represented by a folksonomy to a simple matrix. Based on the different resulting matrices, they evaluate several similarity measures (such as Jaccard or Cosine) against WordNet and DMOZ.

Zhang et al. use co-occurrence analysis in order to create a probabilistic, generative model simulating the users' behavior in assigning a tag to an URL. They apply HACM [HP98] – a rarely used clustering algorithm – to create a taxonomic hierarchy of tags.

In this work the results of the co-occurrence analysis are treated as an incomplete preprocessing step. Co-occurrence analysis is merely a fuzzy and unreliable source that needs further review by a user in order to be treated as some kind of real semantic structure. In prior work some kind of feedback loop is missing. On the other hand for some applications, there is no need for real semantic structures. Simple statistical correlations are sufficient enough.

3.1.2. Association Rule Mining

Association Rule Mining (ARM) is a very popular method for finding interesting relations in large sets of item sets. Agrawal et al. [AIS93] describe a well-established algorithm for mining association rules. Association rules have been extensively studied in the literature. However, most widely used algorithm are based on the original *a-priori* algorithm proposed by Agrawal et al. For more details on ARM see section 5.4.2.

Schmitz et al. [SHJS06] describe the idea of using association rules to determine *hypernymy and hyponymy relations* between tags in social tagging data. They have a strong emphasis on formal concept analysis and its usage in context of social tagging data. In this work the focus is on deriving less formal structures through ARM.

Heymann et al. [HRGM08] use ARM based in combinations with other measures for *link prediction* for social tags. They address the problem of "social tag prediction". Their target is to find additional, relevant tags for an object given a set of already assigned tags. Unlike Schmitz et al., they do not care about the type of semantic relations between tags. They just want to find additional tags for a certain object. Unlike the approach presented in section 5.4.2, they do association rule mining based on tag assignments for a certain *URL* without distinguishing between different users. This leads to a less personalized view on the social tagging data. The mined association rules are used for tag prediction. In this work the association rules are to infer tag bundles in order to arrange the tags in groups with the same topic.

Frequent Itemset Mining (which is a part of ARM) to find frequent tag patterns in the context of finding like-minded users has been targeted by Li et al. from Yahoo! [LGZ08]. Although this work does not address the problem of finding relations between tags it is still important for this aspect since the approach is analogues to attempts to find relations between tags with ARM.

3.2. Mapping of External Structured Sources

Many methods are conceivable for mapping tags to concepts in an ontology or other structured input. In general, a possible solution uses a combination of string distances, stemming algorithms, and comparison of graph structures – both of the ontology on the one hand and the relations between users, tags and resources in a folksonomy on the other hand.

Al-Khalifa et al. follow an approach where tags are first normalized, i.e. stemming algorithms are applied, then tags are grouped and general tags are removed [AKDG07]. In a last step, the resulting stemmed tags are mapped to (stemmed) concepts of an existing ontologies.

Laniado et al. [LEC07] investigate how an ontology (for their case the *noun hierarchy* of WordNet) can be integrated into a navigation interface for an existing folksonomy. When it comes to mapping tags to concepts in WordNet, they state that only 8% of the different tags in their data sample (480,000 different tags collected from 30,000 del.icio.us users) find a corresponding concept in WordNet. Regarding the most popular tags they observe

a higher percentage of matches. For tag disambiguation (homonyms), they use a semantic similarity metric based on the work of Pedersen et al. [PPM04].

Angeletou et al. [ASSM07] describe an approach called FLOR where tags are mapped to existing Semantic Web Entities. There are three steps in the process: *i) Lexical Processing:* The tag set is cleaned, meaning possibly irrelevant tags are removed. This includes for example non-English and tags with numbers. *ii) Sense Definition and Semantic Expansion:* Tags are mapped to WordNet concepts using the context of a tag if needed. Synonyms are identified. *iii) Semantic Enrichment:* The tags with their synonyms are mapped to corresponding concepts by considering string similarities and the neighborhood of the concepts.

Generally speaking, the described approaches are interesting and to some points promising. Depending on the characteristics of the social tagging data there might be problems with mapping tags to concepts in WordNet or an Ontology. If there is a certain domain specific vocabulary, e.g. tags used in a medical forum (e.g. medhelp[1]), a music community (e.g. lastfm[2]) or in an enterprise social software with special expressions and word usage, the overlap with concepts in WordNet is probably low. The mapping to narrow domains with elaborated ontologies might deliver more useful results.

Cattuto et al. [CBHS08a] compare automatically derived semantic similarities between tags of a folksonomy with the similarity of the corresponding concepts computed from the given graph structure in WordNet. Not mentioned in this paper is how the actual mapping between tags and concepts is achieved. They only state that "roughly 61% of the 10 000 most frequent tags in delicious can be found in WordNet". Depending on the method how a match between a tag and a WordNet concept is determined the actual numbers typically vary significantly. If stemming [MRS08] has been used there likely occur many false matches (false positives). Exact word mapping in contrast might lead to fewer matches. Having a string similarity measure (such as the Levenshtein distance or other – see [Nav01]) might have like stemming many false positives — depending on the applied threshold. Unlike Laniado et al. Cattuto et al. have only investigated the most frequent tags. The former have included the complete set of tags. In the long tail, the tags that are less frequently used, one most likely finds less matches.

Though structured sources may be a valuable input, the mapping between tags and items from the sources are most likely incomplete and error-prone to some extent. To conclude there is no gold standard available at the moment. If there will ever be one the future will show.

3.3. Social Tagging Clustering and Social Network Analysis

The (semi-) automated extraction of patterns in folksonomies has received quite some attention in recent years. Most of the work has been focused in the direction of finding relations between tags – such as hierarchies of meaning or semantic similarities. A number of efforts have also been made to discover communities according to folksonomy structure.

[1] `http://www.medhelp.org/`
[2] http://www.last.fm/

Cattuto et al. [CSB+07] explore network properties of folksonomies. They see a folksonomy as tripartite hypergraph consisting of user, tag and resource nodes linked by tag assignments, where a user annotates a resource with a tag. Among other investigations they extract a tag co-occurrence network, meaning they build a symmetric similarity matrix of tags, where each entry in the matrix corresponds to co-occurrence value of two tags.

Java et al. [JJF08] use NCut for simultaneously clustering user graphs (i.e. users connected by some form of relation) and user tags. The found clusters represent possible communities.

Grahl et al. [GHS07] do a conceptual clustering of a folksonomy. They use k-means and folkrank [HJSS06c] to compute conceptual hierarchies of tags. Another work for co-occurrence based clustering of tags to find related tags is described by Begelman et al. [BKS06]. Giannakidou [GKVK08] cluster tags combining co-occurrence with a semantic similarity. Shepitsen et al.[SGMB08] use a cosine measure between tag sets to get a set of resources for a given tag(s). The received resources are ranked according to the user interests generated from tag clustering in a separate computation. We have a similar understanding of the usefulness of tagging, but follow the further interpretation of topics and the links between communities and resources.

Brooks et al. [BM06] show that tags can be used to cluster related document – at least to a certain amount. They compare the tf-idf values of blog posts with the same tag. They based their study on technorati[3] data sets. While we follow their understanding of the importance of semantic content of tagging, we go further analyzing tag structure and topics.

Krause et al. [KSHS08] utilize tag vectors of users in the context of spam detection in folksonomies. They apply a Naïve Bayes classification to detect spam user where the tag vector of a user is one of 25 considered features. Because the tag vector of a user is a useful property for classifying spam, it confirms our assumption that (the interests of) a user can be described by his or her tag vector.

Li et al. [LGZ08] have developed an Internet Social Interest Discovery system (ISID) targeting to find users with common interests. In a first step it determines topics using frequent item set mining, in a second step it clusters found topics and maps users to clusters according to computed frequent tag patterns. In contrast to the later described approach they determine frequent pattern sets and cluster those persons who have used these patterns. We work on the tag vectors of the user directly which has the advantage that we do not have to compute frequent tag sets first. User might use similar tags but not in the same combinations.

Zarnadi et al. [ZC08] use the cosine similarity over the tag vectors of users to find persons with similar tags. This work is most similar to the later described approach except that they utilize the computed similarities between users to rank query results. In contrast to most of the previous work in this area, the focus in this work is on detecting communities directly through utilizing the tagging data of users – in conjunction with tagged resources.

[3] http://technorati.com/

3.4. Tagging Ontologies

In the context of the semantic web there are several existing approaches to *model* social tagging patterns within an ontology. Kim et al. provide a quite excessive overview of tagging ontologies [KSB+08]. Described are the following ones: Newman[4], SCOT, Knerr, Echarte, MOAT, NAO. In general, these are simple ontologies and there are no real major differences. Also for example SCOT builds on the tagging ontology of Newman and extends it with additional relations such as `scot:Coocurrence` and concepts such as a `scot:TagCloud`. To mention is that most of these projects are inactive. As for the case of SCOT the web site is even in the possession of a domain trader[5]. By querying sindice with "tag" or "tagging",[6] Newmans tagging ontology is the mostly used ontology for modeling tags. Based on these models and additional design consideration given by a blog post of Gruber [Gru05] a data model has been developed. The details of the chosen model are elaborated in chapter 6.

3.5. Thesaurus Editor

The creation of thesauri has been a laborious process before the modern age of computers. With computer support the creation process is promised to be much more efficient. This section provides some examples of desktop and web based thesaurus editors.

SKOSEd[7] [JBS08] is a Protégé[8] plugin for editing thesauri based on the Simple Knowledge Organisation System (SKOS [BM05]) ontology. Although this editor at first glance appears to be promising, one might get the impression that it is more in an alpha state. It will most likely not reach a state where it can be as a product in the near future.

Another tool for editing thesauri using SKOS is *TopBraid Composer*[9] by *TopQuadrant*. It is a general purpose modeling application for RDF (and OWL[10]). SKOS is therefore supported as well. Being Eclipse (SWT) based it is a desktop application. The target user group is domain experts having a comprehensive training in semantic web technologies and the TopBraid Composer.

Soboleo[11] [ZB07] is a web based tool for creating and editing SKOS based thesauri. A demo of the editor can be accessed via its homepage[12]. Figure 3.1 shows a screenshot of the Soboleo tool. The left column shows an excerpt of the thesaurus in a tree based navigation interface. The current selected term is *Supervised Learning*. In the column in the middle the existing relations are defined: A preferred label in English "Supervised Learning" and a German equivalent "Überwachtes Lernen". It is possible to specify alternative and hidden labels. Additionally, it is possible to provide a description of the term. Broader,

[4] `http://www.holygoat.co.uk/projects/tags/`
[5] As on March 20, 2011
[6] `http://sindice.com/search?q=tag`, as on March 20, 2011
[7] `http://code.google.com/p/skoseditor/`
[8] Protégé is an ontology editor and knowledge-base framework – see `http://protege.stanford.edu/`.
[9] `http://www.topquadrant.com/products/TB_Composer.html`
[10] `http://www.w3.org/TR/owl-ref/`
[11] `http://www.soboleo.com/`
[12] `http://tool.soboleo.com/`

Figure 3.1: *Soboleo Screenshot for the term "Supervised Learning".*

narrower and related topics can be defined in the same column under the tab *Relations* with an analog user interface representation. The terms have to be entered in the text field. The target user group seems to be domain experts with some introduction to the tool.

Poolparty[13] is a product developed by punkt. netServices GmbH, a company located in Vienna (AT). It is a thesaurus management system including a SKOS editor (see fig. 3.2). It is completely web based with a user interface implemented using YUI[14]. All data is stored in a RDF backend. Additionally, Poolparty has many different features, such as suggesting terms for a given text (e.g. from an URL).

Figure 3.2 displays a screenshot of the thesaurus user interface. The left column contains a tree with the terms of the thesaurus. The current selected term is Munich. There are terms on the same level: Berlin, Hamburg, and Leipzig. A broader Term is Germany which has a broader term Western Europe and so further. There is one narrower term "Odeonsplatz" defined. In the right column the characteristics of the concept can be viewed and modified. The in the screenshot visible tab "SKOS" contains the SKOS thesaurus relations that are specified for the current concept. There are two preferred labels present – Munich (en) and München (de).

In the tab "Linked Data" one can link the current concept to *Linked Data*[15] sources, such

[13] `http://poolparty.punkt.at/`
[14] `http://developer.yahoo.com/yui/`
[15] `http://linkeddata.org/`

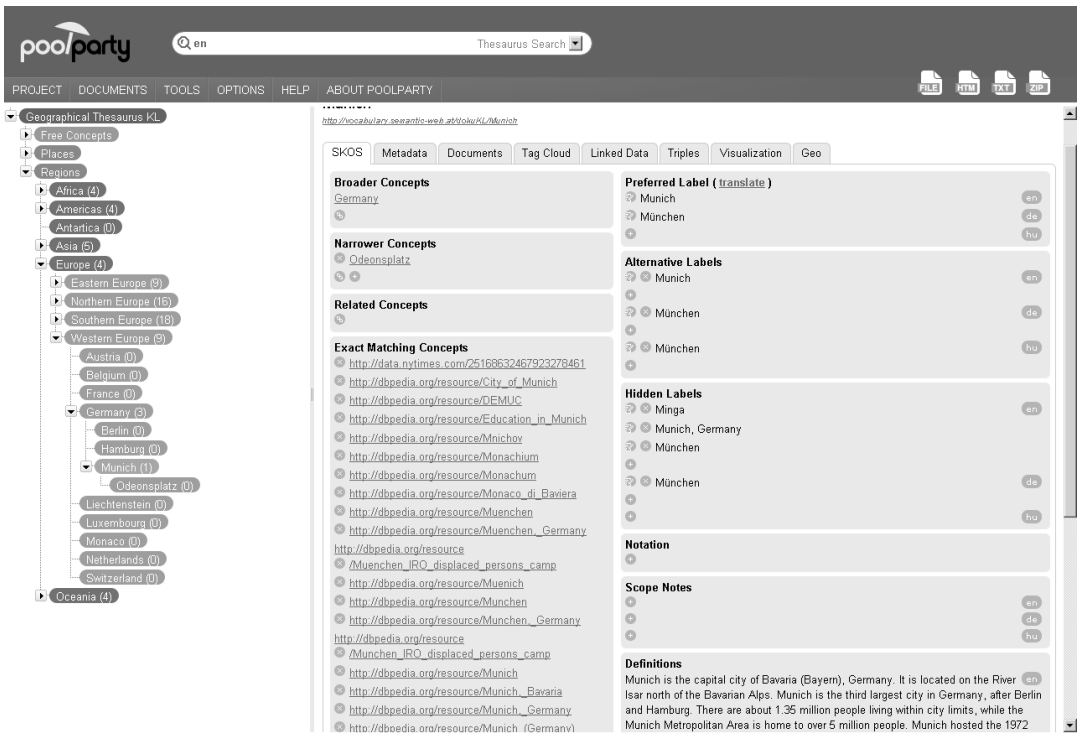


Figure 3.2: *Poolparty Screenshot for the term "Munich".*

as Freebase[16], Sindice, Geonames, DBpedia, and so own. The mapping of the current concept to concepts in external data sources show up in the "Exact Matching Concepts" area. The target user group seems to be domain experts with some introduction to the tool since it has many features with the corresponding complexity of usage.

In general, it appears as if there is no web based, easy to use thesaurus presently available. This impression has been solidified after talks to long year knowledge management experts at Siemens Corporate technologies. Poolparty and TopBraid Composer are undeniably powerful tools, but they are not that *easy to use* for an end user in the context of web 2.0 or enterprise 2.0. The thesaurus editor presented in section 5.7 works with plain tags and provides a simplistic web based interface.

[16]The corresponding freebase URL for Munich is `http://rdf.freebase.com/rdf/guid/9202a8c04000641f8000000004f37cd5`

CHAPTER 4

Use Cases and Requirements

When action grows unprofitable, gather information; when information grows unprofitable, sleep.

— Ursula K. Le Guin (* 1929), The Left Hand of Darkness

Contents

This chapter begins with an outline of scenarios and use cases in which the tagging framework finds its application. A list of requirements is elaborated and in a later part subsumed. Not all later implemented uses cases are described, but a selection of the most important ones. All use cases have emerged out of discussions, interview and some kind of brain storming with application owners and affiliated developers. The results are summed up in the following sections.

"*Use cases, stated simply, allow description of sequences of events that, taken together, lead to a system doing something useful. As simple as this sounds, this is important. When confronted only with a pile of requirements, it's often impossible to make sense of what the authors of the requirements really wanted the system to do.*" [BS02]

In general choosing the right granularity in which to describe an use case is a subjective task. Additionally, deciding what exactly to elaborate depends strongly on the individual taste of a software architect or respectively if one is working in or for a company on the

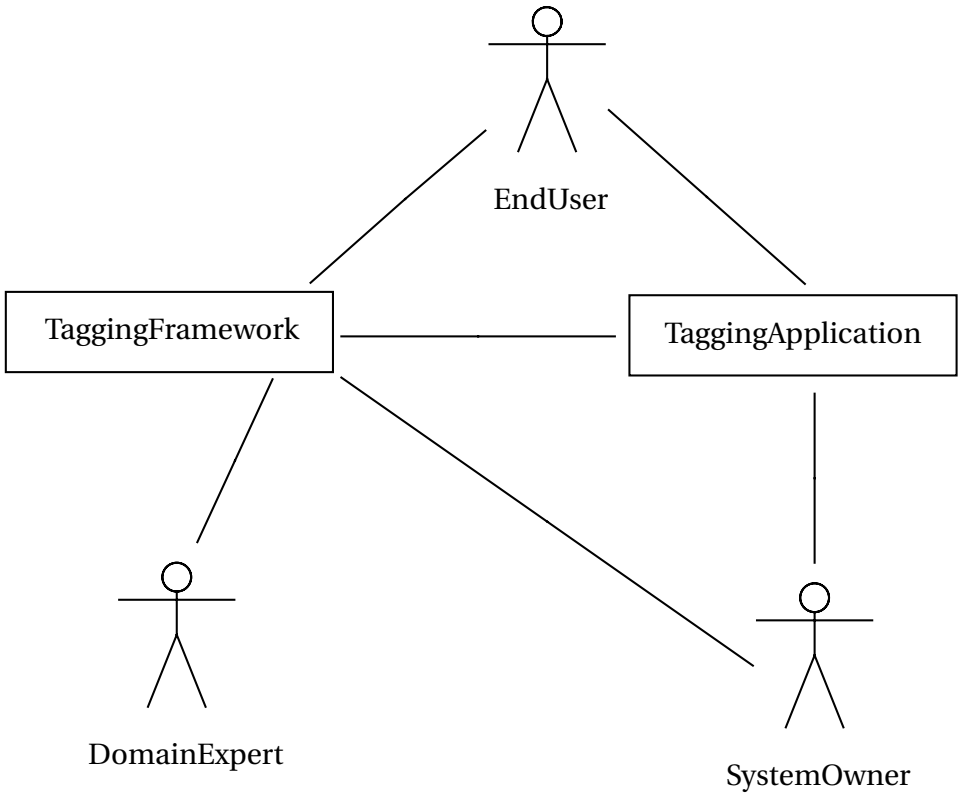


Figure 4.1: *Actors in the context of the tagging framework system: `TaggingApplication` is a application that makes use of the `TaggingFramework`. The latter is the proposed software artifact. `EndUser` and `SystemOwner` interact with both applications. The role `DomainExpert` interacts with `TaggingFramework` in order to create special thesauri.*

required style guide. Use case description follow the notations used in [BD03]. Recommendations made in [CBB+10] function as additional guidelines.

4.1. General Use Cases

In this section the general use cases in which a tagging framework can be utilized are elaborated. The relations between tags (either explicit ones defined by a user or automatically derived ones) are used to support the user interaction with a social application. Figure 4.1 depicts the actors that are involved in the use cases around the tagging framework. The following actors, entities that interact with each other in the setup – either human or an external system – are to identify:

- `TaggingApplication` (system): An application that supports tagging.
- `TaggingFramework` (system): The core system.
- `EndUser` (human): A standard user that interacts with a tagging application.

• `DomainExpert` (human): If the thesaurus editor is used to generate a more sophisticated thesaurus (for a special usage, such as to develop or establish a general terminology inside a department) this is done by a domain expert.

• `SystemOwner` (human): The person that manages or owns a tagging application.

4.1.1. Tag Suggestions during Tag Assignments

`TagSuggestionsTagAssignments` is one basic use case (see table 4.1). A user is supported during the process of assigning tags to an object. For the sake of simplicity it is assumed that there is a text field in which the user enters all tags that he or she wants to attach to an object. This use case is especially important, because if users are supported during the tag assignment process, the quality and quantity of the assigned tags is assumed to be increased. Having a "decent" tagging practice is crucial for social tagging to work. For example, spelling errors can be avoided.

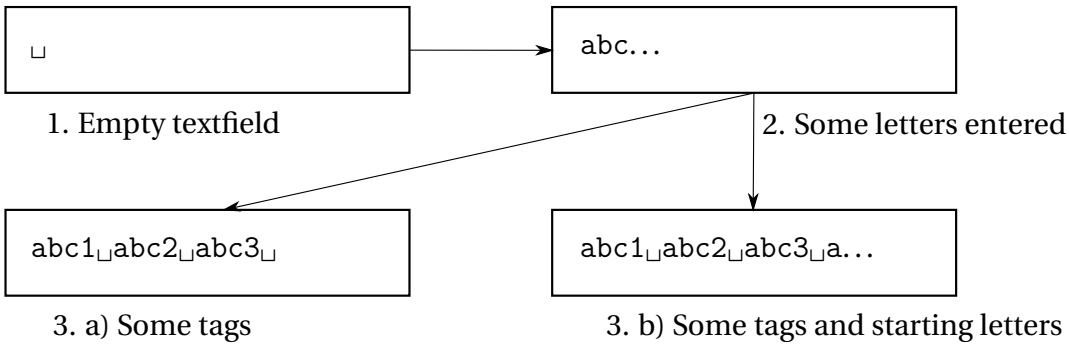


Figure 4.2: *Tag suggestions with different states of the text field.*

Figure 4.2 illustrates the different states of the text field. At first (1.) the text field is empty. In this case suggestions for an user can made taking different parameters into account: *(i)* the object to tag, *(ii)* the popular, and recent tags in the current or all systems *(iii)* the recent or frequently used tags of the current user. These suggestions are then typically presented to the user as clickable items below the text box[1]. A click on a suggested tag inserts this tag into the text field.

At a second state (2.) there are some letters for the first tag entered. The suggestions consider these letters accordingly – either as a sub-string or prefix match. In the last step *(3.)* there are several tags already given and either starting letters for the next tag are provided (3.b) – or not (3.a)). Having one or more tags provided, the results of chapter 5 can be used in the suggestion algorithm.

For the cases *(2.)* and *(3.a + 3.b)* normally a representation as drop down list is provided. A user can either select an item by a mouse click or by hitting the "return" key after having an item selected by a corresponding number of "arrow down" and "arrow up". This is a rudimentary pattern analogous to the pattern used in Google's search suggestions.
Considering this use case, following requirements can be identified:

[1] Not displayed in the figure.

Use case name	`TagSuggestionsTagAssignments`
Participating actor	Initiated by `EndUser`
Entry condition	An `EndUser` activates the edit tag assignment option.
Flow of events	1. An `EndUser` wants to create a tag assignment. He or she has to enter tags in an empty text field. No tags or letters are entered yet. General tags are recommended. 2. One or more letters are entered in the text field. Tags are recommended based on these letters. 3. One or more tags are already provided. Suggestions are based on that tag(s) and – if present – on the first letters for the new tag. 4. The `EndUser` submits the tags to the server.
Exit condition	The tag assignment is stored in the `TaggingApplication`.

Table 4.1: *Use Case: Tag Suggestions During Tag Assignments*

- *Cross application:* The tagging data of all available applications have to be considered.

- *Users should be identifiable:* For personalization of the suggestion algorithms it is important that the tagging data can be traced back to the individual users.

- *Easy and loose integration into heterogeneous tagging applications:* The tagging applications can have quite distinct characteristics. Hence the way the tagging framework must offer a generic way to be used by an application.

- *General tag suggestion algorithm:* The tag suggestion algorithm should be independent of the type of information item to tag. This is especially important when no textual content for the entity to tag is available. Examples are multimedia content, such as audio recordings, pictures, and videos.

4.1.2. Information Navigation

`Exploration` (see table 4.2) or alternatively information navigation is an use case in which a user does not specifically search for a certain information item, but tries to gain an overview of items available for a certain topic.

Figure 4.3 (a) depicts an instance of tag cloud[2] with the most important tag “Web 2.0”. A tag cloud is typically in alphabetical order beginning from top left to bottom right – the one in figure 4.3 (a) does not follow that convention. The font size of a tag reflects the

[2] modified from `http://en.wikipedia.org/wiki/File:Web_2.0_Map.svg`

importance (normally the frequency of usage) of a tag. Only the n-th most important tags are displayed.

A tag cloud can be used to get a quick overview of the topics inside an (social) application. This is especially useful for users that are new to an application and want to explore content. By observing the changes of a tag cloud over a certain period of time, a tag cloud can be used for trend detection [LK08].

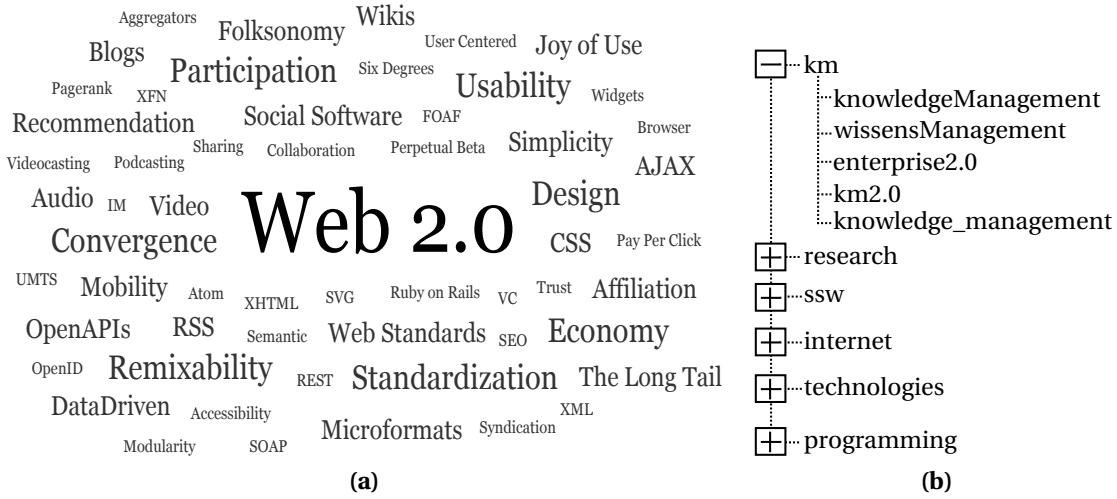


Figure 4.3: *Tag cloud (a) and tag based tree (b)*

Figure 4.3 (b) shows an example of a tag based tree. The tree metaphor is widely used, e.g. in file managers such as Windows Explorer or KDE Dolphin. Although a tree might not be the best choice of information representation (see 2.2.2), as user interface element it is still popular, and users are familiar with it [JPGB05]. The usage of standard user interface elements is highly recommended in the sense of "don't make me think"[Kru05].

Related to this use case, following requirements can be determined:

- *Cross application:* The tagging data of all available applications have to be considered.
- *Users should be identifiable:* This is especially needed for computing a personalized tag cloud or determining the top tags in a tag based tree.
- *Easy and loose integration into heterogeneous tagging applications:* The tagging applications vary in their characteristics.

4.1.3. Semantically Enhanced Search

The first part of the use case `TagSuggestionsSearch` as described in table 4.3 is similar to the use case `TagSuggestionsTagAssignments`. A user enters tags in a text field for querying information items. Corresponding suggestion are displayed in an adequate manner. A difference to the tag assignment use case is that not only the potentially "best" tags are suggested, but also "non-optimal" tags such as typos or unpopular spelling variants.

Use case name	`Exploration`
Participating actor	Initiated by `EndUser`
Entry condition	An `EndUser` accesses a type of overview page for social tagging data.
Flow of events	1. An `EndUser` wants to explore the information objects contained in `TaggingApplications`. An adequate navigable interface is displayed (such as a tag cloud or a tag based tree shown in figure 4.3). 2. By interaction with the user interface the user navigates along (related) tags in order to get an overview of the existing information objects.
Exit condition	The `EndUser` has gained enough information or leaves without the desired exploration.

Table 4.2: *Use Case: Exploration*

After having a user triggered a search the tag query can be optionally rewritten (either by the user or a system setting). This leads to more and/or better results. Additionally, alternative or further tags can be suggested to the user after a search – in the style of ‘`Did you mean ...`’.

Optionally, filters are needed to restrict the search result for a possible combination of date range for the tag assignment and a selection of users/systems.
Considering this use case, following requirements can be identified:

- *Cross application:* The tagging data of all available applications have to be considered.

- *Users should be identifiable:* This is important for incorporating social aspects into the search algorithms. Information items tagged by a colleague can be more relevant for a user than items tagged by users of a different part of a company. In some contexts the opposite is desired. For knowledge sharing amongst locally distributed fellow employees this can be a catalyst.

- *Easy and loose integration into heterogeneous tagging applications:* The tagging applications can have diverse characteristics.

4.2. Thesaurus Editor

Considering the elaboration in chapter 2 (concerning the lack of structure that comes with social tagging), there has been the need for a form of thesaurus editor identified. With the help of the editor a user can define relations between tags. The use case is described in table 4.4. Figure 5.21 on on page 108 shows a mockup of a possible user interface. Users

Use case name	`TagSuggestionsSearch`
Participating actor	Initiated by `EndUser`
Entry condition	An `EndUser` visits the search page.
Flow of events	1. An `EndUser` wants to search for tagged item. He or she has to enter tags in an empty text field. No tags or letters are entered yet. General tags are recommended. 2. One or more letters are entered in the text field. Tags are recommended based on these letters. 3. One or more tags are already provided. Suggestions are based on that tag(s) and if present on the first letters for the new tag. 4. It is possible to either enable the user to specify that a query should be automatically rewritten or make this the default behavior or not. 5. After a searching for a set of tags ways of rewriting the query are presented to the user.
Exit condition	The desired items are found or the user gives up.

Table 4.3: *Use Case: TagSuggestionsSearch*

must be enabled to define relations between tags in an easy manner. This is crucial for user adoption. Sketched in the figure is a web interface, where a user can select a tag and define relations to it by dragging other tags into boxes on the top representing different type of relations. These tags can either be ones contained in the suggestions listed in the boxes at the bottom or from the list of tags on the left that are filtered according to an user input. In the mock up the currently selected tag is “km”. “wm” is defined as a synonym tag, “enterprise2.0” as narrower tag to it.

4.3. Integration into Enterprise Tools

Prominently for an integration scenario in the context of enterprise tools is the way data is made accessible to these tools. Having desktop applications, this is achieved with some endpoint that provides the requested data in a serialized manner. For web based applications – which have been popular for a couple of years – an approach for offering widgets is a typical choice.

“*For nearly all systems, quality attributes such as performance, reliability, security, and modifiability are every bit as important as making sure that the software computes the correct answer. A software system’s ability to produce correct results isn’t helpful if it takes too long doing it, or the system doesn’t stay up long enough to deliver it, or the system reveals the results to your competition or your enemy.*”[CBB+10]

Use case name	`ThesaurusEditor`
Participating actor	Initiated by `EndUser` or `DomainExpert`
Entry condition	A `EndUser` or `DomainExpert` visits the thesaurus editor page.
Flow of events	1. An `EndUser` or `DomainExpert` is looking for a tag he or she wants to define relations to. 2. Another tag is selected to whom the first tag is supposed to have an relation to. 3. A kind of relation is selected.
Exit condition	The desired relation between two tags is stored inside the `TaggingFramework`.
Special requirements	If personalization of the recommendations is required, the user has to be identifiable.

Table 4.4: *Use Case: ThesaurusEditor*

In the context of integrating the tagging framework into enterprise tools following pseudo and nonfunctional requirements can be identified:

• *Cross application data aggregation*: Data from all social tagging applications must be collected.

• *Simple data exchange format:* Having a simple and generic data exchange format enables the export of social tagging data from nearly any tagging application. Hence the data exchange format has to be a common denominator across tagging application.

• *Little or no deployment effort:* It is crucial for the adoption of the tagging framework that there is little or no deployment costs for the side of the target platform.

• *HTTPS endpoint:* The HTTP is the de facto standard for services in a larger heterogeneous network landscape. HTTPS has to be used out of security/ trust reasons.

• *Same service level requirements (SLR):* SLRs, such as availability, performance, scalability and other that are required for a tagging application must be held to be the tagging framework as well.

• *Adequate data updates:* Updates to the social tagging data of the associated tagging applications must be dealt with accordingly – if possible in real-time.

• *Cross Site integration:* Issues with the same origin policy have to be considered – no standard XMLHttpRequest (XHR) is possible.

• *User interface skinable:* The representation of the user interface widgets must be skinable[3] and independent from JavaScript libraries that might lead to conflicts with ones used in the host tagging application.

• *Reasonable response times:* Requests made to the tagging framework have to be fulfilled in a reasonable response time – preferable less than 0.1 seconds[4]

• *Scalability:* With a larger number of tagging applications using the tagging framework the data stored in the tagging framework increases. Hence the storage and server architecture have to deal with more data and additionally more requests.

• *UTF-8 support:* This a simple requirement that is supposed to be self-evident these days. Unfortunately it is not, as many applications show.

4.4. Summary of Requirements

The summary of requirements consists of the requirements deferred from the described use cases and the non-functional requirements that are linked to being an operational system in an enterprise environment. Table 4.5 provides a summary of the identified requirements.

ID	Name	Description
1	Cross application	Tagging data of all applications must be considered.
2	Personalization	User information should be included and used whenever applicable.
3	Easy and loose integration	The tagging applications can have very different characteristics. Hence the tagging framework must offer a generic way to be used by a external applications. Skinning has to be supported.
4	Type of tagging data agnostic algorithms	Algorithms on social tagging data must be agnostic of the type of tagged entities.
5	Simple tagging data export format	The tagging framework must have access to the social tagging data contained in the various applications. Having very heterogeneous types of applications the export format must be rather simple and easy to implement.
6	Service Level Requirements (SLR)	The tagging framework must have the same SLRs as the applications that use it.
7	Cross Site Integration	Same origin policy issues must be avoided.

[3] The look and feel of a widget must be adoptable to the host application.

[4] "0.1 second is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result."[Nie93]

8	Reasonable Response Time	Especially in the area of user interaction response times should be depending on the use case not slower then 1 second – preferred less than 0.1 seconds.
9	Scalability	More associated tagging applications have to be supported. This includes an increased number of active users as well as more social tagging data that has to be managed.

Table 4.5: *Summary of requirements.*

CHAPTER 5

A Social Tagging Framework

Anything that can be automatically done for you can be automatically done to you.

— Wyland's Law of Automation

Contents

In this chapter the general approach for a tagging framework that offers services to external applications is presented. This includes an architectural design of a tagging framework as well as methods for extending folksonomies with relations. At first, a general model for a folksonomy is specified. Then the test data sets the implementation of a tagging framework has been evaluated against are described.

Later the proposed approaches are introduced. First methods for extracting structural information out of folksonomy through statistical analysis methods are outlined. Then ways for mapping tags to an existing structured information source are discussed. Both general approaches reflect the *semantic challenge* and the *hidden structure challenge* formulated as research issue in chapter 1. Via statistical analysis hidden structures are crystallized – including (semantic) correlations between tags (Co-Occurrence Analysis and Association Rule Mining) as well as implicit structures involving persons (Discovering Communities of Interest and Urgent Request Channeling). The algorithm for suggesting tags with a full text as input does not target any of the formulated challenges directly. Tag suggestions assist users in avoiding typos and spelling mistakes in tags. Additionally, it supports establishing a common tagging practice because the usage of more frequently used tags is encouraged.

The last section introduces a web based thesaurus editor where a user can define relations between tags manually. This a very central and interesting component of the tagging framework (see end of chapter 1) and therefore deserves special attention.

5.1. Architectural Design

Figure 5.1 displays a plain overview of the targeted architectural approach. The tagging applications itself remains untouched – as far as possible. Each application manages its social tagging data. The tagging framework offers services to the tagging application and aggregates the social tagging data. Individual standard tagging services remain in the tagging applications. This depends on the characteristics of social software application.

5.2. Folksonomy Model

In this work folksonomy is defined similar to the definition given by Hotho et al. [HJSS06c]:

Definition 5.1 (Folksonomy) *A folksonomy is a tuple* $\mathbb{F} := (U, T, R, S, Y)$ *where*

- U, T, R, S *are finite sets, whose elements are called users, tags, resources and system respectively*
- Y *is a quarnary relation between them, i. e.* $Y \subseteq U \times T \times R \times S$*, called tag assignments.*

A folksonomy contains multinary relations between user, tags and resources. Additionally, the system, meaning the tagging application in which the individual tag assignment

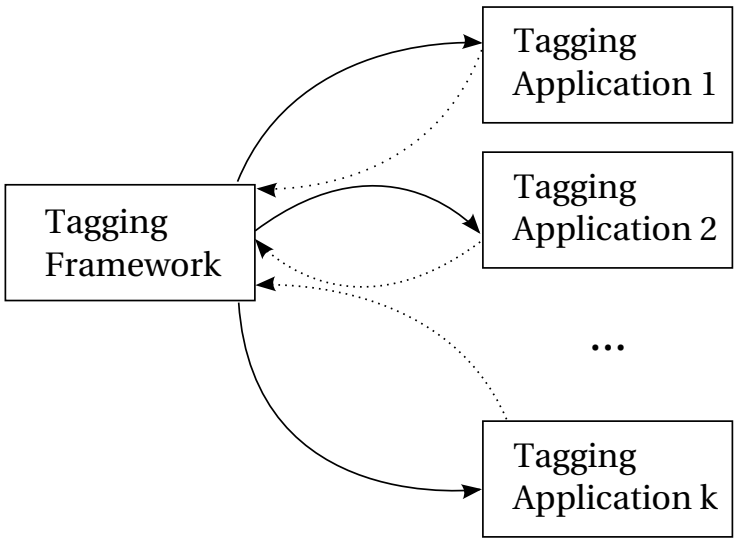


Figure 5.1: *Tagging Framework Architecture: A tagging framework offers services to external tagging application. These tagging application use the Tagging Framework as service and have to be adopted as less a possible.*

occurred, is also considered [Gru05]. Having the system part included enables one to distinguish the tagging application where a tag assignment has occurred. Without a system part one cannot distinguish between resources that are referenced by in multiple systems. This is for example the case with a wiki page and a bookmark to the wiki page. Both times the same R (the URI) is used.

Left out in this definition is the date on which the tag assignment has occurred. The time is not important for the definitions used in section 5.4.1. A user cannot assign the same tag to an object at different points in time. Hence, a tag assignment is uniquely identified by a user, tag, resource and system. For other use cases the date is an important meta-data though. For example, in tag suggestions tags that have not been used for a long time should get a corresponding penalty in the resulting suggestion ranking.

Figure 5.2 displays the model in an UML class diagram. A TagAssigment (Y in definition 5.1) consists of a User, Tag, Date, Resource and a System. As already mentioned the date of a tag assignment is not important for most use cases. For other uses cases such as showing activity streams or using tag assignment for trend detection, this meta-datum is important.

Table 5.1 contains several examples of tag assignments, such as ta1 that is represented by the tuple (u1, “ajax,” r1, s1). Typically users are represented by an unique identifier, such as an email address or an identifier provided by a user directory. The tag is a simple string, resources are identified by an URI [BLFM05] and systems can be for example identified by a host name.

Tag Assignment	User	Tag	Resource	System
ta1	u1	ajax	r1	s1
ta2	u1	web	r1	s1
ta3	u1	css	r1	s1
ta4	u2	javascript	r2	s1
ta5	u2	web	r2	s1
ta6	u2	css	r2	s1
ta7	u3	design	r3	s1
ta8	u3	photoshop	r3	s1
ta9	u3	web	r3	s1
ta10	u3	tutorial	r3	s1

Table 5.1: *Example Tag Assignments*

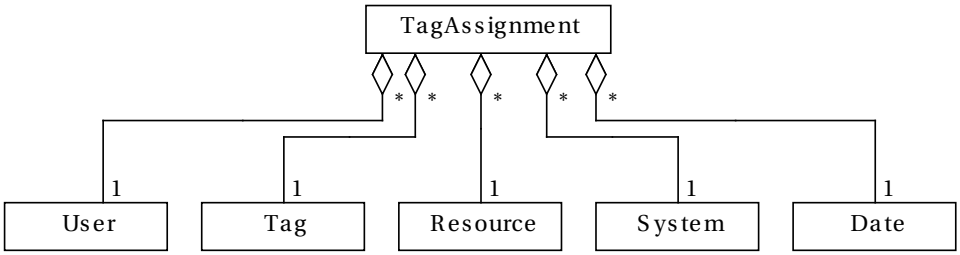


Figure 5.2: *Tag Assignment Model in an UML class diagram.*

5.3. Test Data Sets

For testing different approaches, two data sets were selected: Social tagging data available from social software tools inside the intranet of Siemens and a bookmark data set aggregated from Delicious. Both data sets contain tags from user generated content. Hence, both can contain messy data. The problems described in chapter 1 can be observed.

5.3.1. Siemens

The Siemens data set consists of social tagging data from the three major social software applications inside the Siemens intranet: *Blogosphere*, *Wikisphere* and *TechnoWeb*. Figure 1.2 displays the banners for these three applications. Inside Siemens, social software applications are used to foster knowledge exchange and provide tools for collaboration of Siemens employees. For Siemens, social software is an important part of the "working environment of the future". Given a distributed IT landscape there is a need for new approaches that manage the complexity of information provision and can proof their efficiency in daily work tasks. The vision for a personalized information provision in corporate business processes is as follows: Not the employee has to search for relevant information; rather an information object "finds" the recipients, who are potentially interested in that piece of in-

formation for their current working context. Instead of static profiles of interest, dynamic profiles based on user interaction and activity streams will be applied.

The different applications have been introduced in more details in chapter 1. Only the characteristics of the used data set is described here.

Data Characteristics of Aggregated Siemens Data

Pos.	Tag	frequency	Pos.	Tag	frequency
1	infrastructure	801	26	siemenstv	141
2	shs_it	767	27	index	128
3	innovation	409	28	feature	127
4	web2.0	375	29	responsibility	127
5	communication	297	30	blog	126
6	faq	263	31	rhapsody	125
7	technology	260	32	münchen	123
8	fp	224	33	groc_transformation	122
9	video	215	34	sustainability	122
10	caring_hands	213	35	green	120
11	cc	207	36	glossary	119
12	project management	205	37	google	115
13	siemens	187	38	mind	114
14	wiki	185	39	transformation_blueprint	109
15	management	183	40	blogs	109
16	mgw	182	41	mind_faq	109
17	tools	182	42	microsoft	107
18	abbreviation	179	43	social	106
19	quickfix	176	44	web	101
20	sharepoint	174	45	sector_commodity_engineer	101
21	atca_media_gateway	171	46	sqm	101
22	collaboration	161	47	quality	101
23	energy	157	48	sap	100
24	development	157	49	strategy	99
25	blogging	157	50	social_media	97

Table 5.2: *Top 50 tags Siemens data set.*

The Siemens data sample was exported on 11th of January 2011. The first tag assignment is from 25th of August 2005. The last tag assignment is from 11th of January 2011. There are 42,440 tag assignments from 1339 individual users. 9348 different tags are applied to 11,512 different resources.

The 50 most frequent tags for the Siemens data set are listed in table 5.2. Table 5.3 contains statistical characteristics of the frequency tags are used in the Siemens data set. The

Measure	Overall frequency of tags	Number of tags per user
Minimum	1	1
Maximum	801	1,621
μ	4.54	15.02
Median	1	5
Variance	340.97	3,184.67
σ	18.47	56.43
n	9,348	> 10,000

Table 5.3: *Statistics for the Siemens data set. The Second column describes the frequency of tags in general. The third column contains the tag distribution per user.*

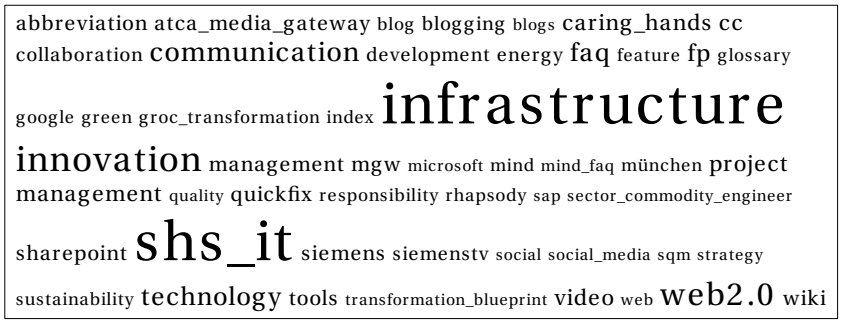


Figure 5.3: *Tag cloud Siemens data set.*

Second column describes the frequency of tags in general. The third column contains the tag distribution per user.

Figure 5.3 shows are graphical representation of these most frequent tags in the form of a tag cloud.

5.3.2. Delicious

Additionally, to the Siemens test data, a second data set has been harvested from delicious. During a period of about eight weeks (end January - March 2009), the RSS feeds for the bookmarks of 2300 randomly chosen users of the popular bookmarking service Delicious were periodically fetched. The data was aggregated from delicious rss feeds (using a java program with apache httpclient[1] and rome[2] as libraries and MySQL as database). During this time period three accounts were removed and 27 users did not assign any tags, which leads to a total number of 2270 investigated users. The total number of resources for all users aggregate to 462,415 (with duplicates), of which 345,674 unique resources form R.

Table 5.4 contains statistical characteristics of the frequency tags are used in the Delicious data set. The Second column describes the frequency of tags in general. The third

[1] `http://hc.apache.org/httpclient-3.x/`
[2] `https://rome.dev.java.net/`

Figure 5.4: *Tag cloud Delicious data set.*

column contains the tag distribution per user.

Measure	Overall frequency of tags	Number of tags per user
Minimum	1	1
Maximum	39,467	3,328
μ	30.17	238.48
Median	3	165.5
Variance	180,011	73,414.47
σ	424.29	270.95
n	97,521	2,270

Table 5.4: *Statistics for the Delicious data set. The Second column describes the frequency of tags in general. The third column contains the tag distribution per user.*

The number of resources tagged by a user ranged from a single resource (three users) to over 1,000 different resources (eight users). Surprisingly these eight users were not spam and also no spam resources in the sample set could be detected. On average there were 201 resources per user. For some more details about the data set see [KVZ09].

Table 5.5 contains the fifty most frequently used tags in the Delicious data set.

There were 2,942,633 tags in total and 97,522 uniquely different tags forming the set of tags T. While 27 users did not display any tagging activity, the average number of different tags per user was 201. The median of different tags per user was 165. One hyperactive user registered 3328 distinct tags. For a graphical representation of the number of distinct tags against the number of users having used the corresponding number of distinct tags see figure 5.5. A typical long tail distribution of tags [Ton06] has been observed. The first bookmark origins from 05/01/1989 which is very likely a data error considering the age of the Internet.

Pos.	Tag	frequency	Pos.	Tag	frequency
1	design	39467	26	development	12157
2	webdesign	37677	27	webdev	12059
3	tools	35718	28	jquery	11248
4	photography	25979	29	2009	10672
5	web	24712	30	technology	10628
6	tutorial	23176	31	google	10350
7	web2.0	22689	32	news	10260
8	software	22564	33	Education	10154
9	reference	21963	34	politics	9843
10	blog	21505	35	linux	9781
11	inspiration	19533	36	social	9426
12	video	19354	37	marketing	9407
13	programming	19330	38	socialmedia	9296
14	twitter	18954	39	internet	9264
15	tips	16816	40	online	9220
16	howto	16615	41	wordpress	9097
17	css	16525	42	research	8998
18	resources	16267	43	food	8768
19	free	15880	44	opensource	8475
20	tutorials	14829	45	photo	8442
21	music	14264	46	media	8411
22	javascript	13566	47	flash	8264
23	business	13415	48	windows	8220
24	Photoshop	12357	49	science	8158
25	art	12274	50	search	8126

Table 5.5: *Top 50 tags Delicious data set.*

5.4. Data Mining and Statistical Algorithms

Social tagging data can be exploited by applying data mining algorithms that work on set of items. First a rudimentary algorithm for analyzing co-occurrences of tags is described. In second section an approach for applying association rule mining on social tagging data is presented. Social tagging is user centered. Hence a method for determining interest based user groups has been developed.

5.4.1. Co-Occurrence Analysis

One very popular approach for finding relations between tags is using the co-occurrence of tags, meaning two tags have been used together to annotate an object. This method is simple and can only deliver some kind of unspecific relation. The exact kind of semantic

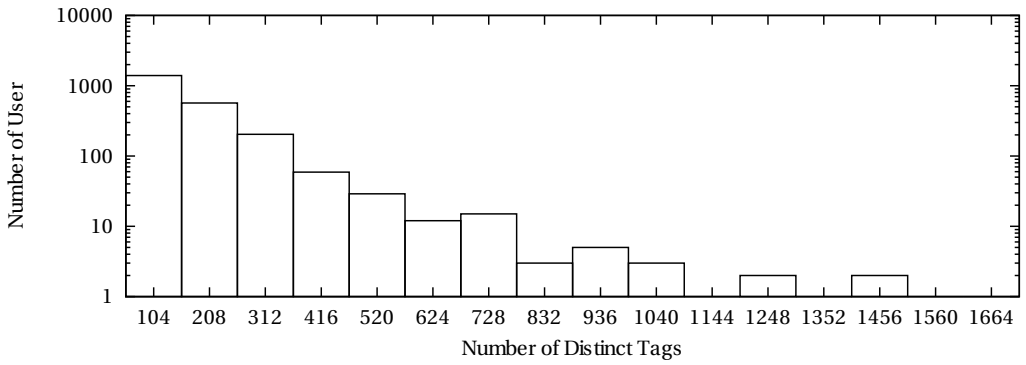


Figure 5.5: *Histogram: Number of distinct tags vs number of user.*

relation between two tags is very hard to determine and depends on the tagging practices of a single user in a tagging application.

At first an utility function called *cover* is defined. The function collects all user-resource tuples from all tag assignments where at least one user has applied a tag to a single resource. The frequency of a tag A in a tagging system is equivalent to the cardinality of $cover(A)$.

Definition 5.2 (Cover) *Let $A \in T$ be a tag, then*

$$cover(A) = \{(u,r) \in U \times R \mid \exists u \in U : (u,t,r,s) \in Y \wedge t = A\}$$

defines the finite set of user-resource tuples that have been tagged with A.

Using table 5.1 for example $cover(\text{css}) = \{(u1,r1),(u2,r2)\}$ and $cover(\text{ajax}) = \{(u1,r1)\}$.

Having *cover* defined the absolute co-occurrence of two tags – meaning two tags have been used together in a tag assignment – can be defined as followed:

Definition 5.3 (Absolute Co-Occurrence) *Let $A,B \in T$ be tags, then the absolute co-occurrence AC is defined as:*

$$AC(A,B) = |cover(A) \cap cover(B)|$$

This is the most popular approach for computing relations between tags in recent work probably because it is easy and efficiently to compute.

Example (with data from table 5.1): $AC(\text{css},\text{ajax}) = |cover(\text{css}) \cap cover(\text{ajax})| = |\{(u1,r1)\}| = 1$

Its major drawback is the fact that the absolute frequency of a tag is not considered adequately. This means having three tags A, B, C. A is a very frequently used tag. B and C are less frequently used. Typically $AC(B,A)$ is greater than $AC(B,C)$, although B and C are more closely related. In general, the AC may lead to distorted results in the interpretation of the strength of a co-occurrence relation.

An alternative method for computing relations between tags can be formulated with the relative co-occurrence in which the frequency of the individual tag is also taken into account. The relative co-occurrence is a special form of the Jaccard similarity coefficient [HKP06].

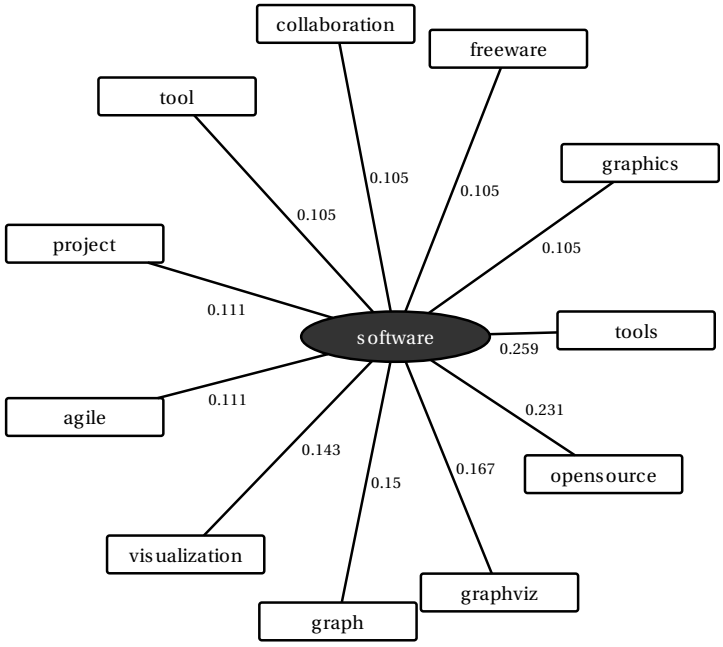


Figure 5.6: *Tag Relations: Eleven tags with the highest RC value to the tag "software". They are ordered clockwise beginning with "tools" (at three o'clock). The lengths of the edges are proportional to the computed similarity of two tags.*

Definition 5.4 (Relative Co-Occurrence) *Let $A, B \in T$, then the relative co-occurrence RC is defined as:*

$$\begin{aligned} RC(A,B) &= \frac{|cover(A) \cap cover(B)|}{|cover(A) \cup cover(B)|} \\ &= \frac{|cover(A) \cap cover(B)|}{|cover(A)| + |cover(B)| - |cover(A) \cap cover(B)|} \end{aligned}$$

A potential semantic similarity between two tags A and B can be estimated by the corresponding $AC(A,B)$ or $RC(A,C)$ value — the higher the RC or AC value of two tags the stronger the potential semantic proximity of those two tags.

Example (with data from table 5.1):

$$RC(\text{css}, \text{ajax}) = \frac{|cover(\text{css}) \cap cover(\text{ajax})|}{|cover(\text{css})| + |cover(\text{ajax})| - |cover(\text{css}) \cap cover(\text{ajax})|} = \frac{1}{2+1-1} = 0.5$$

The already mentioned advantage of RC in contrast to AC can be illustrated by the RC values of the tags "design," "photoshop" and "web". $RC(\text{design}, \text{photoshop}) = 1$ is greater than $RC(\text{design}, \text{web}) = 1/3$. Also $RC(\text{photoshop}, \text{web}) = 1/3$ is less than $RC(\text{design}, \text{photoshop})$. The mutual absolute co-occurrence for all three tags is 1 and does not take the absolute frequency of an individual tag into account.

Fig. 5.6 shows an example for similar tags based on the relative co-occurrence. The graph is deduced from the tagging practice of a single delicious user – only tag assignments of

this special user have been considered. In the center is the tag “software” and in clockwise order (beginning with “tools” – at three o’clock) the eleven most similar tags determined by the RC value are displayed. For this user “software” and “tools” have the strongest relation whereas “software” and “graphics” are less related.

5.4.2. Association Rule Mining

Using association rule mining (ARM) in order to mine tag bundles is another approach which has been tested during the work on this thesis. Although no expedient hypernymy and hyponymy relations between tags could be derived by applying ARM using this method to compute tag bundles has shown to be promising. The details are described in the following section. The results were published in [KVEL10].

Social tagging follows very heterogeneous and individual usage patterns. Each user and each application has different characteristics. In general one can observe a long tail distribution of tag frequencies [Ton06]. This means that users tend to employ some tags very frequently and a huge number of tags very infrequently. Having many different tags leads to information scattering and therefore navigational interfaces based on unfiltered tags quickly become very inefficient. This applies both to simple lists of tags that are either in an alphabetical order or sorted based on the frequency of a tag. Tag clouds, as alternative representation, display only an excerpt of the more commonly used tags.

A related characteristic is introduced by the variety of reasons motivating an users tagging practice in an information system. Golder and Huberman [GH06] identify seven possible functions. Similar, Marlow et al. [MNBD06] describe incentives and motivations for users to annotate resources with tags – see chapter 1. From these considerations one can see that the usage of tagging is manifold. The *personal information* aspects are easily recognized by taking the individual nature of social tagging into account, especially the motivation “Future retrieval”. For a lot of cases a user wants to organize his or her personal information items (bookmarks, pictures, books, bibliography, notes, etc.).

Association Rule Mining

Association Rule Mining is a popular data mining method. Association rules have been extensively studied in the literature. However, most widely used algorithms are based on the original *Apriori* algorithm proposed by Agrawal et al [AIS93]. A typical application of ARM is the analysis of transaction data recorded by point-of-sale (POS) systems in supermarkets. The results are, e.g. integrated in the decision process of how to arrange items in a supermarket. This sometimes leads to surprising results. Observations in a grocery store show that people who buy diapers also buy beer [FBH00]. Based on that insight diapers can then be placed together with beer.

In analogy to grouping items that are frequently bought together, a target in this application is to group tags that are often used in conjunction so that the represented resources (bookmarks, documents, etc.) can be more easily accessed.

Association rules are in general of the form:

$$X \longrightarrow Y\,[\text{Confidence}, \text{Support}]$$

Confidence stands for $P(Y|X)$, meaning how likely Y is given X. *Support* is the number of transactions containing both X and Y. A transaction for the super market example consists of the items a customer has bought together at exactly one visit.

Given a minimum support, association rules can be computed with the algorithm described by Agrawal et al. The algorithm is listed in pseudo code in algorithm 1. This version is adopted from the one that can be found in the English version of Wikipedia [Wik12][3]. It does not contain details, such as the used candidate generation part (line 5), but is more clearly structured than other versions. T is a database with transactions, ϵ is a support threshold. The algorithm makes several runs through the data set until it does not find frequent itemsets for a certain length. L_k is the set of all frequent itemsets with length k. C_k refers to the candidate itemsets with length k that have to be considered.

Algorithm 1 Apriori

```
    function Apriori(T, ε)
        L_1 ← { large 1-itemsets }
        k ← 2
        while L_{k-1} ≠ ∅ do
5:          C_k ← {c ∈ a ∪ {b} | a ∈ L_{k-i} ∧ b ∈ ⋃L_{k-1} ∧ b ∉ a}
            for transactions t ∈ T do
                C_t ← {c | c ∈ C_k ∧ c ⊆ t}
                for candidates c ∈ C_t do
                    count[c] ← count[c] + 1
10:             end for
            end for
            L_k ← {c ∈ C_k | count[c] ≥ ε}
            k ← k + 1
        end while
15:     return ⋃_k L_k
    end function
```

A *tagging*, meaning a user has tagged a resource with several tags, is a special type of transaction. Based on these *tagging* transaction rules of the form $\{x_{i1}, \ldots, x_{ik}\} \longrightarrow y_i\,[\text{Confidence}, \text{Support}]$ where $x_{ij} \in T$ and $y_i \in T$ are computed.

By restricting the set of generated association rules with thresholds for support (*minSup*) and minimum confidence (*minConfidence*) only a selection of rules is considered as input for the target tag bundles. In a final step the rules are joined into bundles if they share the same head y_i.

Computed association rules have the following form, for example:

$$\{\text{management}, \text{crisis}, \text{failure}\} \longrightarrow \text{finance}\,[\,0.5,\, 60\,]$$

[3] The state of the wiki page, at the time the page was visited, surely needs cleanup, but the pseudo code is valid.

or

$$\{\text{history}, \text{trust}\} \longrightarrow \text{finance}\,[\,0.7,\,20\,]$$

In this example both rules have the same head and are therefore merged into a bundle:

$$\{\text{management}, \text{crisis}, \text{failure}, \text{history}, \text{trust}\} \longrightarrow \text{finance}$$

Depending on the individual tagging behavior and the parameter thresholds, there are a number of tag bundles discovered. The resulting tag bundles can function as a suggestion for a user on how to organize his or her *personal* tags.

Tag Bundles

A tag bundle consists of a head with a frequent tag being the common denominator for the linked resources. Additionally, there is a set with sub-tags which are more specific and can reflect different aspects of the resources in question. An example can be seen in figure 5.7. By using the relationship between tags associated with a resource, an aim is to discover tag bundles in user tagging. Not only do tags elucidate the content of a document, but the same tags are also in a semantic relationship with each other simply by being used to describe the same source of information.

Tagging behavior, as opposed to keyword extraction from text, is geared towards the complete and succinct description of content and the organization of documents for a keyword based search. A user will distribute tags on several granularities, for instance “java” to describe more general concepts and “bean” for specific uses.

By discovering the usage of such broadly descriptive terms in combination with other more specific terms, a *tag bundle* can be created which gathers all specifics to a general term into a set. This set reflects the conceptual taxonomy and associated documents from a user perspective.

Tag bundles stem from personal information management, e.g. used when organizing a blog (can consist of many posts when used as a notepad) or bookmarks [Ehm10]. This creates an individual tag space associated with every user which is dependent on his or her point of view. While general concepts are easy to reconcile across world views, specifics tend to be perceived in a slightly different light. Tag bundles reconcile concepts with each other by offering the general terms as a bridge between user perceptions.

Conversely, the same tag can appear in different tag bundles of a single user, reflecting the different meanings of a term. Such overlapping bundles shows tags used in different contexts, e.g. java can be placed in a bundle books together with other book related tags such as “tutorial” or “toread” as well as in a bundle programming together with tags such as “tools” or “tips”. Additionally, the problem of ambiguous meanings of tags, such as challenge presented by acronyms and homonyms (see section 2.2.2), must be considered. Depending on the context, a tag with several meanings can occur in several tag bundles, each representing different topical collections of resources.

Evaluation

The ARM approach has been tested on social tagging data from the Siemens data set. The results are very promising, but since the amount of available data does not nearly reach the numbers obtained in Internet usage, the evaluation of the approach is done on data collected from Delicious.

Preprocessing For cleaning the social tagging data only a conversion to lower case has been applied. Stemming was not used since it depends on the language of the tags, and first tests have shown that language detection on tags remains inconclusive. This may be because tags are typically very specific and often just a simple phrase. No thesaurus has been utilized because the overlap of tags and a thesaurus such as WordNet is expected to be low. When it comes to mapping tags to concepts in WordNet, Laniado et al. state that only 8 % of the different tags in their data sample (480,000 different tags collected from 30,000 Delicious users) find a corresponding concept in WordNet [LEC07].

Association Rule mining Three different parameter thresholds have been tested: (0.5, 8), (0.7, 4) and (0.9, 3) (confidence, minimum support). These parameter thresholds were chosen based on preliminary experiments where these thresholds seemed to be the most practical.

Bundles Depending on the characteristics of the individual tagging data, different numbers of users with bundles were computed: For (0.5, 8): 825, (0.7, 4): 1207 and (0.9, 3): 1330. It was not possible to derive tag bundles for users if they did not repeatedly use more than one or two tags per resource or if they used many different tags. Association rule mining cannot be utilized to derive tag bundles in these cases.

Parameter	Feature	Min	Max	μ	Median	Variance	σ
0.5, 8	tags per bundle	1	127	4,09	13	37,24	6,1
	bundles per user	1	91	5,88	1	70,32	8,39
0.7, 4	tags per bundle	1	246	5,48	4	73,06	8,55
	bundles per user	1	225	9,67	6	263,39	16,23
0.9, 3	tags per bundle	1	303	6,24	5	92,37	9,61
	bundles per user	1	307	12,74	2,5	479	21,89

Table 5.6: *Statistics for generated tag bundles.*

Table 5.6 shows the statistics for the generated tag bundles. Min stands for minimum, Max for maximum, μ denotes the sample mean, and σ[4] is the standard deviation. For (0.5, 8) there tends to be less tags per bundle, since the threshold for the minimum support is rather high. There are fewer rules created, but rules are more easily accepted as a base for a

[4]Note that in statistics μ and σ are normally used to refer to the characteristics of the population, not the sample. However, in this evaluation a distinction between the population and sample is not necessary.

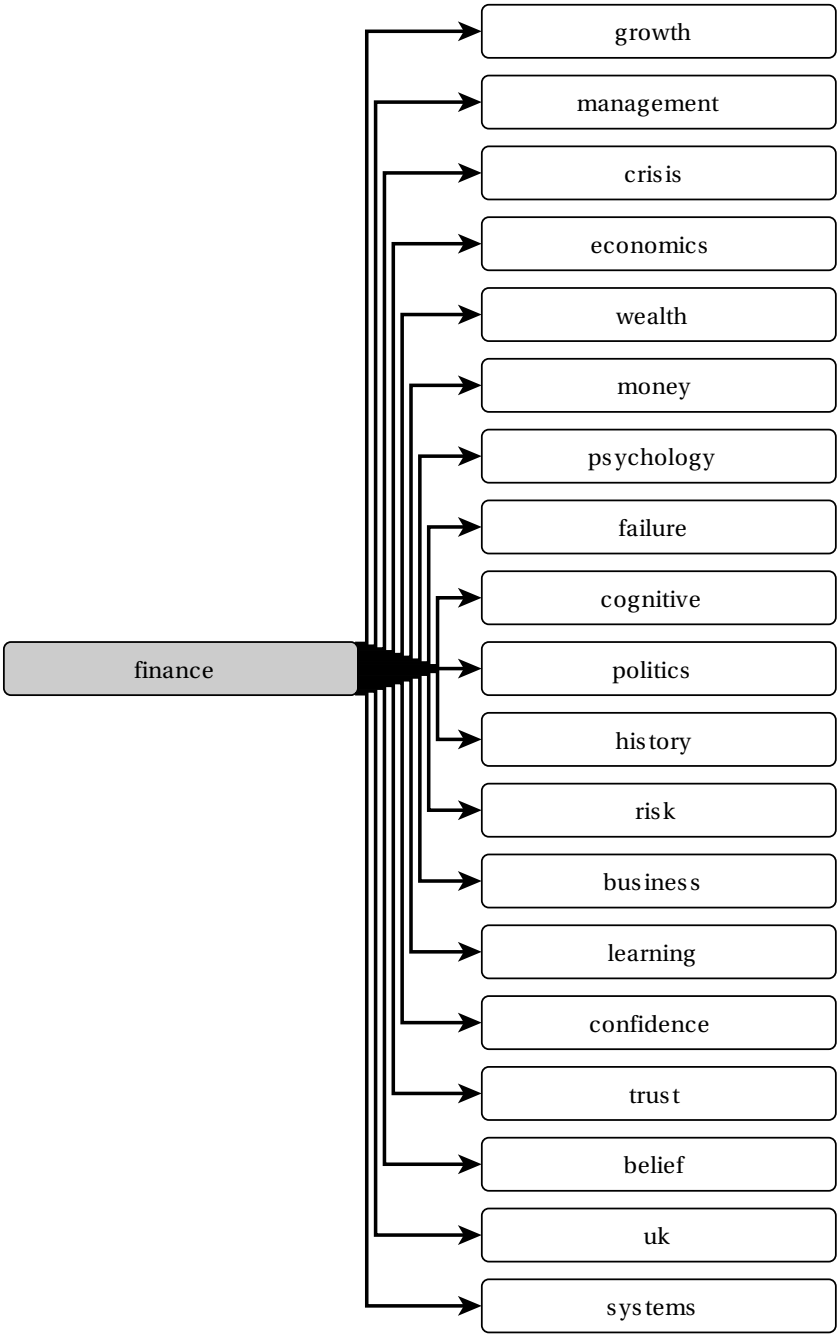


Figure 5.7: *Example Tag Bundle for finance (From the test run with* Confidence > 0.7 *and* MinSupport > 4*)*

bundle. Also, there are less bundles per user on average. (0.7, 4) and (0.9, 3) have more tags per bundle and the number of bundles per user is higher on average.

In Figure 5.7 is an example of a tag bundle. In this bundle all computed rules with the head "finance" are merged together. Contained in the tag bundle are tags such as "failure," "risk," "economics" or "crisis" indicating that the user for whom this bundle was created, seemed to be interested in resources related to the current financial crisis (the data was collected in early 2009).

Normalized Google Distance In order to determine if the derived tag bundles provide useful grouping of tags, the Normalized Google Distance (NGD) [CV07] has been used as measurement for the semantic relatedness of two terms. NGD takes advantage of the number of hits returned by Google to compute the semantic distance between concepts. The basic idea behind NGD is following: if one has two terms, first Google is queried for each term separately. Then the number of returned results is set in relation to the number of results returned by a query using both terms together.

Given two search terms x and y, the normalized Google distance between x and y, NGD(x, y), can be obtained as follows

$$\mathrm{NGD}(x,y) = \frac{\max\{\log f(x), \log f(y)\} - \log f(x,y)}{\log M - \min\{\log f(x), \log f(y)\}}$$

where $f(x)$ is the number of Google hits for the search term x, $f(y)$ is the number of Google hits for the search term y, $f(x,y)$ is the number of Google hits for the tuple of search terms x y and M is the number of web pages indexed by Google[5].

Parameter	μ	Median	Variance	σ
0.5, 8	0,390	0,528	0,050	0,230
0.7, 4	0,390	0,453	0,050	0,220
0.9, 3	0,380	0,242	0,050	0,220
random tags 1	0,640	0,636	0,040	0,210
random tags 2	0,660	0,588	0,040	0,200

Table 5.7: *NGD for tag bundles in comparison to the results for random tags.*

Table 5.7 shows the statistical characteristics of the NGD values between the tag bundle head and the bundled tags. The first three rows show the results for ARM with different parameter combinations. For each bundle head the normalized Google distance to each tag contained in the tag bundle is computed. The arithmetic average over each tag bundle for each user is for each parameter threshold setting about 0.4. For putting the determined NGDs into context, the pairwise NGD for two samples of five hundred randomly selected tags has been computed. The last two rows contain the pairwise NGD of these two runs.

[5]The Google search engine indexes contains approximately ten billion pages ($M \approx 10^{10}$). Google does not publish the exact numbers and they are subject to change anyway. 8,058,044,651 has been used as the one given in the original paper.

The expected value (μ) of the random tags is noticeably bigger than the expected values of the experiments. Although this is no proof – with an excessive empirical base – it still gives a strong indication that the (semantic) similarity between the head of the tag bundles is bigger than the (semantic) similarity between random tags. It was not possible in a reasonable time or with a reasonable number of requests per second to perform a test with a larger set of random tags.

5.4.3. Discovering Communities of Interest

The following section is not directly related to the sections before with semantic relations as common denominator. It aims at carving out the role of social tagging as *social* form of knowledge organization. Social Tagging is user centered. A real person is applying tags to a resource. Therefore tags reflect the vocabulary of individual person or a group of people.

This aspect is very important and deserves therefore an extra investigation. By following individual interests as reflected through the individual tagging behavior, relations between users describing a social network can be established.

In Kammergruber et al. [KVZ09] an approach is described for showing how tagging activities can be used to identify groups of people having similar interests. The information contained in social tagging data reflects the point of view and understanding of a community, presenting a valuable source of information for the discovery of community structure, content and intent. Based on the tag frequency vectors of users, a density-based clustering using a cosine distance function for determining the similarity between users is applied in order to find these communities of interest.

In the context of this approach, social tagging allows links to be established between users within a community sharing a common tagging context – meaning the usage of certain tags. This link is rooted in the interests being followed by the individual members of the community, and is expressed in the tagging performed on data deemed to be of interest to such a community member.

By following individual interests as reflected in tagging behavior, relations between users describing a social network can be established. Hendler et al. [HG08] points out the possible network effects – following Metcalfe's Law – in social tagging applications (as one of the early accesses of the Web 2.0 phenomena).

By establishing these links between individual members of a community, several features of interest can be explored:

- *Interest based user groups:* Based on groups of users, derived from tagging performed in line with social networks following similar interest and tagging behavior. The respective focus and intent can be automatically identified.

- *Link recommendation based on the existing structure within a group:* Users are statically linked via social networks (such as Facebook or Twitter) or platforms with social features (such as Flickr or Delicious). Through the comparison of automatically discovered tagging based networks and statically maintained counterparts, missing links can be identified and suggested to members of a community. This need not necessarily be links to other mem-

bers with similar interests, but also links to previously unknown resources and data identified to lie within the field of interest to a given user.

It has been an endeavor of knowledge management within Siemens to use the self-organizing and distributed parallel input of crowds to support knowledge structuring and dissemination [Ehm10]. The approach presented in this section will address a core challenge encountered in information bases too large for a single individual to grasp: the discovery of related and needed information. Through the automatic support of community organization a significant impact and increase of knowledge transfer is expected.

Communities of Interest

Focusing on tagging performed by communities of users allows the analysis of resources as perceived from a user's perspective. But since social tagging is performed in a distributed fashion, the semantic understanding of selected tags varies slightly from user to user. Unless an ontology is used to create a frame for a common reference point, tags will invariably be utilized in an imprecise manner.

For this reason, a clustering approach allowing the discovery of topics in annotated media allows the grouping of users by similar sets of tags. Different users might use different tags on occasion, but the greater number of words utilized will be very similar. The flexibility allowed by grouping such similar tag sets enables the interpretation of users with great overlap in the use of their tags to suggest a number of common interests, and vice versa. Frequent usage of certain tags reflects the interest of users for a topics related to these tags. Hence users with similar frequent tags are likely to be interested in common topics.

Discerning Features Instead of discerning user intent by grouping resources of their interest, tags provide a reliable alternative to gauge the attention of a user. In practice this often entails the addition of a single key word or key phrase to identify a given resource as belonging to a certain topic, issue or interest. The selection of these key words is usually not constrained in any way, but means providing suggestions to the user in order to keep the number of spelling variations low.

Defining Common Interest In order to discern similarity between interests of users, a vector space $V^{p\times n}$ is defined, representing tags that have been assigned by users. p is the number of users, n is the number of tags.

Since the intention is to cluster this vector space, a number of measures can be taken to normalize the free-text tags given out by users. Three commonly used steps have been employed:

- *Removal of punctuation* is a simple step ensuring that the infrequent use of special characters in tagging does not interfere with keyword matching.

- *Lower case* key phrases remove the different treatment of identical but differently capitalized words.

- *Porter Stemming* [Por80] reduces the number of keywords with significant impact on the semantic clarity of a tagging vector space.

The application of these normalization steps reduces the number of different tags: *(i)* removing special characters from the tags in T (see definition 5.1) and transform every tag to lower case (e.g. "Web_2.0" and "web2.0" are merged to "web20") and *(ii)* applying porter stemming (e.g. "blogs" and "blogging" are merged to "blog"). This normalization step introduces a fuzziness regarding semantics. For the task of applying a clustering algorithm, this is insignificant.

Tags have a higher amount of information than a comparable text. Hence, other preprocessing steps, for example weighting of terms (such as tf-idf) are not applied. The occurrence of individual tags can be directly interpreted as relevance without considering related and influencing terms as in text analysis.

The normalization of the tags in V leads to a derived vector space V', which is a $p \times m$ matrix (with $m < n$ representing the number of normalized tags). This vector space V' is the basis for the later described analysis algorithms.

Within this (high dimensional) vector space the proximity of individual users can be interpreted as a related proximity in their interests. This is captured by the cosine similarity between each user pair (u_i, u_j) with their corresponding tag vectors v_i and v_j. The angular discrepancy described by $cos(v_i, v_j) = \frac{v_i \cdot v_j}{\|v_i\| \|v_j\|}$ is used as a similarity measure between users. Clustering similar user vectors leads to a number of topics commonly tagged by all members of a group.

The cosine distance has a range from 0 to 1 since tags can only have a non-negative frequency. A value close to 0 means that the vectors are independent. 1 means that the vectors are exactly the same. Values in-between represent intermediate similarity or dissimilarity of two vectors accordingly.

An example: Supposed $m = 3$ and there are 3 tags "java," "web," "design". User u_1 has used "java" 3 times, "web" 5 times and design 8 times. User u_2 has used "java" 4 times, "web" 3 times and design 0 times. The vectors v_1 for user u_1 is then $(3,5,8)$. v_2 for user u_2 is $(4,3,0)$. The angle between those vectors can be computed as follows:

$$cos(v_1, v_2) = \frac{(3,5,8) \cdot (4,3,0)}{\|(3,5,8)\| \|(4,3,0)\|} = \frac{3*4+5*3+0*8}{\sqrt{3^2+5^2+8^2} * \sqrt{4^2+3^2+0^2}} \approx 0.55$$

After computing the similarity matrix for each pair of users, DBSCAN (*density-based spatial clustering of applications with noise*) [EKSX96] is applied as clustering algorithm. Other similarity based clustering algorithms, for example, hierarchical clustering (such as Single-linkage clustering) would also have been possible. Being resistant to noise and not requiring a number of clusters as input DBSCAN is a more reasonable choice.

The main concept behind DBSCAN is the concept of *density reachability*. DBSCAN has two input parameters ϵ and *minPoints*. If there are at least a certain number (*minPoints*) of points in an environment with radius ϵ (or short *eps*) of a point A, these points become part of the cluster where A belongs to. A is called a core point. For each point that is found in the ϵ-environment, all points within an ϵ-environment are added to the cluster and a further expansion is executed.

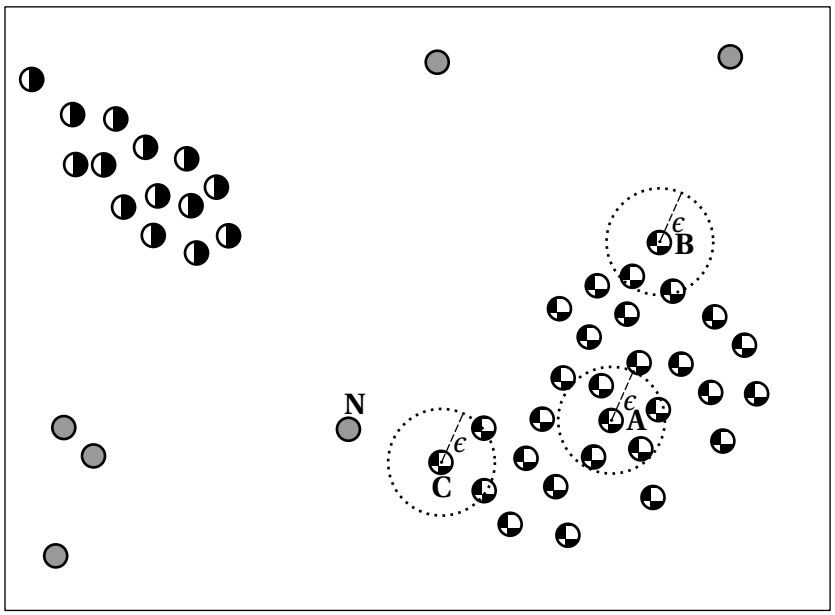


Figure 5.8: *DBSCAN clustering example with minPoints 3 and some ϵ. There are two clusters, one on the top left and another one at the right bottom. Gray points (such as N) are noise. A is a core point. B and C belong to the same cluster because they are density reachable through points between them. They are no core points because they lack the required minPoints in the ϵ environment.*

Figure 5.8 depicts an example for a DBSCAN clustering in a visual representation. The parameter *minPoints* is set to 3. An adequate ϵ has been selected. Shown are two clusters, one on the top left and another one at the right bottom. A is a core point, because there are at least 3 points (*minPoints*) in its environment with radius ϵ. B and C belong to the same cluster because they are density reachable through point between them. Since B and C only have 2 points in their ϵ they are not core points themselves. Gray points, such as N are noise. The figure shows that DBSCAN is noise resistant if ϵ and *minPoints* are selected accordingly. It is possible to assign points that do not belong to a cluster to be classified as noise. For heterogeneous social tagging data this is an important requirement.

Algorithm 2 contains a pseudo code representation of DBSCAN. There are two methods that are used but not listed: *regionQuery*(P,eps) returns all points contained in an ϵ-environment of P; *nextCluster()* returns an unique identifier for the next cluster. For a more in depth description of DBSCAN see [ES00].

Interpreting Tag Clusters Having discovered tag clusters, there are two specific conclusions to be drawn from them:

- *Link suggestions*:
By regarding the discovered tag clusters it is now possible to compare them to existing user groups and other statically maintained communities. The membership of a user in a tagging cluster suggests a community should exist to reflect their common interest, or if such

Algorithm 2 DBSCAN

```
function DBCAN(D, eps, MinPts)
    C ← 0
    for all unvisited point P in D do
        mark P as visited
        N ← regionQuery(P, eps)
        if sizeOf(N) < MinPts then
            mark P as NOISE
        else
            C ← nextCluster()
            expandCluster(P, N, C, eps, MinPts)
        end if
    end for
end function

function expandCluster(P, N, C, eps, MinPts)
    add P to cluster C
    for all point P' in N do
        if P' is not visited then
            mark P' as visited
            N' ← regionQuery(P', eps)
            if sizeOf(N') ≥ MinPts then
                N ← N ∪ N'
            end if
        end if
        if P' is not yet member of any cluster then
            add P' to cluster C
        end if
    end for
end function
```

tag name	User #1	User #2	User #3	User #4	User #5	User #6	User #7	User #8	User #9
systemmediaimag	10	30	350	30	60	170	10	40	50
systemfiletypejpg	10	30	190	30	30	170	0	0	20
systemmediadocu	10	0	0	20	0	0	0	50	10
systemfiletypepng	0	0	30	0	20	0	10	0	10
systemfiletypepdf	10	0	0	20	0	0	0	50	10
systemfiletypegif	0	0	120	0	10	0	0	40	20
microsoft	0	0	0	0	0	0	0	10	0
systemfiletypejpeg	0	0	10	0	0	0	0	0	0
volum	0	0	0	0	0	0	0	10	0

Table 5.8: *Cluster #5: Small cluster capturing system tags. The normalized tags can be seen on the left with the frequency of usage for a corresponding user indicated in each column.*

a community does already exist, a non-included user should at least be made aware of the existence of such a community.

- *Recommending interesting resources*:
Resources with tags for a cluster can be of interest to all members of a cluster. Should a relationship between a resource and a user not yet exist, such a link can be extrapolated and suggested automatically.

Case Study

The approach delineated in the previous section has been tested in the Siemens Wikisphere and the Siemens Blogosphere. But since the amount of available social tagging data does not reach the numbers achieved in Internet applications the approach has been evaluated against the delicious data set. Delicious was the first system to utilize tagging on a large scale and is hence a popular source of folksonomy data in research. The characteristics of the delicious data set are described in section 5.3.2.

Results After applying the normalization steps described in the previous section, the vector space V created by 2,270 users and 97,522 tags generate a reduced vector spaced V' yielding 80,134 distinct and normalized tags. This in itself already indicates the high dissimilarity and semantic validity of used tags.

Based on V', a user similarity matrix using the cosine similarity is computed in order to be able to perform topic based clustering of tag contents. The choice of the similarity measure depends strongly on the characteristics of the data it is applied on. In a study by Spertus et al. [SSB05] on data collected from Orkut,[6] the Euclidean distance-based similarity

[6] `http://www.orkut.com` a social networking site

ClusterId	number of user	topic
0	2099	NOISE
1	29	photography
2	111	(web-) design
3	12	video/ youtube
4	10	cooking
5	9	system media objects

Table 5.9: *Result of applying DBSCAN with ϵ= 0.3 and minPoints = 9*

has led to the best empirical results among seven – actually more sophisticated – similarity metrics. For the experiment with the delicious data set, the cosine similarity measure has been a straight forward choice with reasonable good results. For other data sets another similarity measure might be a better choice. DBSCAN clustering, utilizing values from the user similarity matrix to gauge the degree of common interest, discovers five clusters in the data set (see table 5.9). For minPoints ranging from values of five to nine and between 0.2 and 0.4 quite similar clusters are consistently found, indicating a stable and useful result. Cluster #1 focuses on photography, as can be seen by the heavy occurrence of commonly used tags, such as photography, photoshop or camera.

An excerpt of common tags in this cluster with the corresponding usage frequency is shown in table 5.10. Cluster #2 includes people using tags related to design, especially web-design. Cluster #3 contains users with many bookmarks tagging Youtube or other video based resources. In Cluster #4 individuals interested in cooking can be found. The common interest is expressed by the use of tags such as *recipe, chicken, dessert, beef, bread, soup, cake, food, fish, pasta, vegetarian, bacon, bean, shrimp,* and so on. Cluster #5 holds user bookmarking media files such as pictures (see table 5.8). The people in these clusters were not contained in each other's social network on delicious. This does not preclude them knowing each other, but one may assume with some degree of assuredness that most cluster members are not aware of each other.

tag name	User #1	User #2	User #3	User #4	User #5	User #6	User #7	User #8	User #9	User #10	User #11	User #12	User #13	User #14
photographi	4920	3540	3420	3030	3020	2460	2390	2050	1940	1720	1640	1430	1360	1340
photoshop	1100	350	390	70	320	120	160	240	470	100	120	410	40	40
blog	10	330	80	20	160	160	300	140	390	80	0	40	40	60
tutori	50	710	2650	0	450	0	220	540	920	210	90	0	330	120
video	120	0	270	10	80	0	100	0	40	320	0	0	140	0
flash	850	60	10	10	130	0	150	270	140	50	300	100	200	0
softwar	50	50	210	80	240	0	10	100	90	270	0	0	10	40
camera	10	150	640	20	430	160	140	20	260	20	0	200	200	10
light	1150	50	0	0	120	0	110	220	220	140	0	80	210	0
photo	30	100	940	0	90	0	30	220	370	0	0	0	140	60
busi	720	100	0	0	40	360	20	190	260	80	0	220	70	60
refer	20	50	480	10	320	0	70	40	340	100	0	0	0	130
art	0	340	0	20	160	150	60	0	10	80	0	0	90	110
tip	0	700	920	0	90	0	70	620	220	0	0	0	180	30
magazin	50	50	320	10	120	0	80	10	0	0	30	0	10	10
flickr	150	80	0	0	0	0	10	100	30	10	30	0	50	0
design	0	10	0	60	0	350	150	70	130	110	0	0	0	0
webdesign	0	40	0	0	0	0	80	40	260	0	0	0	20	90
wed	290	160	0	0	80	470	50	90	90	0	80	0	10	0
tool	0	0	290	0	120	0	20	200	290	0	0	40	30	10
book	130	100	110	190	40	0	50	110	10	0	30	0	0	80
inspir	0	60	80	0	160	0	80	1170	620	0	0	0	60	40

Table 5.10: *Excerpt Cluster #1, Topic: Photography and web design. The normalized tags can be seen on the left with the frequency of usage for a corresponding user indicated in each column.*

5.4.4. Urgent Request Channeling

Another scenario where a social tagging framework can offer services to external applications is in the area of targeted message distribution. The problem is related to traditional recommender systems, for example where movies are suggested to users based on ratings they made in the past [SKKR01]. While the specific application of recommender systems may vary, the basic idea to match items with users based on traces a user left is always the same.

Applied to the world of social tagging, the main traces a user leaves are the tag assignments. Based on these tags, an implicit profile of topics a user is interested emerges. With the help of these user tag profiles a prediction can be made, which users might be relevant for a given tagged item. Information overload (see chapter 2) is an important keyword here.

It is a challenging problem that gets bigger the easier it is to publish something and the more people make use of it. In this section an approach to reduce this problem for the use case of a question and answer platform is presented. Some of the findings were published in [WAH+12] and [LHMK12].

The algorithm has been designed to include special characteristics of TechnoWeb, but is with small adoptions applicable to other platforms. For example, TechnoWeb has user networks for certain topics. These networks are tagged. For the proposed algorithm these tags are considered. Other aspects of the algorithm work with plain folksonomies as well.

Introduction

Using crowdsourcing as a means to solve problems has risen significantly during the last years [Gas10, Sur05, LJL+07]. E-brokering companies like NineSigma, Innocentive, and the likes offer platforms where technological problems can be submitted and get distributed to experts [How09]. These experts can be distributed around the globe and are the ones who are most likely able to answer a specific question. The unique selling proposition of these companies is to know which experts have a higher probability to help solve the problem relative to others. These expert databases are primarily maintained manually with a high effort and a high level of quality.

The challenge analogous to that of the advertisement industry: Who are the recipients with the highest return on advertisement investments? Ideally, they would like to send a catalog or other costly mails only to those people who would buy their product afterwards. The better the potential customer can be profiled, the better the mails can be targeted [BKN09, Dav06, Spo11]. During the last decades data mining technologies have been improved to serve these needs [Lar04].

The advertisement industry has changed in times of social media [HML08, MB09, YLW+09, MSVV07, MEPG07, MM11, ZZ11]. Companies like Google, Facebook, and Amazon demonstrate that the ones who have more precise data about their users are the ones who can place advertisements in a more targeted manner. Being more focused increases the probability that a reader of an advertisement will buy a product [MB09, YLW+09]. Data about users is aggregated with almost every activity within a platform or even associated

platforms (see e.g. Google AdWords/AdSense[7] [MSVV07]). For example, this enables Amazon to recommend products to customers based on past activities of other similar customers. This activity data of users is called the digital trace.

Within the firewall of an enterprise setting with large, globally distributed divisions there is a similar challenge: finding the right expert who is able to help an individual in solving a problem without requiring a formal connection between two employees. Companies like MessageMind[8] offer tools which screen all connected repositories (such as SharePoint, emails and social media). The goal is to know who is working in what field and most probably has expertise to solve a certain problem.

In countries with a more strict data protection by law or in companies with strict privacy protection policies, the digital trace is of limited richness. The appropriate use of semantic technologies nevertheless allows an acceptable expert identification also in the setting of sparse digital traces [LKE11]. The less precise the expert identification works, the greater the challenge to handle the trade-off between sending the request to too many people (spamming) or skipping the right persons.

In this section a novel algorithm and a case study is presented. It is about the so-called urgent request functionality, a corporate problem solving engine of Siemens' TechnoWeb: "find people to get answers". Metrics for how to measure the quality of such expert identification algorithms are additionally developed.

Broadcasting vs. Target Messaging

The most common method of crowdsourcing is to message the whole crowd and hope that the crowd will have someone that has the knowledge and will respond (Figure 5.9).

Such a broadcasting approach with constantly messaging the entire crowd can lead to crowd fatigue. Maintaining a high level of awareness for messages to the crowd without crowd fatigue is one of the major challenges to the longevity of the crowd. Crowd fatigue is a problem which erodes the effectiveness and willingness of the members of the crowd to continually support the various posts made to them. In contrast to a broadcasting approach, this work introduces a target messaging approach in crowd sourcing with the limitation of sparse data for user profiling (Figure 5.9).

The Siemens Case

In 1999, a Siemens-internal social media platform called TechnoWeb was launched within a geographically distributed corporate software development center with 7,000 employees [HJ01]. The main reason for introducing TechnoWeb at that time was the so called Technology Breeding [Ack06] – gathering and sharing knowledge about new trends and technologies – as a component of the technology management process. As the software development projects became more and geographically distributed, the main value of TechnoWeb was to bridge the gap between experts working in different countries around the globe [LH05, MHH06]. During this time (2000) the urgent request feature in TechnoWeb

[7] `http://adwords.google.com`
[8] `http://www.messagemind.com/`

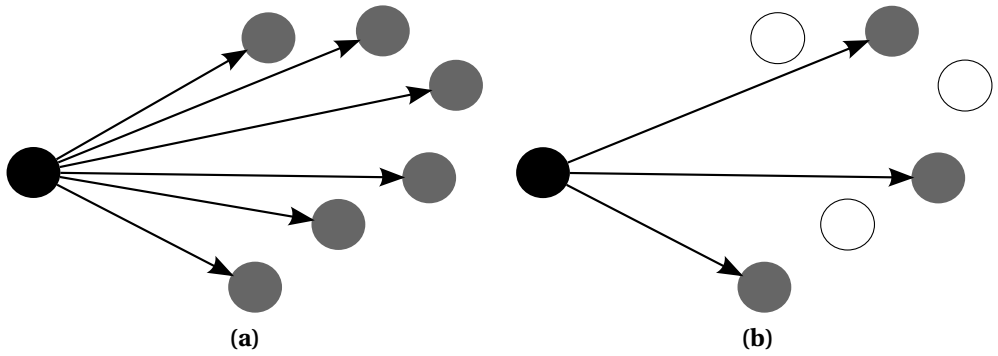


Figure 5.9: *Instead of sending the request to everybody (Broadcasting (a)) the request is sent to those identified experts with the highest probability to answer (Target Messaging (b)).*

has been introduced: everybody in the company was allowed to broadcast (supported by TechnoWeb) an urgent request for a technological problem per email without any censorship or filtering to all TechnoWeb users around the globe. This was a quite unusual approach in the year 2000.

The effect was amazing: the response rate to the colleagues was surprisingly high and extremely fast. An average of seven answers per urgent request has been counted and the first answer came on *average* within 50 minutes. This broadcasting based corporate problem solving was seen as a just-in-time reuse of expert knowledge and worked much better for tacit knowledge than all the knowledge databases for explicit knowledge at this time. Experts were more motivated to help some colleague or at least give them some advice if they recognized that the colleague was in need of help in this moment rather than feeding some database with explicit knowledge where it is uncertain that their contribution will ever be read.

The urgent request became the key use-case of TechnoWeb. Per interview of urgent request senders success stories have been found with significant business impact for Siemens and those success stories helped the TechnoWeb team to get a positive branding for TechnoWeb with “find people to get answers”. Since then, TechnoWeb has grown, been completely redesigned, and officially corporate-wide launched as TechnoWeb 2.0 in 2010. The main challenge for the redesign of TechnoWeb was to cope with the scaling effects of the urgent requests. At the beginning with 3,000 users, all working in the field of software engineering, it was possible to broadcast an urgent request. Even if people cannot support the sender of the request, they are interested what colleagues in the same unit are working on. As long as they receive not more than approximately one urgent request per day this is no issue. Strict rules have been defined for which cases it is appropriate to broadcast an urgent request (e.g. urgency, no TechnoWeb Group exists for the specific topic).

On days where 3 or even more urgent requests were broadcasted, first complaints from users were received. This is according to the TechnoWeb team experience the upper limit and is only accepted if the request is in the field of the professional experience, e.g. if a ma-

terial scientist receives an urgent request for a software problem. Otherwise he or she feels spammed. Therefore, in TechnoWeb 2.0 the sender of an urgent request had to select at least one of nine categories of professional fields (e.g. material science, energy, software...). Each member of TechnoWeb could deselect categories in his or her personal notification settings in order to receive only relevant urgent requests.

The observation was that 65 % of the users did not change their settings at all and 21 % of the senders of urgent request selected more than one category. When TechnoWeb reached 15,000 users, an urgent request was distributed to 12,500 people in average. This was more broadcasting than target messaging. In October 2011 the decision on introducing a more sophisticated targeting algorithm based on the tags of an urgent request has been made. The ideas behind the introduced algorithm and its design are the content of the following sections.

TechnoWeb's Urgent Request Channeling

The approximately 23,000 users of TechnoWeb (see the beginning of this chapter) can follow tags, can join a TechnoWeb Group in a field of interest (a network also has tags assigned) and can assign tags whenever they post a news story. These tags are the basic data of their digital trace[9]. All these tags are managed together with tags from other social media applications in an enterprise wide Tagging Framework.

When typing the text of an urgent request on TechnoWeb, the tagging framework makes some automatic tag suggestions (see section 5.5) which can easily be selected by the sender. Additionally, the sender can manually add tags which fit best to the urgent request.

Before posting an urgent request on TechnoWeb, it is required for the sender to select the estimated Business Impact of the urgent request (Figure 5.10) with a slider.

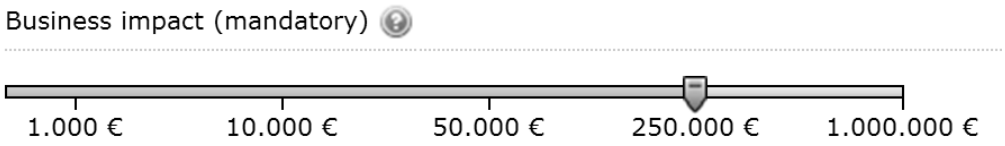


Figure 5.10: *Slider to select the Business Impact of an answer to his request.*

The Business Impact that has set ranges from € 1,000 to € 1,000,000. The higher the Business Impact the less rigorous the selection of recipients is. If the urgent request has a Business Impact of just € 1,000, it is not appropriate to contact thousands of experts. In this case the urgent request is only sent to those experts which are directly associated to one of the tags, of the urgent request. The risk is taken that an expert is left out of the recipient list which might have replied to the urgent request. As the data analyses shows later on, some relevant experts are missed. As long as the sender nevertheless gets enough support, this

[9] For data protection reasons these features are implemented in a way that the user has the full control over the visibility of his data and no performance control is possible. E.g.: if the user follows a tag, nobody except himself or herself can see which tags he follows. However it is not surprising for the user that he or she will then receive urgent requests related to this tag.

is acceptable. If the urgent request that gets posted has a Business Impact of € 1,000,000 or more, the risk of losing a potential answer from an expert is not acceptable. In this case, the urgent request is broadcasted to the whole TechnoWeb community.

For the Business Impacts in between € 1,000 and € 1,000,000, the group of the recipient list is further and further extended, for example, by adding experts which are associated to co-tags (tags which are often used together with the given tags) [LKE11, KVEL10] or similar tags (e.g. spelling variants), experts which are member in so-called partner networks of a network associated to the given tags, and experts which are often answering to urgent requests.

User Selection Algorithm

As mentioned, an urgent request has tags and a business impact. By taking these two factors into account, recipients of notifications for an urgent request are determined. A high business impact justifies a higher number of notifications and involvement of more employees. A low business impact must not distract the attention of too many knowledge workers. Hence, for important urgent requests a fuzzier selection of users is acceptable. Five Business Impact levels $b = 1$ (€ 1,000), $b = 2$ (€ 10,000), $b = 3$ (€ 50,000), $b = 4$ (€ 250,000), and $b = 5$ (larger than €1,000,000) are defined as input for the algorithm to match the importance of an urgent request. Figure 5.11 provides an overview behind the concept where the algorithm is integrated into the urgent request process.

Several indicators can be used to identify candidates for an urgent request:

1. *Tag assignment*: A user has used a tag in a tag assignment, such as a blog post or wiki page. This is most important data source. The collection of all tag assignments of all users in all systems is referred to as folksonomy.

2. *Follow tag*: A user has defined personal follow tags. These tags are used to indicate an interest in that topic. There is a personalized view provided with TechnoWeb activities for these tags.

3. *Commented*: A user has made a comment on a tagged urgent request or news entry.

4. *Member in network*: A user is a member of a network that has a tag. A network is some kind of group that has been found to discuss certain topics, such as Java development. This is analogues to a Xing, LinkedIn, or Facebook group.

5. *Member in partner network*: A user is a member of network that is a partner network of a network that has a tag. For example the "GWT"[10] network is a partner network of the "Java" network.

6. *Top commenters*: TechnoWeb users that have commented on an Urgent request more than a certain threshold.

[10] Google Web Toolkit – see `http://www.gwtproject.org/`

These indicators describe a less specific match from top to bottom. If a user has used a tag assignment frequently it is very likely that the user is an expert in a certain topic or a least interested in it. This means a person can answer a question him- or herself directly or if this is not the case, provide another form of assistance, such as recommend colleagues. The algorithm uses heuristics that have been inferred from users that actually answered an urgent request in the past. Depending on the business impact the tags of an urgent request are mapped to tags from the mentioned sources. The lower the business impact the exacter the tags must match. Partner networks are not used as sources for business impact 2. Top commenters are only included for business impact 4. For business impact 4 this expansion makes sense since active members have shown the willingness to help before. If there are many false positives then this is no problem. A problem that is marked with a value of € 250,000 can justify many notifications that go to irrelevant persons.

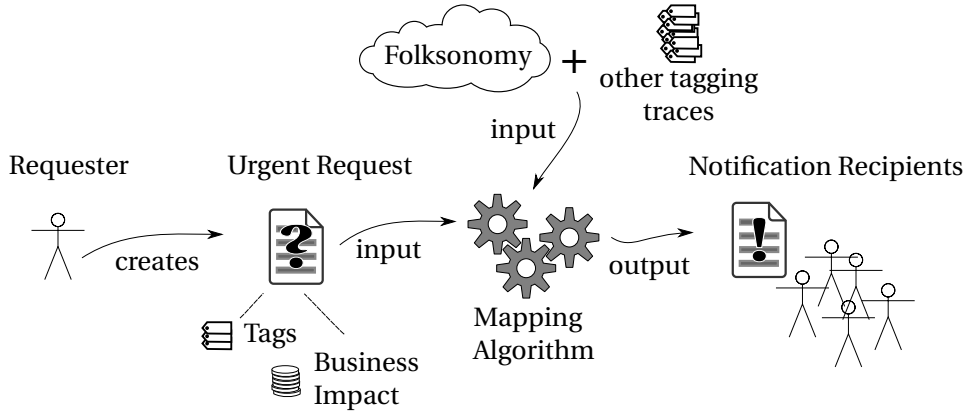


Figure 5.11: *The idea behind the urgent request targeted message distribution is that requests are not simply sent as broadcast, but are distributed to selected users based on user profiles and the business impact and tags of an urgent request. Additionally to the folksonomy for the case of TechnoWeb there are some other tag sources for users available. For example, a user can be member in a network that has tags itself. The latent connection between the members of a network and the tags of a network is considered adequately in the algorithm.*

The user selection algorithm takes following steps:

1. Expand tags: Match variants of the provided tags based on a Levenshtein similarity. With higher business impact the used similarity threshold is lower. If business impact ≥ 3: include co-tags of provided tags. Only frequent co-tags are considered having a relative co-occurrence value (see definition 5.4) higher than a certain threshold. This threshold is lower for business impact 4 than for business impact 3. For business impact 1 and 2 no co-tags are included.

2. Find users that have used at least one of the expanded tags in a tag assignment.

3. Find users having at least one of expanded tags as follow tags.

4. Find users that have commented on urgent requests that have been tagged with at least one of the expanded tags.

5. Find users that are members in a network that has been tagged with at least one of the expanded tags.

6. If business impact ≥ 2: find users that are members in a partner network of a network that has been tagged with at least one of the expanded tags .

7. If business impact ≥ 4: include user that have commented more than a certain number of times.

After these steps a set of matching users is determined. For applications where it is necessary to have a ranked list of users, it is possible to create an overall score for each user. This score depends on how many of the above conditions a found user meets. Additionally, how well a user fits to a criterion can be considered. For example: A user has all tags of an urgent request as a follow tag. This leads to a higher score than a user that has only one tag as follow tag. Other ways for creating a score, such as giving higher score values for exact tag matches than to fuzzy matches, are possible. In the implementation of the tagging framework that is used in production (see chapter 6) such score are integrated in various other use cases that are beyond the scope of this work. For the message targeting algorithm a ranked list of matches is not needed, since a user can either be a match (and receive a notification) or not. This is independent from a score value reflecting the fitness of a match. Hence, scoring aspects are not elaborated in this work.

Metrics for benchmarking the target messaging algorithm

Let $E_{all,i}$, be the number of emails sent out for the i-th urgent request according to the broadcasting algorithm and let $E_{tgt,i}$, be the number of emails sent out for the i-th urgent request according to the target messaging algorithm. Then for each of the n urgent requests the spam reduction factor is defined by

$$r_i = \frac{E_{all,i}}{E_{tgt,i}} \qquad \forall\, i \in [1;n] \tag{5.1}$$

Example: By broadcasting a request to only to 150 users instead 1,500 users the spam is reduced by a factor of $r_i = 10$. However, reducing spam without finding the right persons is pointless. The 'right' persons are users who can and want to support the sender of the urgent request. In the ideal case it is desirable to achieve a spam reduction factor r_i as large as possible but nevertheless get 100 % of the replies which result from broadcasting. Therefore, an important metric can be defined as the relation of the spam reduction to the expert hit reduction. With the number of comments $C_{all,i}$ according to the broadcasting algorithm and the number of comments $C_{tgt,i}$ according to the target messaging algorithm a gain factor is defined by

$$G_i = r_i \frac{C_{tgt,i}}{C_{all,i}} \qquad \forall\, i \in [1;n] \tag{5.2}$$

for each of the n urgent requests sent.

Example: if a spam reduction factor of $r_i = 10$ is achieved but at the same time just $C_{tgt,i} = 2$ answers instead of $C_{all,i} = 20$ were provided, then the result is a gain factor $G_i = 1$. In other words: a random selection of the recipients of the message is predicted to get about the same result as the targeting algorithm. If the gain factor $0 \leq G_i < 1$ the target messaging algorithm is even worse than a random selection of recipients. If $G_i > 1$ then the target messaging algorithm is better than a random selection of recipients. The gain factor G_i is the added value of the target algorithm towards a trivial, random selection.

A further metric is the conversion rate

$$c_i = \frac{C_{tgt,i}}{E_{tgt,i}} 1000\,‰ \qquad \forall i \in [1;n] \tag{5.3}$$

which measures the ratio of how many of all the recipients of a targeted message actually respond.

With $c_{all,i} = \frac{C_{all,i}}{E_{all,i}}$ and equations 5.1-5.3 the gain factor $G_i = \frac{c_i}{c_{all,i}}$ also can be understood as the factor how much better the conversion rate of the target messaging algorithm is compared to the conversion rate when broadcasting. In the following all three metrics are used to asses the targeting algorithm: spam reduction factor r_i, gain factor G_i and conversion ratio c_i.

Test Scenario for comparing the semantic target messaging algorithm with the broadcasting algorithm

A quantitative approach is used for comparing the two algorithms. The evaluation of other data mining and statistical algorithms introduced in this chapter has been conducted with the data sets described earlier in this chapter. For testing the message targeting algorithm the described data sets lack additional information, such as comments on urgent requests. Tagged urgent requests are included in the collected social tagging data sets for Siemens, though. Without additional information an evaluation based on the Siemens folksonomy is unfortunately not possible.

Therefore another data set has been collected. It contains the urgent requests sent between beginning of June and end of September 2011 within Siemens. These where the last 4 months of the old algorithm which, as mentioned before, was an almost-broadcasting algorithm as these urgent requests were sent to 12 500 users on average. During that time the number of users grew from 15,000 to 18,000. The advantage of this test data is that with the TechnoWeb database it can be traced not just how many experts answered but exactly who answered.

For simulating the new algorithm, a simulation tool on a test server (a server running a mirror of the TechnoWeb instance) has been installed. The test server has been updated for each month with the tagging data and user-data status of the last day of this month. Due to the growing number of users a small error has been introduced. This error is maximal 5 % but in average less than 2 %. For the evaluation scenario this does not lead to a significant distortion of the results.

In this period $n = 138$ urgent requests were published, which got at least one answer (an additional 10 urgent requests were too specific and did not get any answer. Those 10 were not considered here as in this case there will be no measurable difference between the old and the new algorithm). The number of urgent requests is in the range of one per calendar day, but more than one per work day. As the old algorithm did not need the Business Impact parameter, there is no data for the business impact of those 138 urgent requests. However, as this is an input parameter for the target messaging algorithm, the number of comments has been simulated for all five Business Impact levels $b = 1,\ldots, b = 5$. The advantage of this approach is that this leads to 4 times more data and hence potentially more insights.

The live-database of TechnoWeb provided the data of who answered to urgent requests, the simulation tool has shown for Business Impact $b = 1$ which of these answering experts would have been on the recipient list according to the new algorithm, the same for $b = 2$ and 3 and 4. Business Impact $b = 5$ does not make sense to be simulated as the target algorithm is per definition broadcasting in this case which would be the same as the old algorithm.

Therefore, the definition in equation 5.1 can be reformulated. The well estimated spam reduction factor is determined by

$$\hat{r}_i(b) = \frac{\hat{E}_{old,i}}{E_{new,i}(b)} \qquad \forall i \in [1,n], \forall b \in [1;5] \tag{5.4}$$

to be dependent on the business impact level b. Additionally, the reformulated equation 5.2 (gain factor) becomes

$$\hat{G}_i(b) = \hat{r}_i(b)\frac{C_{new,i}(b)}{C_{old,i}} \qquad \forall i \in [1,n], \forall b \in [1;5] \tag{5.5}$$

and equation 5.3 (conversion rate) is adapted to

$$c_i = \frac{C_{new,i}(b)}{E_{new,i}(b)} 1000‰ \qquad \forall i \in [1;n], \forall b \in [1;5] \tag{5.6}$$

with the measures of

- $\hat{E}_{old,i}$ = Emails sent out according to the old algorithm
- $E_{new,i}(b)$ = Emails sent out according to the new algorithm with a simulated business impact level b
- $C_{old,i}$= Comments according to the old algorithm
- $C_{new,i}(b)$ = Comments according to the new algorithm with a simulated business impact level b

Analyses of the spam reduction factors

The target messaging algorithm is designed in a way that the higher the business impact is, the higher the number of recipients. Hence, the risk of not sending emails to experts that might respond to an urgent request is reduced. Therefore, the total spam reduction factor

$$\bar{\hat{r}}(b) = \frac{\sum_{i=1}^{n} \hat{E}_{old,i}}{\sum_{i=1}^{n} \hat{E}_{new,i}(b)} \qquad \forall b \in [1;5] \tag{5.7}$$

decreases with increasing business impact level b (figure 5.12, see also the full data set at figure 5.16). At $b = 5$ the algorithm is broadcasting, resulting per definition to a spam reduction factor of $\bar{r}(5) = 1$.

Note that equation 5.7 is similar but not equal to the average of all the single spam reduction factors $\hat{r}_i(b)$ (compare with equation 5.4). Equation 5.7 is the more precise and more appropriate metric than the average due to the high variance of the data.

In absolute numbers, this means that within the first three months after the launch of the new algorithm, more than 1,000,000 notification emails were saved.

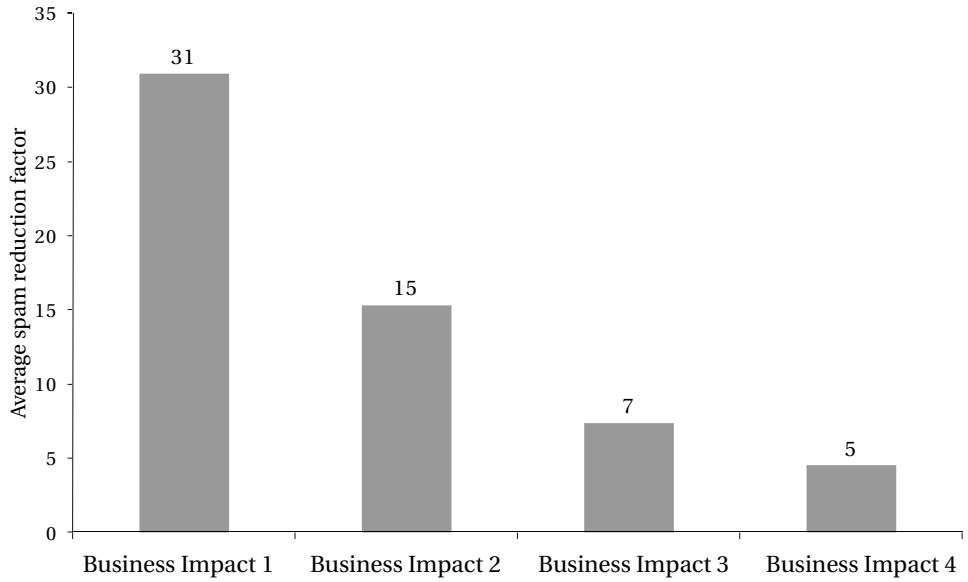


Figure 5.12: *Total spam reduction factor $\bar{\hat{r}}(b)$: the higher the business impact the less rigorous is the targeting.*

Analyses of the gain factors

Sending out a message to less people is per se not a challenge. The question is, if the response rate is under-proportionally decreasing – as measured with the gain factor $G_i(b)$. The ideal case with no decrease of the response rate is not realistic in the Siemens setting as there is always some knowledge in the brains of the experts which has no representation in their digital trace. Therefore, even the best algorithm is not able to do a perfect topic/expert matching.

It is reasonable to start with the (arithmetic) average gain factor

$$\bar{\hat{G}}(b)_{arithmetic} = \frac{1}{n}\sum_{i=1}^{n}\hat{G}_i(b) \qquad \forall b \in [1;5] \tag{5.8}$$

which decreases for increasing business impact (figure 5.13).

For business impact b=5 the gain factor is again per definition $\bar{G}(5) = 1$. Note that the standard deviations are much larger than the mean values itself. The arithmetic averages and standard deviations are therefore no appropriate representation of the data.

Based on the data distribution (figure 5.14) and ratio based computation of the gain factors according to equation 5.5, the logarithmic scale of the histograms (figure 5.14) and the geometric mean value

$$\bar{\hat{G}}(b)_{geometric} = \sqrt[n]{\prod_{i=1}^{n}\hat{G}_i(b)} \qquad \forall b \in [1;5] \tag{5.9}$$

is a more appropriate representation of the data[11].

The average gain factors are all larger than one – which means that the message targeting algorithm does some real targeting and is not just a random selector. A p-value computation (table 5.11) based on a normal distribution in logarithmic scale (figure 5.14) proves the statistical significance of $\hat{G}_i(1) \geq 1$. For higher business impact levels b the targeting is less focused and therefore the significance of being better than a random selector is lower than for b=1 (table 5.11).

Hypothesis	p-value in %	% in sample
H1: Algorithm for b=1 is better than random: $G_i(1) \geq 1$	2.46[12]	1.52
H2: Algorithm for b=2 is better than random: $G_i(2) \geq 1$	6.62	6.85
H3: Algorithm for b=3 is better than random: $G_i(3) \geq 1$	12.97	12.35
H4: Algorithm for b=4 is better than random: $G_i(4) \geq 1$	10.54	11.40

Table 5.11: *Statistical significance for hypotheses H1 to H4*

It is not surprising that the measured gain factor values nevertheless are in average not very high. The main reason for that is that there are a lot of new TechnoWeb users. These users have no or sparse digital traces.

The large variance of the gain factors (figure 5.14) can be understood by a more detailed investigation of the reasons:

• Each urgent request has a different topic. Some topics fit better to be solved by crowd sourcing than others.

[11] According to equation 5.5, the gain factor $\hat{G}_i(b) = 0$ is zero if the number of comments $C_{new,i}$= 0 is zero. If this happens, the measurement of the gain factor is not possible as there is nothing like a 0.9 comment but either there is at least one comment or there is no comment. Hence, those urgent requests with zero comments at a certain business impact level b were deleted for the logarithmic computation and for equation 5.9. Therefore n is reduced to n=114 for b=4; n=81 for b=3; n=73 for b=2; n=66 for b=1.

[12] $p \leq 0.05$

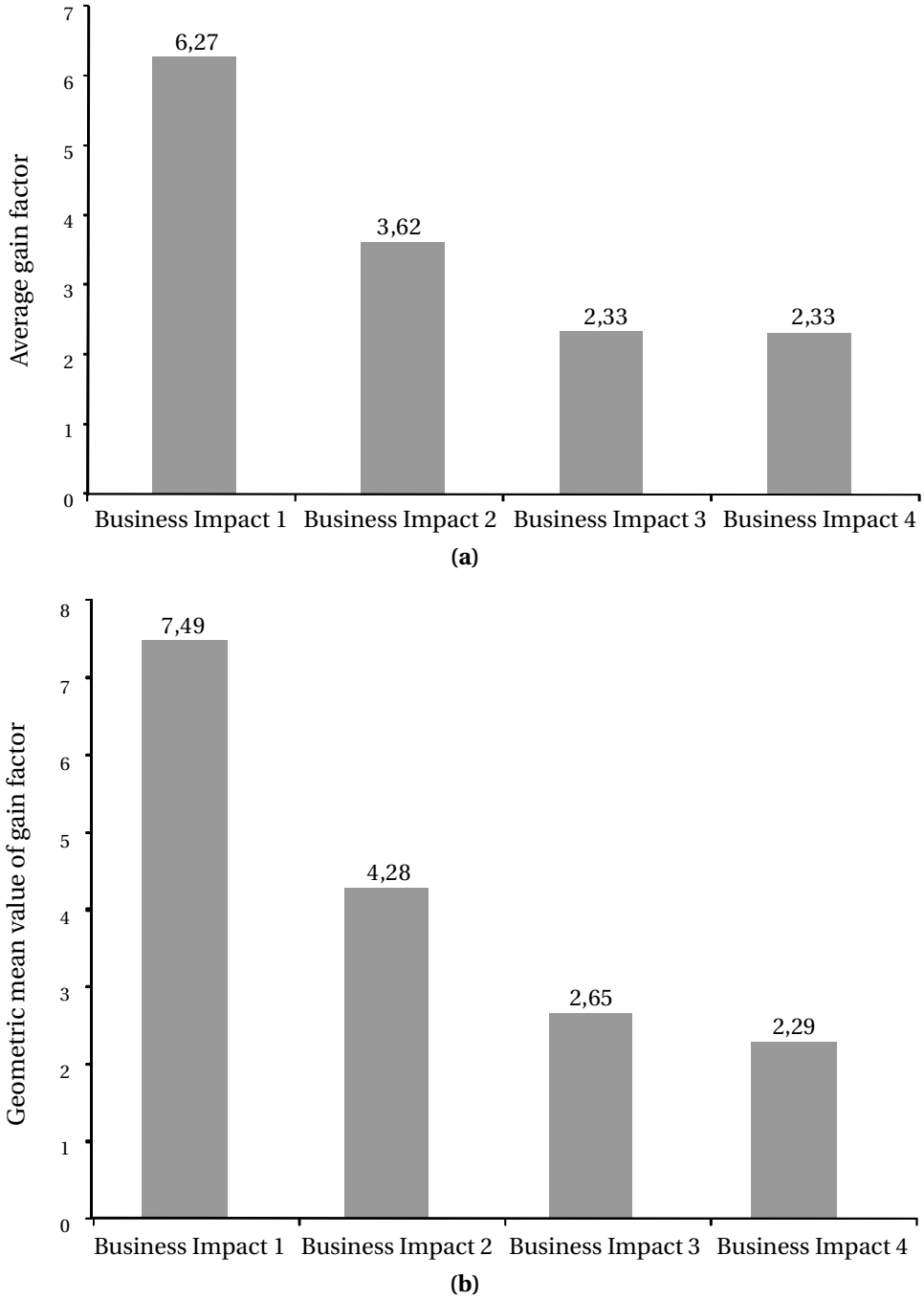


Figure 5.13: *a) the arithmetic mean values $\bar{\tilde{G}}(b)_{arithmetic}$ are decreasing with increasing business impact b. The corresponding standard deviations: 14.04 for b=1; 6.65 for b=2; 3.81 for b=3; 1.99 for b=4. b) the geometric mean values $\bar{\tilde{G}}(b)_{geometric}$ are decreasing with increasing business impact b. The corresponding standard deviations factors: 2.49 for b=1; 2.63 for b=2; 2.37 for b=3; 1.94 for b=4.*

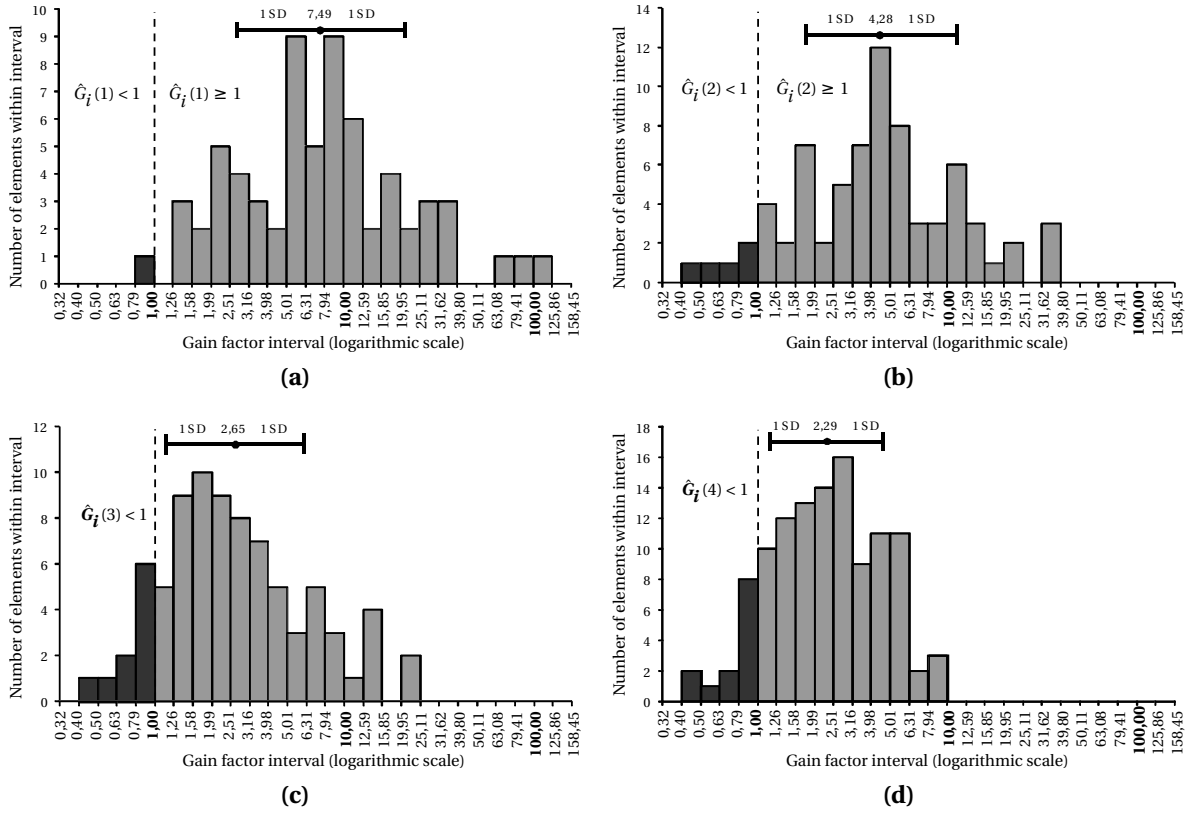


Figure 5.14: *Full data set of the gain factors $\hat{G}_i(b)$ in a logarithmic scale for the different business impact levels a) b=1, b) b=2, c) b=3, d) b=4 and the corresponding geometric mean values and standard deviation factors. The higher the business impact value, the less targeted the urgent requests are distributed, the higher the risk that in some specific cases the distribution algorithm might be worse than random selection ($\hat{G}_i(b) < 1$). "1 SD" stands for "one standard deviation".*

• The formulation of an urgent request is different each time. Some senders of urgent requests have better skills to pose a problem in an interesting and understandable way than others.

• Depending on the description of the posed problem, the tagging framework suggests some tags and the sender of the urgent request selects some of the tags or adds some new tags. Some topics have more significant and unique tags than others.

• Then, the target messaging algorithm identifies all experts who are directly or indirectly associated to those tags. The newer the topic is or the newer the TechnoWeb users are in this field of expertise the less best-fit recipients can be identified.

• As soon as the urgent request is sent out, the response rate is dependent on the time and current project situation of the potentially responding expert. During the holiday season, the potentially responding expert might be on vacation and during the final phase of a customer project he might have other priorities.

The high variance of gain factors is therefore not surprising.

At the end of the day, neither the reduction of mails sent nor the number of responses matter if an urgent request remains unanswered. What matters is whether the sender of the urgent request got useful advice or help.

Example: Best Gain According to figure 5.14a, the highest gain factor determined is $G_i(1) = 104$. This urgent request has the name "Anodizing" and was tagged precisely with the tags "anodize" and "surface treatment". The old algorithm broadcasted this urgent request per email to $\hat{E}_{old,i} = 12,902$ people.The sender got $\hat{C}_{old,i} = 2$ comments. The new algorithm identified just $\hat{E}_{new,i}(1) = 62$ experts associated to those tags when applying the selection criteria for business impact level $b = 1$ (€ 1,000). One of those 62 recipients was one of the two experts who responded. Therefore $\hat{C}_{new,i}(1) = 1$. According to equation 5.4 the spam reduction factor

$$\hat{r}_i(1) = \frac{\hat{E}_{old,i}}{E_{new,i}(1)} = \frac{12,902}{62} = 208$$

and according to equation 5.5 the gain factor is

$$\hat{G}_i(1) = \hat{r}_i(1)\frac{C_{new,i}(1)}{C_{old,i}} = 208\frac{1}{2} = 104$$

This example also shows the quantization problem if the number of comments is a small number. Assumed that the second responder of the urgent request in the above example had a better digital trace in this field, then would $C_{new,i}(1) = 2$ and the gain factor $\hat{G}_i(1) = 208$, which is an extremely high gain factor. On the other hand if the first responder had also no digital trace in this field, then $C_{new,i}(1) = 0$ and the gain factor would have been $\hat{G}_i(1) = 0$, which is the worst possible gain factor.

For the same quantization reason a lot of urgent requests have a gain factor of zero at the business impact level b 1 and 2 and some even for 3 or 4.

Similar to the averages in figure 5.13a and 5.13b almost each of the single urgent requests has a decreasing gain factor with increasing business impact (disregarding the quantization problems). Not surprising: less rigorous targeting for higher business impacts means a lower gain factor.

Another view on the data is the histogram of the gain factors in figure 5.14. This view on the data shows that there is a number of urgent requests with a gain factor between zero and one.

Example: Gain Factor less than 1 An urgent request in the field of oil and gas was broadcasted according to the old algorithm and successfully answered. The urgent request was tagged with "oil&gas". This tag is the name of a business unit and therefore the sender thought that this is an appropriate tag. At this time the (simulated) target messaging algorithm did not separate the tag "oil&gas" into a tag "oil" and a tag "gas". Nobody was identified who is associated to the tag "oil&gas" as this tag was never used before.

More critical is the case that an urgent request is in the field of a new technology. Surprisingly, in large companies those urgent requests nevertheless find answers. However, if the technology is "cutting edge" and nobody has a digital trace in this field yet, then even the best target messaging algorithm will not find the right person. This is known as the *user cold start problem* in the context of recommender systems. Recommendations strongly depends on the information available for a user in order to compute satisfying suggestions.

One simple solution for such cases is to broadcast the urgent requests to all TechnoWeb users, but per definition this will not be done for a business impact of € 1,000. Only for the highest business impact level b = 5 – in other words a business impact larger than € 1,000,000. For improving other types of recommendations regarding new users, such as suggesting interesting discussions, some kind of "bootstrapping" mechanism is needed. This can be achieved with the support of some kind of an initial interview [GKL10] or other mechanisms, such as suggesting popular or random items to new users [RAC+02].

Analyses of the conversion rates If an urgent request is sent out, it will not be read by all recipients and only some of those readers are able to support and of those only some will actually do it. The conversion rate c_i, according to equation 5.6, measures how many of 1,000 recipients that receive an urgent request actually responded to it. Pricing of Facebook advertisements for page impressions when compared to advertisement clicks at the web page of Facebook [13] show that advertisement clicks are approximately 500 times more expensive than page impressions. As the web advertisement market is an important market, it can be assumed that advertisement placement algorithms of Facebook or Google can be seen as a benchmark [MSVV07], [MM11, ZZ11].

Transferred to the urgent request scenario, this would mean that one can expect 2 of 1,000 of those who receive the email (get the title of the urgent request into their mailbox) will click (open) the mail. It can be assumed that not more than one out of 20 or one out of 50 is able to help and really responds to the urgent request. This would lead to a total conversion rate c_i between 0.04 and 0.1 of 1,000 recipients.

[13] `http://www.facebook.com/advertising/` as of 2012/01/23

Luckily, the situation in an enterprise context (Figure 5.15 and 5.16) is better than on the open market. Most employees are motivated to support their colleagues. The typical responses are hints or contact persons. It cannot be expected that somebody spends too much time on responding to urgent requests. They are also curious about the technological challenges of other colleagues around the globe. They are even more curious if the business impact is on level $b = 4$ (around € 250,000) or $b = 5$ (larger than € 1,000,000).

The average[14] conversion rate

$$\bar{c}(b) = \frac{\sum_{i=1}^{n} C_{new,i}(b)}{\sum_{i=1}^{n} E_{new,i}(b)} \qquad \forall b \in [1;5] \tag{5.10}$$

is again decreasing with increasing business impact (figure 5.15, see also the full data set of figure 5.16) as the conversion rate is strongly dependent on how accurate the message is targeted and as mentioned for higher business impact a less rigorous targeting can be accepted.

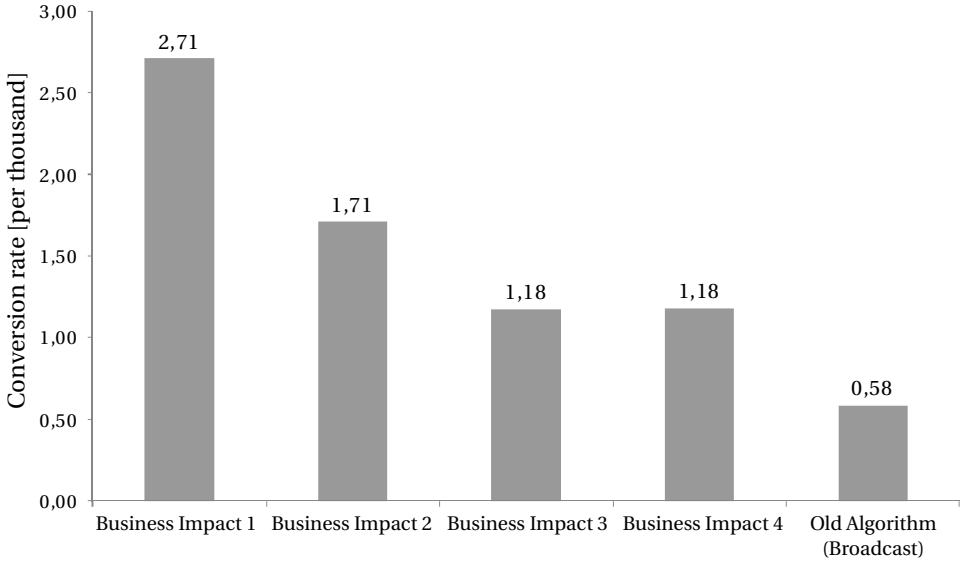


Figure 5.15: *Average conversion rates $\bar{c}(b)$: the number of responses to an urgent request per 1,000 emails sent.*

The conversion rate $\bar{c}(3) \approx \bar{c}(4)$ is on average almost equal for the business impact levels 3 and 4. One of the differences between these two business impact levels is that urgent requests with business impact level 4 are also sent to employees which already responded to any other urgent request before, independently in which field of expertise. For these employees the probability to respond is higher than for employees, who never responded to an urgent request before.

[14] This unusual way of computing a total average is preferred because it compensates for quantization problems.

Note that in the present case study, the recipients were not informed about the business impact of the urgent request due to the setup of the test scenario. It can be guessed that employees will be even more motivated to respond to an urgent request if they are informed about the high business impact.

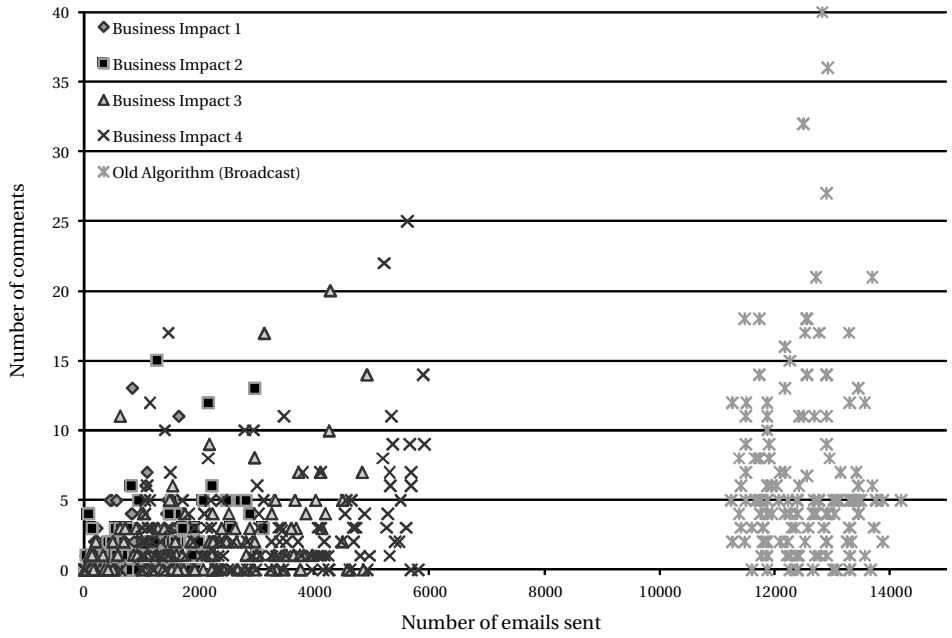


Figure 5.16: *Full data set of the number $\hat{E}_{old,i}$ of emails sent according to the old algorithm (right side) and number of emails $\hat{E}_{new,i}$ sent according to the new algorithm (left side) vs. the number $\hat{C}_{old,i}$ respectively $\hat{C}_{new,i}$ of answers received on these urgent requests.*

Notifications for October 2011 to March 2012 The smart distribution of urgent requests in TechnoWeb started regular operations in October 2011. The new intelligent urgent request channeling reduced the amount of email notifications by 324,000 (79 %) in only one month, addressing experts for requests with a cumulated business impact of more than € 5,000,000.

Table 5.12 gives an overview about the number of urgent requests for each BI level with the corresponding number of notifications sent out and the average number of answers. The data cover a 6-month period from October 2011 to March 2012.

Conclusion The case study demonstrated that even in sparse digital trace data, the target messaging algorithm for urgent request distribution can both lead to a significant reduction of the number of emails sent while delivering an acceptable response rate (figure 5.15 and 5.16).

It has been shown that the target messaging algorithm is significantly (Table 5.11, figure 5.14) better (on average by a factor 7.49) than a random selector for the lowest business

	# Urgent Request	∅ Notifications sent / BI	∅ Answers / BI
BI=1	97	797	3.8
BI=2	16	2,708	5.5
BI=3	22	4,799	5.5
BI=4	33	5,851	7.8
BI=5	14	19,811	20.8

Table 5.12: *Number of Urgent Requests at each BI and corresponding average notifications and answers.*

impact (1,000€). The lower the business impact the more important is the targeting. The higher the business impact the higher is the business risk to lose an important answer and consequently the less targeted is the algorithm designed to be.

The alternative to broadcast every urgent request to all TechnoWeb users is not an option as experts would see such unfocussed mails as spam and would turn off the urgent request notifications which would lead to a worse situation than the current situation: the target messaging algorithm is only as good as the digital trace data of the experts.

The results show that it is important to focus on the usability of the urgent request's user interface and all TechnoWeb features and activities which improve the richness of the expert's digital trace. Spending resources in further improving the target messaging algorithm is not needed. The better usage of tags will lead to higher gain factors. This will occur due to the user's experience and can be aided by guiding features on TechnoWeb as well as increased awareness of the benefits of tagging.

Corporate problem solving methods based on target messaging algorithms are already mature enough for creating value in real world applications like the crowdsourcing approach of TechnoWeb's urgent request. In TechnoWeb 2.0 it has lead to a significant reduction of the number of emails sent while delivering an acceptable response rate.

5.5. Suggesting Tags for a Full Text

In order to make tagging easier for a user, having tag suggestions for an entered text (such as a blog post) is a very handy feature. Users are encouraged to provide more and "better" tags. Typically a blog post, a wiki page or a question in a forum has a title and a text body. Examples, where pages or blog post have this form, are Wikipedia, Yahoo! Answers, Atlassian Confluence, Word Press and many others. For the sake of generalization, other structural elements of texts, such as sections, are ignored. Additionally, the title of an entity is considered optional for the developed algorithm. In some cases, such as comments on a page or blog post, a title is not always required. For example, comments in disqus,[15] a popular comment service, have no title.

Figure 5.17 depicts an example for a case where a post has a title and text body. In this

[15] `http://disqus.com/`

example, on the left there is an excerpt taken from the Wikipedia page on Social software. On the right suggested tags are displayed. The figure contains the actual suggestion generated by the algorithm described in this section – based on the social tagging data from the Siemens intranet. Worth mentioning is that there is no enforcement of a certain tagging practice.

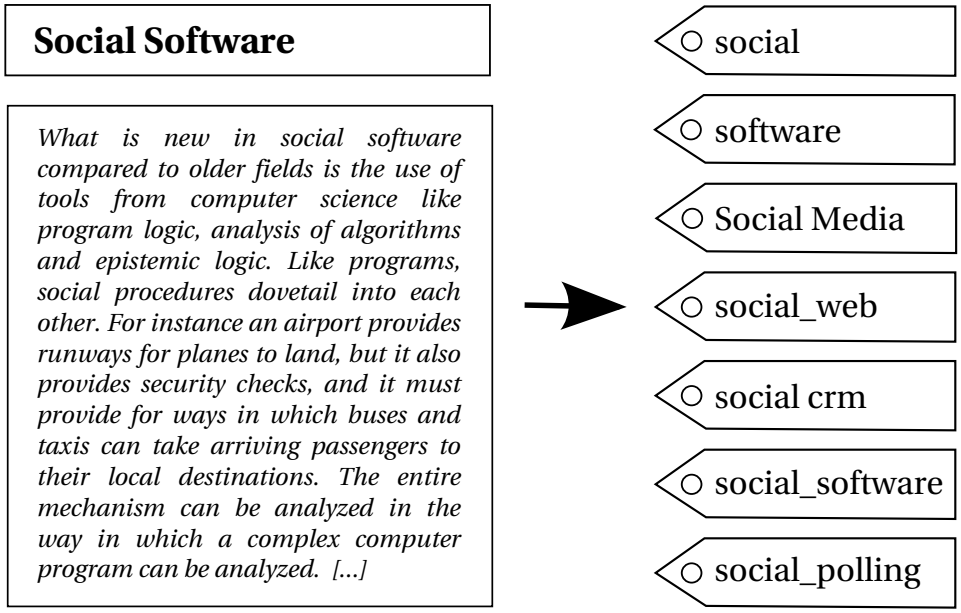


Figure 5.17: *Tag Suggestions for a text. For a given text (in the example an excerpt of the Wikipedia article on Social Software) and a title, tags are suggested.*

5.5.1. Algorithm

An algorithm for suggesting tags for a given text is shown in Algorithm 3. The chosen presentation is a descriptive form of pseudo code for the sake of readability. The algorithm takes as input a `title`, a `text`, and parameters `p` that correct the results of the output:

1. *smoothingTerm:* Has the role to damp the contribution of the second term. Its range is in [0;1].

2. *weightFolkTitle:* This determines the importance of the frequency of the suggested tag (originating from the title) in the folksonomy versus its frequency in the text (of the title). Its range is in [0;1]. Higher values (> 0.5) means that the frequency in the folksonomy is more important. Lower values (< 0.5) means that the frequency in the text is more important. 0.5 means that both frequencies are weighted equally.

3. *minLevenshteinDistanceTitle:* Specifies how exactly a word in the title has to match against a tag in the folksonomy – determined by the Levenshtein distance. Its range is in

[0;1]. A value of 1 means there has to be an exact match. Lower values mean a fuzzier match.

4. *weightFolkText:* It is the same as *weightFolkTitle*, but applied to the tag suggestions origination from the text body part.

5. *minLevenshteinDistanceText:* Analogous to *minLevenshteinDistanceTitle* it specifies how exact the suggested tags have to match against word in the text body.

6. *limit:* The limits the maximum number of returned tag suggestions.

Algorithm 3 Suggest Tags for Text

function *tagSuggestionsForText*(title, text, p)
  sugTitle ← *tags* found in *title* matched with *p*[*minLevenshteinDistanceTitle*]
  sugText ← *tags* found in *text* matched with *p*[*minLevenshteinDistanceText*]
  suggestions ← []
  for all *t* in *sugTitle* ∪ *sugText* **do**
    compute rank r_t for *t* with equation 5.11 using parameters from *p*
    insert (t, r_t) to *suggestions*
  end for
  sort *suggestions* by rank in descending order
  result ← the first *p*[*limit*] elements of *suggestions*
end function

The ranking function of suggested tags is shown in equation 5.11.

$$rank(tag_i) = (a + (1-a)\frac{freqFolk(tag_i) * w_{folk}}{maxFreqFolkFoundTag}) + (a + (1-a)\frac{freqText(tag_i) * (1-w_{folk})}{maxFreqTxtFoundTag}) \tag{5.11}$$

A suggested tag_i gets a ranking value based on its frequency in the folksonomy and the frequency in the text and title of the text. Both frequency values are normalized against the maximum value of all suggested tag in the folksonomy (*maxFreqFolkFoundTag*) and the text (*maxFreqTxtFoundTag*), respectively.

The smoothing term a reduces the influence of very frequent tags (either in the text or in the whole folksonomy). This is analogous to the "maximum tf normalization" used in information retrieval (a variation of the popular tf-id function) – see [MRS08].

5.5.2. Tests

The algorithm has been tested against urgent requests posted in TechnoWeb. 46 urgent requests from July 2011 have been selected in order to evaluate the algorithm and determine reasonable input parameters. This is not a random sample, but one can assume that there is no real correlation between this special month and the validity of the tag suggestions is

only marginal. The number of tag suggestions has been limited to 15. This is a reasonable number of tags that can be displayed to a user in an adequate web user interface.

The previously described algorithm has six input parameters. The first five parameters are within a theoretical range of [0;1]. Some parameter combinations make no sense. Having a Levenshtein distance threshold below 0.6, too many wrong tag against tag matches would be found. A smoothing term below 0.3 is also not reasonable. In order to limit the number of permutations and as consequence the time needed for the computation weightFolkText and weightFolkTitle have been set to the same value for every single test. For ranges a step value of 0.1 has been chosen. In the table "]" means including the value, "[" means exclusively the value.

Parameter	Range
smoothingTerm	[0.3; 1.0[
weightFolkTitle and weightFolkText	[0.1; 1.0]
minLevenshteinDistanceTitle	[0.6; 1.0[

Table 5.13: *Test Parameters Tag Suggestions for Full Text*

Table 5.13 contains the parameter ranges for the test. A test parameter combination is for example "(0.3; 0.9; 0.9; 0.9; 0.9; 10)". This stands for: smoothingTerm= 0.3, weightFolkTitle= 0.9, minLevenshteinDistanceTitle= 0.9, weightFolkText= 0.9, and limit= 10. There are 704 permutations resulting of the combination of the different parameters with the possible values.

Table 5.14 contains parameter combinations with best matches of the computed tags against the actual user assigned tags. The best parameter permutation could predict about 30 % of the assigned tags (28 % if the tags had to matched exactly, 31 % if there where little errors tolerated, such as "events" vs. "event").

The tag suggestion algorithm is used in production for TechnoWeb since summer 2011. For experiences with the service see chapter 6.7 and the interview in appendix A.4. In general the suggestions are good enough regarding to user experience in the daily usage. A "golds standard", such as the Reuters data set[16] typically used in the evaluation of algorithm for text classification, is unfortunately not available in the area of social tagging.

[16]Available at `http://archive.ics.uci.edu/ml/datasets/Reuters-21578+Text+Categorization+Collection`

smoothing-Term	weight-FolkTitle	min Levensht. dist. title	weight-FolkText	min Levensht. dist. text	mean ratio exact match	mean ratio approx. match
0.3	0.8	0.9	0.8	0.9	0.28	0.31
0.4	0.9	0.9	0.9	0.9	0.27	0.31
0.3	0.9	0.9	0.9	0.9	0.27	0.31
0.4	1	0.9	1	0.9	0.27	0.31
0.4	0.8	0.9	0.8	0.9	0.27	0.31
0.3	1	0.9	1	0.9	0.27	0.30
0.3	0.8	0.8	0.8	0.8	0.27	0.30
0.5	0.8	0.9	0.8	0.9	0.27	0.30
0.6	0.8	0.9	0.8	0.9	0.27	0.30
0.7	0.8	0.9	0.8	0.9	0.27	0.30
0.8	0.8	0.9	0.8	0.9	0.27	0.30
0.9	0.8	0.9	0.8	0.9	0.27	0.30
0.3	0.7	0.9	0.7	0.9	0.27	0.30
0.5	0.9	0.9	0.9	0.9	0.27	0.30
0.6	0.9	0.9	0.9	0.9	0.27	0.30

Table 5.14: *Best 15 parameter combinations for the arithmetic mean of the approximate tag matches.*

5.5.3. Discussion

The presented algorithm is a very straightforward approach that can be implemented without too much effort using standard libraries – in Java for example, Apache Lucene. There are other more complicated algorithms, such as ones building on machine learning algorithm. For example, Hess et al. [HDM08] describe an approach using Naive Bayes and Rocchio classifiers to generate tag suggestions. They tested their approach on the Reuters data set[17] and social tagging data originating from the Lycos IQ[18], a question and answer platform comparable to Yahoo! Answers[19]. For the more interesting social tagging data test, they reached a precision value of about 32 %. They compared their results to a kNN based algorithm similar to AutoTag [Mis06]. The kNN based algorithm seemed to be slower than their proposed algorithm and had a precision value of about 26 %.

Although the evaluation of the algorithm presented in this work is based on less data, the results indicate a similar precision value of about 30 %. This is quite a good result for a simplistic approach. Unfortunately, if the folksonomy is young and has too few tags the results of the algorithm are expected less suitable. Obviously, the same problem exists for algorithms based on machine learning – as already mentioned in the previous section. If there is too less data to rely on, the algorithm tends to be less useful.

A closed source web service called "tagthe.net"[20] is worth mentioning. In its FAQs it claims the analysis component to be built on Java open source libraries. Details about the implementation or either the used approach are not mentioned.

5.6. Mapping of External Structured Sources

There have been some attempts to map tags to concepts in an ontology. In general, a possible solution combines string distances, stemming algorithms and comparison of graph structures – both of the ontology on the one hand and the relations between users, tags and resources in a folksonomy on the other hand. As discussed in section 3.2, structured sources may be a valuable input. A general approach for a mapping of external structured sources has been published as patent (see [KZ10])[21]. A method for working with tags linked to structured sources are described in another prior patent (see [EK09]).

The mapping between tags and terms from external sources is typically incomplete and error-prone. This section contains an evaluation on how tags match terms in structured sources when only the actual string value is considered. Depending on the kind of desired structural information, not only full-blown ontologies, but also thesauri such as WordNet (Princeton University – English), Wortschatz (University of Leipzig – German) or GermaNet

[17] Reuters-21578, a test collection for text categorization research available at `http://www.daviddlewis.com/resources/testcollections/reuters21578/`

[18] Lycos IQ has been renamed to COSMiQ in 2011. Lycos Europe Discontinued its services in 2009.

[19] `http://answers.yahoo.com/`

[20] `http://www.tagthe.net/`

[21] Worth mentioning: The author is no fan of the current patent system. The patents were obtained because of contractual commitments.

(University of Tübingen – German) and other structured input such as DMOZ[22] or DBpedia[23] can contain valuable structure information.

5.6.1. External Structured Sources

In order to determine if a mapping can deliver useful results, a mapping of three different sources to social tagging data (described above) has been tested: *WordNet, DBpedia, DMOZ*. Since most of the tags are in English, no tests with non-English sources were conducted.

WordNet's latest version is 3.1, as of June 2011. WordNet differentiates between words belonging to different lexical categories, to be specific: nouns, verbs, adjectives and adverbs. They are grouped into sets of cognitive synonyms (*synsets*), where each synset represents a distinct concept and is referenced by a unique id. WordNet can be referred to as concept based thesaurus. Having synsets defined by their ids, a distinction between homonyms is possible. A synset has several types of labels, such as a preferred label and alternative labels. Each synset can have one preferred label and several alternative labels.

WordNet consists out of about 117,000 synsets that have links to other synsets representing a certain semantic relations. Depending on the lexical category of a synset, different types of relations are possible. For example, the relations between synsets for nouns are hypernym, hyponym, coordinate term, holonym, and meronym. For the mapping test all types of lexical categories where treated equally and only the presence of term was tested. 147,306 different words (labels) were contained in the downloaded version (2nd March of 2011).

DBpedia is a RDF version of Wikipedia, a project by FU Berlin. By parsing certain templates (for example information boxes for cities) or syntactic constructs articles are transferred into a structured form. Following DBpedia sources were investigated:

- *Categories:* Each article can be assigned to a category which itself can be assigned to another category.

- *Redirects:* Certain terms redirect to pages that are assumed to be meant by an user. An example is "GIS" that redirects to "Geographic information system" since it is likely the most common usage of the acronym.

- *Disambiguations:* When certain terms have homonyms, in Wikipedia pages for disambiguation are introduced. For example, the term Java stands for the article related to the island. Under "Java (disambiguation)" other meanings such as articles about Java as programming language can be found.

Each of these data sources in DBpedia contains some kind of weak semantic relations. Categories reflect broader and narrower term relations. Redirects contain potential synonym relations. Disambiguations help find fuzziness in defined relations. The used version (from 21st January of 2010) contained 7,419,435 syntactically different words.

[22] `http://www.dmoz.org/`
[23] `http://www.dbpedia.org`

DMOZ also known as open directory project is the largest human created web directory of the Internet. There is a (slightly messy) RDF version available for download. DMOZ has been interpreted as very big taxonomy. In the used download (25th of October 2010) there were 767,132 different nodes present in the taxonomy. After normalization there were 364,851 unique terms present.

5.6.2. Method

Each structured information source was in a first step indexed with Lucene[24]. In a second step tags were mapped against terms by using a Levenshtein distance metric with threshold: $\{0.1, 0.2, \ldots, 0.9\}$. The Levenshtein distance is a standard string matching algorithm. The algorithm basically counts the number of letters that have to be changed to transform one word to the other. For getting a value between 0 and 1 the number of changes is divided by the maximum of the lengths of the two words. The distance metric is used as similarity measure. Higher values mean there are fewer letters to alter in order to translate one word into the other. For example, "event" and "events" have an absolute Levenshtein distance of 1. The maximum word length of both words is 6 and therefore the relative Levenshtein distance is $\frac{1}{6}$. The Levenshtein similarity is $1 - \frac{1}{6} = \frac{5}{6} \approx 0.83$, which means both words are very similar.

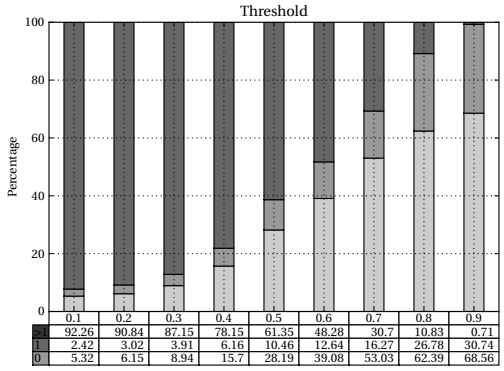
5.6.3. Results and Discussion

Figure 5.18, Figure 5.19, and Figure 5.20 show the results of the mappings for the three structured information sources. Shown is the percentage of tags that where mapped with the different thresholds: light gray means no match, gray exactly one match and dark gray more than one match. An ideal result would be that for each tag exactly one match exits and that each match maps a tag to its semantic counterpart. Even though only one match for the mapping has been found this does not mean that the match is correct. It still can be the wrong semantic variant of a homonym. Having more than one mapping for a tag to a term is slightly better than one match. A user interaction is needed, but having more than one suggestion is better than none. Decreasing threshold values lead to more fuzziness in the mapping, because there is more syntactic variation that leads to a match.

Figure 5.18 contains the results for WordNet. It shows for low thresholds that many tags have more than one mapped term. High threshold leads to more tags having no match at all. Considering all tags including the long tail[25] have worse results than if only the top 100 tags are included. For increasing thresholds, the top 100 tags of Siemens have slightly worse mapping results than the tags from Delicious. The opposite is the case when all tags are considered.

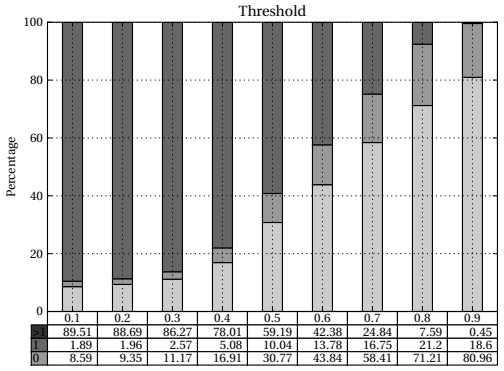
The results for mapping tags to DBpedia terms are shown in Figure 5.19. The top 100 tags lead to very good results for high thresholds. For a Levenshtein distance threshold of less than 0.9, a majority of tags have exactly one match. For the case when all tags are included,

[24] A popular search engine library written in Java: `http://lucene.apache.org/`.
[25] Less frequently used terms – see chapter 1.



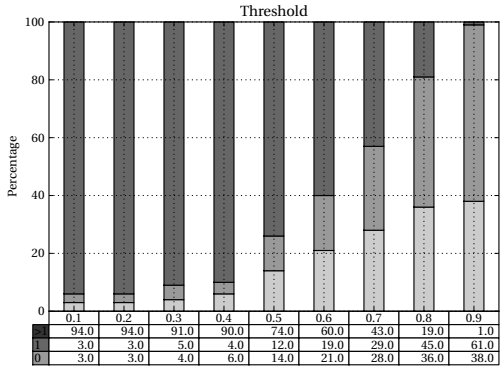
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
>1	92.26	90.84	87.15	78.15	61.35	48.28	30.7	10.83	0.71
1	2.42	3.02	3.91	6.16	10.46	12.64	16.27	26.78	30.74
0	5.32	6.15	8.94	15.7	28.19	39.08	53.03	62.39	68.56

(a) Siemens all Tags.



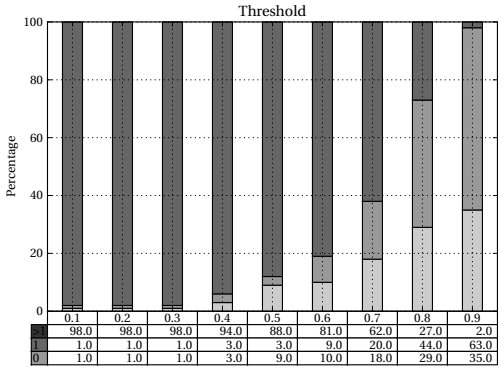
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
>1	89.51	88.69	86.27	78.01	59.19	42.38	24.84	7.59	0.45
1	1.89	1.96	2.57	5.08	10.04	13.78	16.75	21.2	18.6
0	8.59	9.35	11.17	16.91	30.77	43.84	58.41	71.21	80.96

(b) Delicious all Tags.



	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
>1	94.0	94.0	91.0	90.0	74.0	60.0	43.0	19.0	1.0
1	3.0	3.0	5.0	4.0	12.0	19.0	29.0	45.0	61.0
0	3.0	3.0	4.0	6.0	14.0	21.0	28.0	36.0	38.0

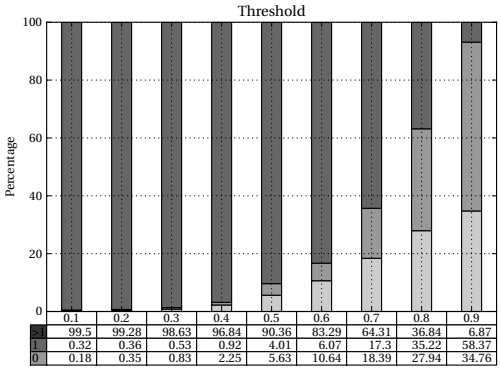
(c) Siemens 100 Most Frequent Tags.



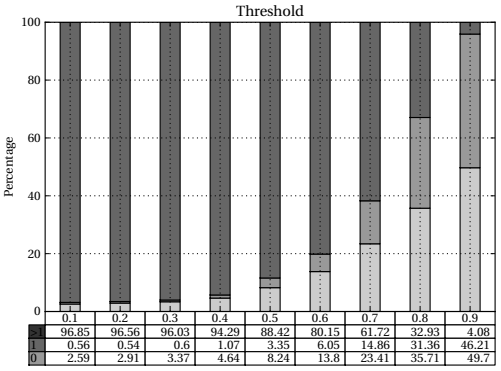
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
>1	98.0	98.0	98.0	94.0	88.0	81.0	62.0	27.0	2.0
1	1.0	1.0	1.0	3.0	3.0	9.0	20.0	44.0	63.0
0	1.0	1.0	1.0	3.0	9.0	10.0	18.0	29.0	35.0

(d) Delicious 100 Most Frequent Tags.

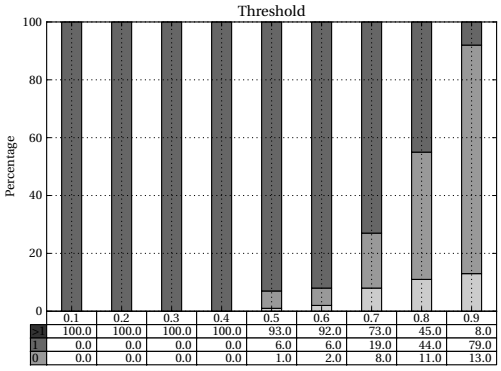
Figure 5.18: *Mapping tags to **WordNet** using Levenshtein distance metrics with thresholds: {0.1, 0.2, . . . , 0.9} – represented by stacked bars. The y-axis shows the percentage of tags matching a term with light gray: no matches, gray: exactly one match and dark gray: more than one match.*



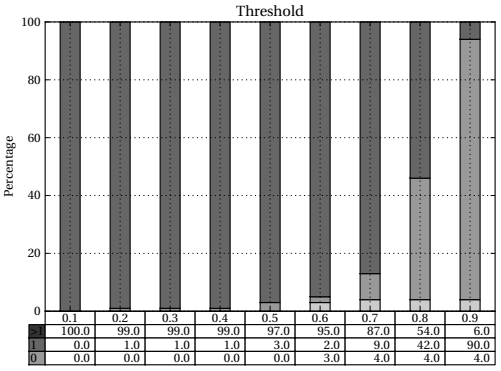
(a) Siemens all Tags.



(b) Delicious all Tags.



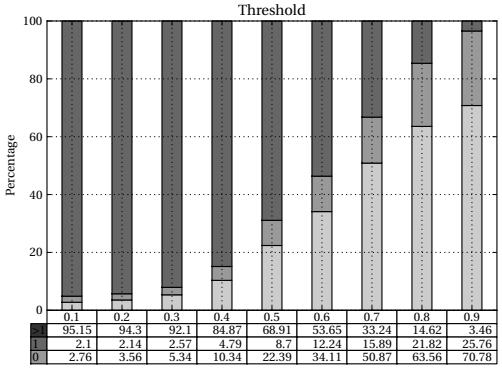
(c) Siemens 100 Most Frequent Tags.



(d) Delicious Most 100 Frequent Tags.

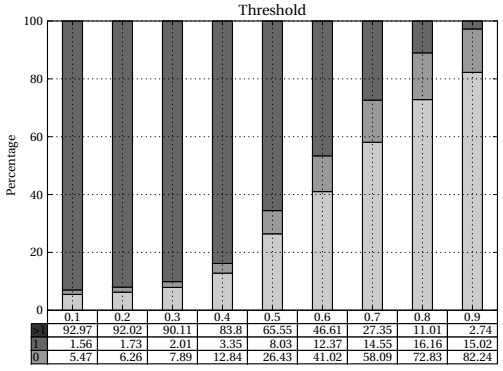
Figure 5.19: *Mapping Tags to **DBpedia** using Levenshtein distance metrics with thresholds: {0.1, 0.2, . . . , 0.9} – represented by stacked bars. The y-axis shows the percentage of tags matching a term with light gray: no matches, gray: exactly one match and dark gray: more than one match.*

the resulting mappings are slightly worse for the tags from Delicious than for the ones of Siemens.



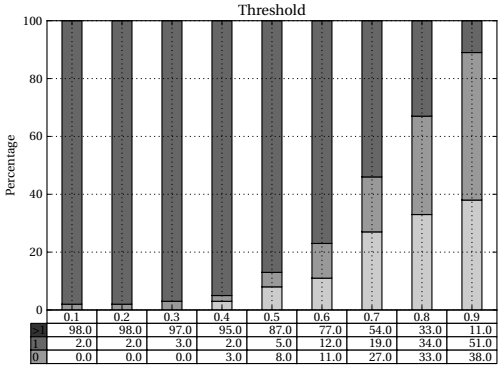
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
>1	95.15	94.3	92.1	84.87	68.91	53.65	33.24	14.62	3.46
1	2.1	2.14	2.57	4.79	8.7	12.24	15.89	21.82	25.76
0	2.76	3.56	5.34	10.34	22.39	34.11	50.87	63.56	70.78

(a) Siemens all Tags.



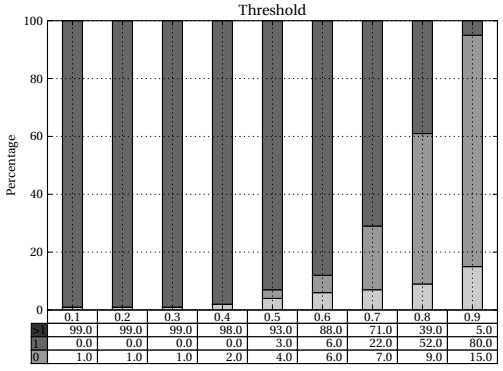
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
>1	92.97	92.02	90.11	83.8	65.55	46.61	27.35	11.01	2.74
1	1.56	1.73	2.01	3.35	8.03	12.37	14.55	16.16	15.02
0	5.47	6.26	7.89	12.84	26.43	41.02	58.09	72.83	82.24

(b) Delicious all Tags.



	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
>1	98.0	98.0	97.0	95.0	87.0	77.0	54.0	33.0	11.0
1	2.0	2.0	3.0	2.0	5.0	12.0	19.0	34.0	51.0
0	0.0	0.0	0.0	3.0	8.0	11.0	27.0	33.0	38.0

(c) Siemens 100 Most Frequent Tags.



	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
>1	99.0	99.0	99.0	98.0	93.0	88.0	71.0	39.0	5.0
1	0.0	0.0	0.0	0.0	3.0	6.0	22.0	52.0	80.0
0	1.0	1.0	1.0	2.0	4.0	6.0	7.0	9.0	15.0

(d) Delicious 100 Most Frequent Tags.

Figure 5.20: *Mapping tags to **DMOZ** using Levenshtein distance metrics with thresholds: {0.1, 0.2, . . . , 0.9} – represented by stacked bars. The y-axis shows the percentage of tags matching a term with light gray: no matches, gray: exactly one match and dark gray: more than one match.*

Figure 5.20 shows the results for the mappings to DMOZ. If all tags are considered even for high thresholds, there are view one-to-one mappings. Considering only the top 100 tags there are more one-to-one mappings for both data sets. In general, the best results deliver the mapping of DBpedia terms for both cases when all and when only the 100 most frequent tags are considered. The mapping to DMOZ seems to be a little better for the top 100 Delicious tags and comparable for the other cases.

A perfect mapping of all tags to their corresponding term is not possible for several reasons. On the one hand there are the problems with the fuzziness of tags – see section 2.2.2. On the other hand not every tag is contained in the target set of terms. For example terms

only used inside Siemens are not very likely to find a corresponding concept in WordNet. This is also applicable for other new or specific tags, such as “toRead”or “webdev”.

5.7. Semi-Automated Approach: Tag Thesaurus Editor

As it turns out the results of statistical analysis of folksonomies or the mapping of tags to ontologies are not accurate enough to generate a suitable thesaurus. Automatically computed similarities may contain errors depending on the folksonomy data, e.g. as a result of different tag usage patterns [GH05]. Mapping tags to concepts of an ontology is equally error-prone since normally not all tags are contained as concepts in an ontology. Additionally, ambiguities of terms (e.g. homonyms or acronyms) might not be resolvable. In contrast, a more precise manual creation of thesauri (following a formal procedure) is expensive and time consuming. Therefore a semi-automatic approach has been developed. The results of the statistical analysis and mapping of tags to pre-existing structures are used as input for a thesaurus editor. It consists of a web interface that can be easily used by a non-expert person.

Summing up, there are three categories of relations between tags:

- statistically computed ones,
- relations found through mapping tags to concepts in an ontology, and
- the manually defined relations between tags.

To determine a semantic similarity between tags each of these kinds of relation can be treated differently and taken into account. Manually defined relations between tags are considered the most valuable ones since a user has defined them. Manually created relations are less likely to be wrong than algorithmically inferred ones.

The results of the statistical analysis and ontology mapping are only suggestions for relations between tags. A user decides whether a proposed relation is correct or not. Only verified relations are included in the final thesaurus.

To make this process as easy as possible, a user can formulate thesaurus relations through a web based thesaurus editor with a drag and drop style interface. With this editor the user can express his or her personal opinion that one tag has some certain relation to another tag, e.g. “ajax” can be a narrower term of “web2.0”. By proposing tag relations based on the described automatic methods the process of creating a thesaurus is simplified since in many cases the user only has to confirm tag relations and does not have to think about these relations but is still able to express additional relations between tags.

The thesaurus editor enables the user to extend his or her own personal tag space with more structure. These relations can be a very personal view with which another user might disagree. But in contrast, given that many users have formulated the same relation between two tags, it is assumed with some certainty that these relations reflect a broader consensus.

For information retrieval and navigational use case arguable following relations are considered most useful:

• *Use synonym*: The tags can be used interchangeably. In general one can distinguish different levels of synonymy. Words can have the exact same meaning or only in some context. For the sake of simplicity those cases are not separated. An example for synonyms is “person” and “individual”.

• *Broader term*: A tag has a more general meaning than another term e.g. “mammal” is a broader term of “primate”.

• *Narrower term*: A tag has a narrower meaning than another term e.g. “primate” is a narrower term of “mammal”.

• *Related term*: This is the weakest relation. Two tags are only related in some way, e.g. “web2.0” and “ajax”.

Additionally, an artificial relation with the name *Ignore Relation* is defined. This relation allows an user to explicitly express that there is no relation between two tags. An automatic algorithm may infer undesired relations that a user can then dismiss.

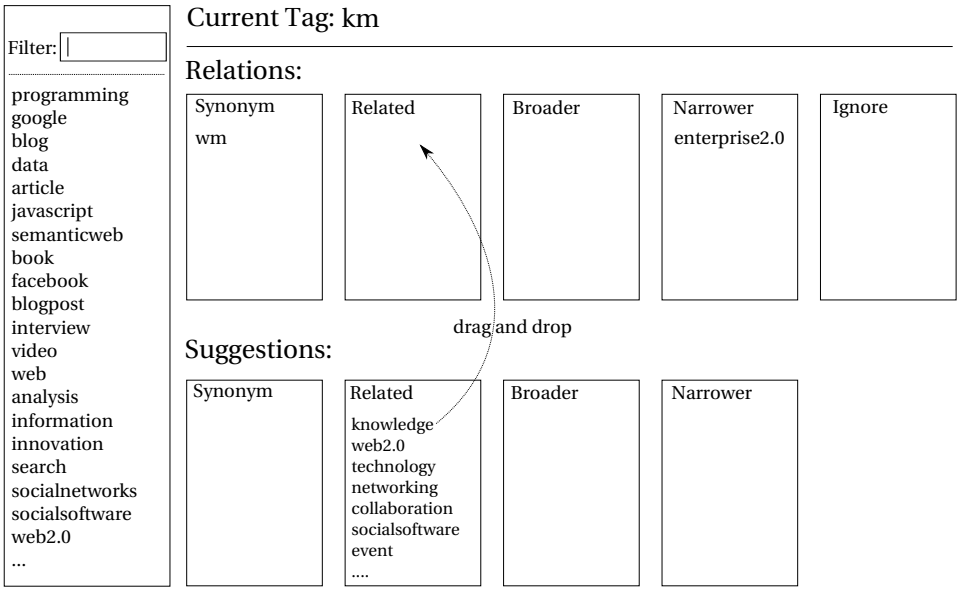


Figure 5.21: *Thesaurus Editor user interface mock up*

User interface design is a tough discipline and there are many traps to avoid. For web design there are many books available. A popular book is “Don’t make me think” [Kru05]. Another classic guide to web design is “Designing Web Usability: The Practice of Simplicity” by Nielsen [Nie99]. By following the best practices in the mentioned books one can create a sophisticated yet simple user interface. The user interface design methods are not in the focus of this work. Hence, a tedious design process typically conducted for UIs developed in the field of human computer interaction (HCI) has been dispensed with.

In order to increase user adoption and to make the definition of relation as simple as possible, a sophisticated and yet powerful user interface has to be provided. Figure 5.21

shows a mockup of the proposed solution for the user interface. A screenshot of an actual implementation of the mockup is displayed in figure 6.7 in chapter 6.

The defined relations between two tags by a user have to be stored in a data structure that can manage tuples of the following form:

Definition 5.5 (Tag Thesaurus Relation) *A tag thesaurus relation is a tuple ttr of the form* (u, tl, r, tr, t) *with*

- $u \in U$ *(an user),*
- $tl \in T$ *(the tag on the left side of a statement),*
- $tr \in T$ *(the tag on the right side of a statement),*
- $r \in \{synonym, broader, narrower, related, ignore\}$ *(the type of relation)*
- *and* t *is a timestamp (the time the relation has been defined).*

For example, if an user with the id "*id123*" defines that "*java*" is a narrower term to "*programming*" at noon on the first of January 2011 (GMT), then tuple (*id123*, *java*, *narrower*, *programming*, $2011-01-01T12:00Z$) has to be stored in an adequate manner. The opposite direction, meaning "programming" is a broader term to "java" should not be stored automatically, but can be provided as a suggested relation. The used relations in the tag thesaurus are not transitive in general.

Combinations of Resulting Thesauri

The aggregation and interpretation of these tuples are explained in the following sections. There are three major usages for the thesaurus editor:

- *Individual Tag Thesaurus:* The thesaurus relations are only used for each individual user separately.
- *Weighted Social Tag Thesaurus:* The thesaurus relations of all users are aggregated in a global tag thesaurus. Each tag relations gets a weight according to the frequency the relation has been defined by individual users.
- *Collaborative Tag Thesaurus:* A group of people define their tag thesaurus in collaborative manner.

Individual Tag Thesaurus

An individual tag thesaurus is constructed by a person based on the set of tags he or she has used himself or herself. It is independent of other people's thesaurus in the sense that it is not synchronized with other individual tag thesauri. This helps organizing the *personal* information space. For example, if someone decides to use a different tag variation for *(nearly)* the same thing, then a synonym relation is defined. This can be for example for the

tags “dhtml” and “ajax”. Sometimes one uses an abbreviation instead of a whole word, for example, “km” instead of “knowledge_management”. Another case is when someone uses a different tag then the rest of the users in a system then he can define his term and the other term synonym. This is especially useful where terms are not very close in terms of a String distance metric.

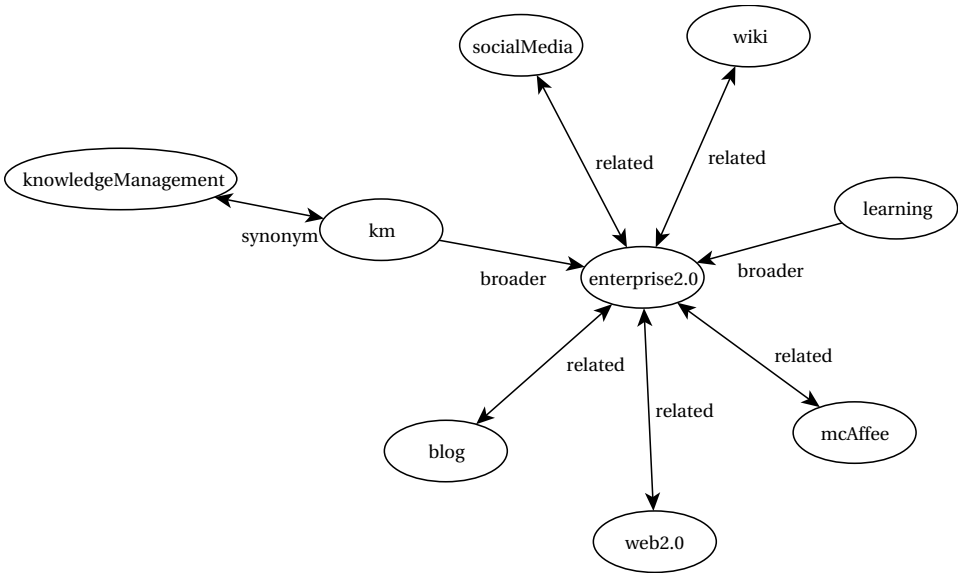


Figure 5.22: *Example for an individual tag thesaurus. Nodes are tags. Relations are represented by labeled directed edges.*

Figure 5.22 shows an excerpt of an example for an individual tag thesaurus. Each tag is represented by a node. The defined relations are shown by directed edges with a corresponding label. The main tag in this example is “enterprise2.0”. The person has defined several relations, such as “enterprise2.0” being related to “mcAffee,” “km” being a broader term to “enterprise2.0” or “km” being synonym to “knowledgeManagement” and vice versa. The timestamp is not included in this representation.

Weighted Social Tag Thesaurus

The weighted tag thesaurus is the result of aggregated tag thesaurus relations of all users in the tagging framework. The relations in the aggregated tag thesaurus have weighted edges for each relation between two tags. The weight is a normed value in [0;1]. It reflects how many users have defined a certain relation between two tags. Each time a relation is defined by a user it is some kind of vote for that relation. The scaling can be achieved by various *norms*.

An example for a norm is dividing the frequency of a relation between two tags by the maximum frequency that one of these tags has be used in a relation with other tags. Other norms, such as a scalar norm or a cosine distance may be applicable as well. The weight reflects some kind of measurement for the certainty of a relation. Higher values express

that a relation is likely to be true for a broader base of users and therefore having a higher probability. Before the actual scaling a threshold for a minimum frequency a relation between two tags has been defined. This depends strongly on the individual usage patterns of the thesaurus editor.

Figure 5.23 contains an excerpt of a resulting weighted social tag thesaurus. The central tag is “enterprise2.0”. Several weighted relations have emerged through the individual tag thesauri. For example “learning” is a broader term of “enterprise2.0” with a weight of 0.57. “web2.0” is related to “enterprise2.0” with a weight of 0.9. This means that the certainty of “web2.0” is related to “enterprise2.0” is higher than the broader term relation of “learning” and “enterprise2.0”.

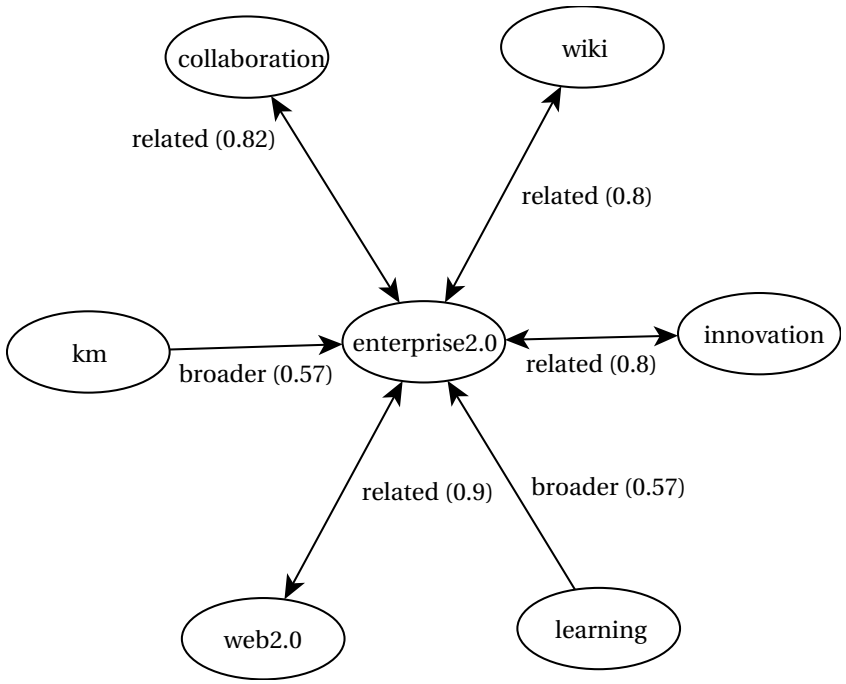


Figure 5.23: *Example for a Weighted Tag Thesaurus*

A weighted tag thesaurus helps collecting the individual tag relations in a “democratic" way and make it accessible for a more general application. This can be the case for application with big folksonomies that are hard to match to existing thesauri and where an editorially created thesaurus is too expensive. Sometimes, also the type of relations is hard to define even for an expert in a domain. An example is a music community, such as last.fm. The terminology for different music genres is fuzzy and can become quite complicated to understand. Terms such as “alternative rock,” “indie,” “rock,” “indie rock,” “emo,” “grunge,” “metal,” “nu metal,” “thrash metal,” “heavy metal,” “power metal” or “metalcore” are somehow related. The exact type of relation can depend strongly on the person that states the relation.

Collaborative Tag Thesaurus

A collaborative tag thesaurus is a thesaurus constructed by several individuals in a common effort. This can be, for example, members of a department working on internal communication. In this scenario the thesaurus editor works similar to the individual tag thesaurus. The difference is, instead of having one person build his or her personal thesaurus, a general thesaurus is created by several people in a collaborative manner. The resulting thesaurus is analogues to the individual tag thesaurus in that way that a relation between two tags is only defined once. Hence, there are no weights on the edges. The resulting graph is comparable to the one in figure 5.22.

Taking the internal communication example a collaborative tag thesaurus can help this kind of department to filter user generated content. Having defined that “green_it,” “green_building,” “sustainability,” “ethical_consumerism” and other terms are related to each other, the classical information retrieval problem of having too less recall can be damped. Depending on the application, only synonym or other types of relations can be selected.

A collaborative tag thesaurus can also be useful in a scenario where a group of people do research on a certain topic. They collect information snippets such as bookmarks to web pages or text excerpts from articles and tag them. Having a tag thesaurus helps here providing an improved semi-structured access to these pieces of information.

CHAPTER 6

STAGS: Implementation of a Social Tagging Framework

I am a design chauvinist. I believe that good design is magical and not to be lightly tinkered with. The difference between a great design and a lousy one is in the meshing of the thousand details that either fit or don't, and the spirit of the passionate intellect that has tied them together, or tried. That's why programming – or buying software – on the basis of "lists of features" is a doomed and misguided effort. The features can be thrown together, as in a garbage can, or carefully laid together and interwoven in elegant unification, as in APL, or the Forth language, or the game of chess.

— Ted Nelson (*1937)

Contents

The target of this chapter is to describe the actual implementation of the proposed tagging framework. It is the artifact constructed for the evaluation of the suggested approach. The implementation decisions are discussed and mapped to the requirements inferred from the use cases – see chapter 4. STAGS (**S**iemens **Tag**ging **S**ervice) is the name chosen to refer to the implementation. Some parts are currently (since 11/2010) used in production inside the Siemens intranet – therefore the 'S' in the acronym. The implementation proves the validity – in practice – of the overall design proposed in chapter 5. Having "real world" simplifies data the evaluation of the system. The experience with its usage inside Siemens are described in section 6.7. Expert user interviews and log file analysis provide a strong evidence for the quality of the design and its implementation. The evaluation of the implemented system is based on the framework of the Updated DeLone and McLean information system (IS) success model (D&M model) [DM03].

6.1. Architecture Overview

The following section discusses the considerations made for the implementation of the tagging framework. In general, the design decisions are based on the requirements described in chapter 4 and the specification of chapter 5. Figure 6.1 depicts an architectural overview of the implementation. STAGS is a system consisting of 4 layers having several internal modules: Data Persistence, Data Access, Data Analysis and Client Interface. Dividing the architecture into 4 layers helps enforcing the design principle of *separation of concerns* – see for example see [Mic09][1]. Especially, when using a design science approach[2] where many iterations and refinements occur, this design paradigm is advisable to apply. A module that has been improved can be easily replaced without affecting other components – for the ideal case.

[1] Note this book is from 2009. The term was most likely introduced by Edsger W. Dijkstra [Dij82].
[2] Or typically for modern software engineering an agile approach.

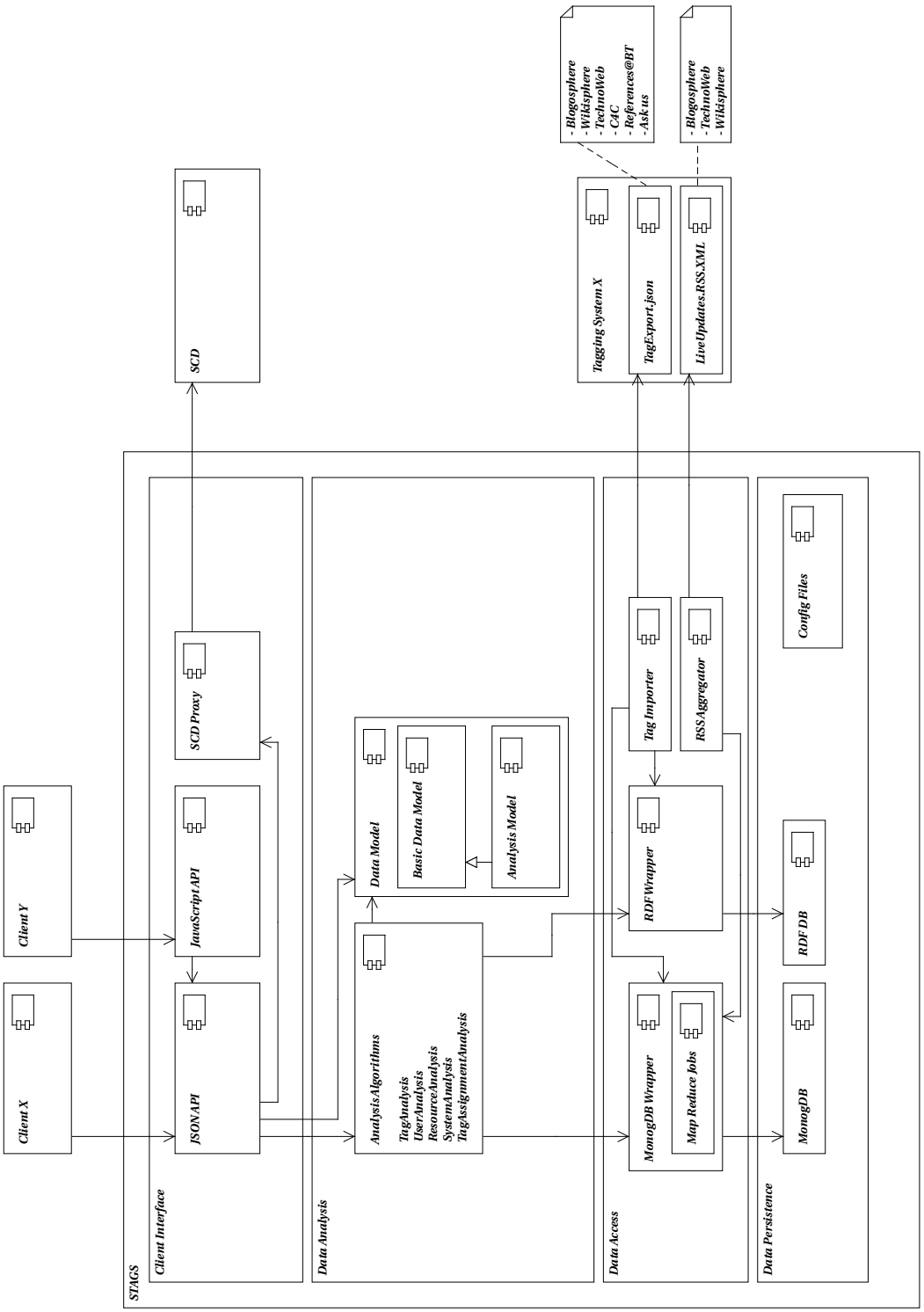


Figure 6.1: *Architecture of the tagging framework – STAGS.*

6.1.1. Data Persistence

At the bottom there is the persistence layer that deals with storing the needed data. There are two data base systems involved. MongoDB[3] and a RDF storage engine[4]. MongoDB is a fast light-weight key value store written in C. The currently used RDF store Jena is a Java implementation of a RDF triple store originally developed as open source software by HP Labs in Bristol and later donated to the Apache Software Foundation.

At the early stages during the creation of STAGS only a RDF triple store has been used as a persistence service for the social tagging data. Later in the development process, especially when STAGS was adopted for production, it turned out that there was no RDF triple store implementation that fulfilled the performance requirements for all use cases. Especially real time analysis, for example, aggregation function such as count or min/ max did not match nearly the needed velocity. MongoDB has excellent performance characteristics. Complex aggregation function are done with a map reduce approach. But since SPARQL as query language is very useful for querying graph structures, a RDF triple store is still included in STAGS. This might change when STAGS evolves in the near future.

6.1.2. Data Access

Above the persistence layer there is the data access layer. It consists of two services providing a single access point to the persistence layer (MongoDB Wrapper and RDFWrapper). Additionally Tag Importer and RSS Aggregator are responsible for importing and updating the social tagging data. The first one is a job that triggers a bulk export for each tagging application and fetches the exported data. The bulk export for each application is conducted once every night – at 02.00 GMT+00. The latter is a job that periodically fetches RSS feeds from applications supporting this mechanism. It is used to provide a near real time view on the activities in the social tagging applications. The RSS feeds in contrast to the bulk tag export do not contain the tag assignments, but resources with tags and authors for the corresponding resources. Who applied tags to a certain resource is not provided.

6.1.3. Data Analysis

In the center there is the data analysis layer. It is the main component where the analysis algorithms are implemented. For each part of a tag assignment (see 5), – user, tag, resource, system – and the tag assignment itself there is a corresponding analysis Java class. Each analysis class provides methods that return one or more (for example a Collection/List of) objects according to the provided parameter that matches the analysis class name. UserAnalysis returns users; TagAnalysis returns tags and so on. The analysis algorithms access the data through the corresponding wrapper provided by the data access layer.

[3] `http://www.mongodb.com`

[4] At first sesame (`http://www.openrdf.org`) then Jena (`http://www.jena-rdf.com`), because it turned out that Jena was faster for the major use cases.

6.1.4. Client Interface

At the top there is the client interface. It consists of three major parts. The JSON API provides a REST-like API for accessing analysis modules of STAGS using JSON as data exchange format. The JavaScript API consists of widgets that can be embedded into any web application via JavaScript tags. It is a convenience wrapper around the JSON API. An application owner does not have to deal with processing and rendering the returned JSON data. This approach is analogues to widgets provided by services in the Internet such as Google Maps widgets or tweet streams provided by Twitter. In figure 6.1 Client X uses the JSON API directly and Client Y makes use of the JavaScript API. Additionally, to both external APIs there is a SCD Proxy integrated that caches user based information for avoiding too many request to the Siemens Corporate Directory (SCD). The SCD is a data base where organizational information about employees is centrally accessible inside the Siemens intranet. Each employee has a unique and permanent id through which he or she can be identified.

6.1.5. Tagging Systems

On the right there is a Tagging System representing a social tagging application, such as Siemens Blogosphere or Wikisphere. An application that wants to make use of STAGS has to provide an interface where its tagging data can be accessed through a bulk export. Additionally for using the RSS update service it has to implement a RSS stream.

The main parts of the implementation of STAGS are in Java. The HTTP interface runs inside an Apache Tomcat[5], an open source web server that serves as Servlet container. Parts that run inside the browser of an user have been developed in JavaScript with the support of different JavaScript libraries. In the early stages of the implementation, Dojo[6] has been used especially for the thesaurus editor. For other widgets JQuery[7] was the library of choice, since it seemed more light-weight and easier to maintain. The following sections provide more details about the used formats and the implementation.

6.2. Design Decisions

Making the right design decisions affects the actual implementation significantly. Poor choices can lead to bad system performance and long development periods. These decisions have to be made based on the requirements and are the results of various iterations. As stated in the Manifesto for Agile Software Development[8]: "*[...] Working software over comprehensive documentation [...] Responding to change over following a plan [...]*".

[5] `http://tomcat.apache.org/`
[6] `http://dojotoolkit.org/`
[7] `http://www.jquery.com/`
[8] A public declaration of many well established software developers – see `http://agilemanifesto.org/`

6.2.1. Data Management

STAGS has two data storage components – see figure 6.1 MongoDB and RDF DB. MongoDB is a powerful NoSQL database. By reducing the typical feature set available in relational data bases, such as transaction, MongoDB gains performance advantages to these traditional data bases at the cost that it cannot be used in certain scenarios. The used RDF store is Jena. RDF stores are very flexible at storing graph data. SPARQL as query language allows a user to query the data in an elegant manner. The major drawbacks for RDF data bases is that they are typically slow in performance[9].

Semantic Web Technologies: RDF triple store

The Resource Description Framework (RDF)[MM04] is made up of a collection of World Wide Web Consortium (W3C) specifications. Basically, RDF is a graph representation format where statements of the form `<subject> <predicate> <object>` can be formulated. `<subject>` and `<predicate>` have to be URIs (see RFC 3986 [BLFM05] for the current specification), `<object>` can be either an URI or a literal (typically any primitive data type defined for XSD Schema [VH08]). Based on that pattern relations between people can be formulated for example this way: `<#alice> <#knows> <#bob>`. For the sake of simplicity some features are not mentioned here, such as blank nodes or reification.

RDF stores can be used for storing data when rapid prototyping of an application is desired. There is no need for a restrictive data schema, relations between nodes can be defined if needed. With SPARQL [FJ09] a powerful language for querying RDF graphs is available. SPARQL is a simple SQL like query language especially developed for dealing with RDF graphs.

Although ontologies (see section 2.3.3) are powerful concepts some of their features are typically not fully utilized in practice. This corresponds with the issue that ontologies of a higher-order logic can lead to problems that are undecidable – or at least there are only known algorithms having a NP complexity. NP complexity makes them unsuitable for large scale data. Hence amongst other reasons full-blown ontologies can be rarely found in productive environments.

For the implementation of STAGS reasoning was not a necessary feature. Hence, no complex ontology was used. While developing STAGS at first sesame[10] has been used. Sesame, as it turned out, had a worse performance for the main use cases than Jena[11] and therefore has been replaced with it in a later iteration.

NoSQL Store: MongoDB

MongoDB[12] is a high performance light-weight NoSQL store written in C. Data is stored in the form of collections and documents. Collections are comparable to tables in traditional,

[9] see the results of the Berlin SPARQL Benchmark (BSBM) [BS09] – new version under `http://www4.wiwiss.fu-berlin.de/bizer/berlinsparqlbenchmark/`

[10] `http://www.openrdf.org`

[11] `http://www.openjena.org/`

[12] `http://www.mongodb.org/`

relational data bases. Documents are analogues to rows. Documents in a collection are represented by a JSON object and can have a varying number of key value mappings. Hence the format is ideal for storing semi structured data. Additionally, JSON has been defined as data exchange format – see section 6.2.3. Having less data representation formats reduces the complexity of a system.

MongoDB has been introduced in addition to the RDF store for performance reasons. Some queries to the RDF store took a too long time. Simple tasks, such as computing a tag cloud had a response time of five or more seconds for a single client. Before the introduction of MongoDB a caching layer based on Ehcache[13] increased the overall performance of STAGS. A caching layer was sufficient for the most frequent accesses to STAGS. Some use cases such as suggesting tags for a user input involve too many permutations of user inputs for caching them effectively. Therefore, several alternative solutions have been discussed and after some tests MongoDB was selected as a supplement or in the long term as a replacement for the RDF store.

6.2.2. REST-like External API

For easy integration of the services offered by the tagging framework, a REST-like[Fie00] design has been chosen. REST is at the core a collection of design principles that a web application has to stick to. A central role plays the concept of a resource. A resource is anything that is important enough to be referenced as thing itself. This is the case: if users "*want to create a hypertext link to it, make or refute assertions about it, retrieve or cache a representation of it, include all or part of it by reference into another representation, annotate it, or perform other operations on it*" [14] (see [RR07]).

The identification of resources (the URL path) has to follow certain patterns. One major point is that an URL of a resource is independent from the state of a resource. Identifiers for a resource are not given by an URL parameter but are included in the path, e.g. `http://woidda.de/user/12345/` is used instead of `http://woidda.de/user?id=12345`. For modifying a resource one has to use the standard HTTP methods (`POST`, `PUT`, `DELETE`). Additionally GET is used for getting the content behind an URL. The desired format in which the content of a resource is returned is determined via content negotiation with corresponding information in the HTTP headers. The idea is that in a RESTful architecture, everything is modeled in terms of resources accessed through an URL, and the four basic operations – CRUD (create, read, update, delete) – are executed via the corresponding HTTP operations GET, POST, PUT, and DELETE.

The API of the tagging framework follows in general the design principles of REST - at least the naming conventions and content negotiation is supported. One major exception is the possibility to add, alter, and delete resources via HTTP PUT, POST, and DELETE. The reason for this is that in some cases it is not possible or at least complex to use the right HTTP methods – especially in the context of a browser. Calling an URL with included parameters is the most simple case and also easy to understand for external developers.

[13] `http://ehcache.org/`

[14] `http://www.w3.org/TR/webarch/#uri-benefits`

The mainly used data interchange format is JSON. JSON, or JavaScript Object Notation, is a simple machine-readable data-interchange format. It is natively supported in JavaScript which makes constructing API applications in JavaScript easy. There are parsers for nearly every popular programming language available. For more information about JSON, visit `http://json.org`.

JSONP is used to avoid errors related to the same origin policy enforced by all major browsers. JSONP is a script tag injection, passing the response from the server in to a user specified function. Examples for well-designed REST-like APIs offered by popular web sites supporting JSONP:

- Flickr API: `http://www.flickr.com/services/api/response.json.html`
- Google Data APIs: `http://code.google.com/apis/gdata/json.html`
- Twitter API: `http://apiwiki.twitter.com/w/page/22554756/Twitter-Search-API-Method`
- Delicious Data APIs: `http://www.delicious.com/help/api`
- Facebook: `http://developers.facebook.com/docs/authentication/javascript`
- LinkedIn: `http://developer.linkedin.com/community/apis/blog/2010/10/25/api-requests-with-json`

This list can be extended with many other examples. Its main purpose is to provide a justification for the chosen data format and show instances of its application in well-established Internet services.

6.2.3. Social Tagging Data Exchange Format

The definition of the social tagging data exchange format results out of the modeling consideration in chapter 5 and non-functional requirements discussed in chapter 4. JSON has been chosen as export format since it is a very simple format and there are parsers and generators for nearly every modern programming language available. Alternatives would have been to rely solely on RSS feeds or use another serialization format, such as XML or YAML[15]. Since JSON is the preferred data exchange format for JavaScript – the "X" in AJAX for XML has lost its authority. Furthermore it is typically easier to work with a single serialization format than to mix different format simply because different parsers APIs have to be used.

The advantage of RSS is that it is typically supported by web applications out of the box. A drawback of RSS is that for exporting tag assignments in that format the semantics of some fields are changed. RDF as exchange format was not accepted by some web application owner. Though RSS has not been the main data exchange format it is used as a format for aggregating near real time updates – see section 6.3.

[15] `http://yaml.org/`

```
{
    "system": "<URL>|<literal>",
    "tagAssignments": [
        {
            "resourceUrl": "<URL>",
            "resourceTitle": "<literal>",
            "tag": "<literal>",
            "user": "<GID>",
            "date": "<YYYY-MM-DDThh:mm:ss>",
        }
    ]
}
```

Listing 6.1: *Tag Export from a source application*

Listing 6.1 contains an abstract definition of the used format. Following data fields are defined:

• `system` (mandatory): The System identifier of application. Instead of an URL, an alias can be used, such as `technoweb` for `"http://technoweb.siemens.com"`. The parameter occurs only once since the export format is used by a single application.

• resourceUrl (mandatory): Points to the location where the tagged information item (network, urgent request, news ...) can be retrieved. If available, a permalink[16].

• `resourceTitle` (mandatory): A casual name for the information item that should be rendered in the user interface instead of an URL.

• `user` (mandatory): An unique identifier for the user that has assigned the tag, such as an email address. For the case of Siemens, the Siemens GID[17].

• `date`: Mandatory. The date the tag was assigned in ISO 8601 representation.

In listing 6.2 an example for a tag export is shown.

```
{
   "system": "technoweb",
   "tagAssignments": [
      {
         "resourceUrl": "https://technoweb.siemens.com/web/" +
                        "wiki-web-based-collaboration-platforms",
         "resourceTitle": "Wiki - Web Based Collaboration
            Platforms",
         "tag": "collaboration",
         "user": "Z0007F00",
```

[16] A permalink or permanent link is a link that "should" be stable and does not change over time – for example, implementations that deal with an altered title of a wiki page.

[17] An internal global identifier for a Siemens employee.

```
10            "date": "2010-04-14T14:16:44",
11        },
12        {
13            "resourceUrl": "...",
14            "resourceTitle": "...",
15            "tag": "...",
16            "user": "...",
17            "date": "...",
18        }
19    ]
20 }
```

Listing 6.2: *Tag Export example*

The tag exchange format has been implemented for the three major social tagging platforms and some smaller social software applications used inside Siemens.

6.3. Data Aggregation

One major difficulty in the design of the architecture is the aggregation and especially the synchronization of tagging data. In general, between two mechanisms can be distinguished: “push” and “pull”. Push means that an application calls an API function to inform STAGS that a change (create/ update/ delete) in its tagging data happened. “Pull” stands for a periodic fetch mechanism. STAGS triggers an update on its tagging data for a certain taggable application. Additionally it can be discerned whether there is a partial update (last changes since a certain time stamp) or full update (the complete data bases).

Several combinations of push vs. pull and partial vs. complete updates are possible. Table 6.1 gives an overview of the combinations. Depending on the capabilities of the taggable applications a selection has to be made. For typical applications the most practicable solution is to make a full update once a day. For the current (in production used) instance this update mechanism is triggered at 2 a.m. GMT every day. This has turned out to be a good point of time for the context of Siemens.

The component for fetching the social tagging data is called *Tag Importer* in figure 6.1. Because for some use cases such as providing a stream containing the newest activities for the social software applications inside Siemens, a daily update mechanism is insufficient. Hence, an aggregator for RSS 2.0 feeds has been implemented. Reducing the social applications inside Siemens to a lowest common denominator lead to a RSS 2.0 format with one extension of having an extra XML tag containing the GID of a Siemens employee – see listing 6.3. This job enables STAGS to have more up to date data for the cost of not having the same semantic granularity of the export format described before. The exact tag assignment occurrence is not reflected in RSS 2.0. A resource has assigned tags (category XML tag), an author (creator), a title, a date of creation, an URL and other fields. Who assigned what tags at what point of time is missing. But since more complex update mechanism were not possible for all applications, the RSS 2.0 feeds have been used to update the social tagging

		1x per day	1x per hour	Event driven
Push	*Complete*	not always possible – depending on the application	not applicable for large data sets and small server	not always possible – depending on the application
	Partial	best solution when there are very few events – not always possible	best solution when there are few events – not always possible	best solution when there are many events – not always possible
Pull	*Complete*	very pragmatic solution – easy to implement	depending on the server load not desired	not possible
	Partial	once a day is too infrequent	depending on the activity on the application even more frequent requests are thinkable	not possible

Table 6.1: *Different social tagging data aggregation methods and their implications.*

data during the day. All computations on the social tagging data, such as co-occurrences of tags, are computed without the RSS 2.0 feed updates.

```
<?xml version="1.0"?> <rss version="2.0">
<channel> <!-- title, link, language, pubDate and so on ... -->
<item>
   <title>Why I still love Java</title>
   <link>http://blogs.siemens.com/story/walter.christian.
      kammergruber/123456</link>
   <description>Although everybody is complaining about
   the verbosity of Java it is IMHO an excellent language
   for implementing major and maintainable applications.
   With eclipse you effectively write less code than you would
   in any scripting language having...
   </description>
   <pubDate>Wed, 15 Sep 2010 09:39:21 GMT</pubDate>
   <author>
      <name>Walter Christian Kammergruber</name>
      <siemens:authorGID>ABCDE123</siemens:authorGID>
   </author>
</item>
</channel></rss>
```

Listing 6.3: *RSS 2.0 Example Excerpt of a RSS 2.0 Feed with Author Extension*

6.4. Implementation of General Use Cases

The following sections describe the implementation of the described uses cases from chapter 4. While describing the theoretical concepts behind the implementation is an important issue, the actual practical implications that result from the implementation might be of bigger interest and therefore the focus relies more on the practical side.

Figure 6.2 depicts the dimensions where relations between tags can come from. This maps to two of the three approaches described in chapter 5: Data mining and statistically algorithm, and tag thesaurus editor. Mapping to an already existing structured source does not involve users as a facet. Hence, it is left out in the figure.

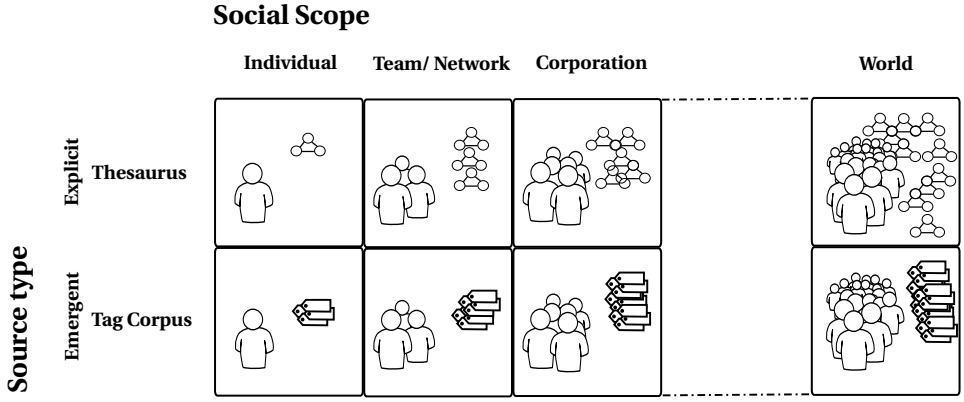


Figure 6.2: *Tag Relation Distinction: Basically two dimension of sources for relations between tags can be distinguished. On the one hand the social scope, where tags are filtered according to the group of users. On the other hand the source type. This means from where the relations were aggregated – either thesaurus relations (defined by an user) or statistically computed ones.*

In the vertical axis a distinction of the source types is made. There are basically two cases where a relation can come from. On the one hand explicit relations are defined by using the thesaurus editor. On the other hand there are implicit relations that are derived from the tagging behavior of individual users or groups of users. In the horizontal axis one can see the social scope of relations between tags. The spectrum begins at the left with the individual user and ends at the right with potentially every user worldwide that is known to the system. The distinction of social scope enables algorithm to consider different levels of personalization. For example: The target is to provide tag suggestions for a user creating a blog post in a personal weblog. Then, the individual tagging practice might be of higher relevance than the tagging practice of all users, across all tagging applications. Whereas if someone is trying to find entities relevant for a certain topic across information spaces, a more broad expansion might be a better choice.

The general use cases described in chapter 4 have been implemented by providing widgets and a REST-like interface. Widgets can typically be integrated into an existing web application without much effort. This is very important where an application owner does

not have the required resources to implement a more complex integration via the REST-like interface. This is a non-functional requirement being a crucial part for the adaption in a real world scenario. Definition 6.1 provides an elaborated description of a widget.

Definition 6.1 (Widget) *Widgets are stripped-down Web applications that are highly portable. They make it easy for nontechnical users to add dynamic content or functionality, such as search tools or maps, to different locations, such as Web pages, portals, desktops, and mobile devices.*
— from [You08]

Examples for widgets in the Internet are the integration of Google Maps into web pages or omnipresent social widgets, such as the Facebook like button. All general use cases can also be implemented via the REST-like API by the respective application owner. Nevertheless, widgets are the method of choice for a simple integration of STAGS.

6.4.1. Tag Suggestions during Tag Assignments

Figure 6.3 shows three screen-shots containing several stages of the tag assignment process that is supported by STAGS. The *auto-complete*[18] UI design pattern has been chosen as best practice for this widget component. As stated on the referenced web page provided by Yahoo!: "*Providing an auto-complete feature for a standard text box field enables the user to be faster and more accurate in presence of ambiguous or hard to remember information from large datasets when the total number of items exceed the usable length of a drop down box. Auto-complete allows faster user input by removing ambiguity about expected input data, avoiding potential mis-typed information as well as narrowing down the correct choices.*" The auto-complete UI design pattern is popular and very present along nearly every major web site.

For the case of STAGS at most seven tags are suggested, which is a recommended number of suggestions in UI. The limit seven – psychologists called it the "magical number seven" when they discovered it in the 1950s – is the typical capacity of what is simplified the brain's working memory [Mil56].

Figure 6.3 shows several states of a tag assignment:

- *(a) First two letters are entered* – `kn`: Tags starting with those letters are suggested. The rank for a tag is according to the recency and frequency the current user and all users of all systems have used it. In this example, different spelling variants are suggested, such as `knowledge_management` and `knowledgemanagement`. In that case the ranking suggests the most popular variant for the current user and the global tag usage at a higher position. The tags used by the current user get a higher rank – a user does not want to be patronized. Less popular variants of a tag are not contained in the list because of their lower rank. Tag friction is decreased.

- *(b) A complete tag has been entered* – `knowledge`: According to that tag related tags are suggested. If more than one tag has already been entered tags for all these tags are shown.

[18] `http://developer.yahoo.com/ypatterns/selection/autocomplete.html`

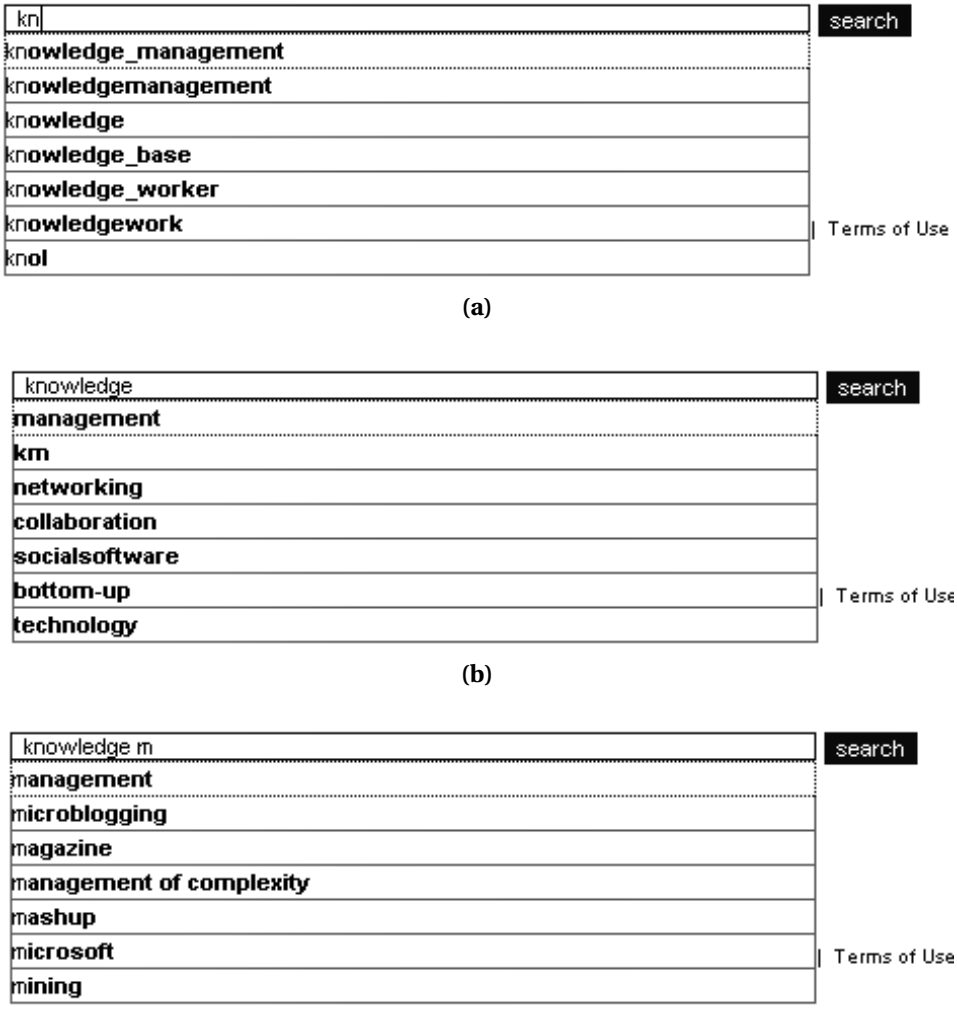


Figure 6.3: *Tag Suggestions*

• *(c) A complete tag and some letters of a new tag have been entered* – `knowledge m`: According to the first tag and the first letter entered for a new tag, tags are suggested. In the example, tags that are related to `knowledge` and start with the letter `m`.

A case where no letters and no complete tags are entered is also possible but not shown in figure 6.3. This reflects the state when the input field is empty. Then, simply a list of tags ranked by a combination of frequency and recency is displayed. In most systems where the widget currently is integrated, this feature is deactivated in order to prevent an irritation on the user side. Most of the time without a certain context, these suggested tags can be irrelevant for the current entity. A version where tag suggestions are created based on the text corpus of an information object has been implemented in a separate project, but is out of scope for this thesis.

Having different sources where suggested tags can originate from, a simple ranking strategy has been developed. Assumed that there are n sources $S_1, \ldots, S_n$: for a given state of the input box, a list of potential tag suggestions is determined by combining the lists of computed tag suggestions for each source S. Each list for each source is ordered by the importance of a tag suggestion and truncated according to a certain number of tags. Therefore, a tag in a list for a source S_i gets a value $rank_i$ that is inverse to its position in the list. A combined rank for a tag_i can be computed via a simple linear scaled weight function:

$$W_{total}(tag_i) = \sum_{k=1}^{n} (a_k \times rank_k(tag_i) + b_k)$$

a_k is a weight that expresses the general importance of a source. b_k is a correction value.

Rank	S_1	S_2	S_3	S_4
1	contest	ceo	cloud	content
2	cc	ct	competition	computing
3	ct_ic	client	ceo	client
4	communication	corporate	chain	communications
5	clearcase	contract	client	contract
6	cloud computing	creativity	communication	cms
7	customer	cc	contract	cable
8	community	communities	compliance	certification
9	cement	change	corporate	community
10	CCTV	ct_ic	creativity	corporate

Table 6.2: *Example tag suggestions for four source $S_1, \ldots S_4$.*

A simple *example*:

Supposed there are four lists of tag suggestions based on different sources S_1, S_2, S_3, S_4 that have to be merged into one ordered list which can be delivered as tag suggestion for an user. Table 6.2 contains suggestions computed based on an user input for four different sources. In this example is not important for which cases these suggestions are computed.

rank	tag	value
1	ceo	54.7
2	client	54.7
3	cc	48.7
4	communication	45.7
5	contest	44.7
6	contract	36.7
7	cloud	35.7

Table 6.3: *Example tag suggestions ranking for the best seven tags.*

Having tag suggestion that start with a "c" is the result for the cases (a) and (c). In (b) there are related tags that might start with another letter included.

Let the parameters be $a_1 = 5$, $a_2 = 3$, $a_3 = 4$, $a_4 = 2$, $b_1 = -0.4$, $b_2 = 0.2$, $b_3 = -0.2$, $b_4 = 0.1$. Then for example the value for the tag "cloud" is computed as follow:

$$(5 * 0 + -0.4) + (3 * 0 + 0.2) + (4 * 9 + -0.2) + (2 * 0 + 0.1) = 35.7$$

The value for the tag "contest" is computed as follow:

$$(5 * 9 + -0.4) + (3 * 0 + 0.2) + (4 * 0 + -0.2) + (2 * 0 + 0.1) = 44.7$$

Table 6.3 contains the seven top most tag suggestions that were computed based on the parameters above and the resulting suggestions displayed in table 6.2.

Determining values for a_i and b_i is not a simple task. There is no gold standard for defining a "good" tag suggestion. This can only be achieved through an user study. For the actual implementation in the deployed instance of STAGS a supposed to be good combination of these values has been selected via tests where several combinations for a_i and b_i were evaluated by expert users. Additionally, selecting the "right" sources based on which tag suggestions are computed depends strongly on the data available. In some systems there might be not that many thesaurus relations available then co-occurrence can be play a more important role. In another system the opposite might be the case.

```
<html>
<head>
    <script type="text/javascript" src="http://stags.siemens.com/
       api/v1.0/js/stags_ui.js"></script>
    <link rel="stylesheet" type="text/css" href="http://stags.
       siemens.com/api/v1.0/js/stags_ui.css"></link>
    <script type="text/javascript">
        function tagsSelected(tagNames) {
            alert(tagNames);
        }
        function initStags() {
            new stags.TagSuggestionTextField(
                        document.getElementById("ts_text_field"),
                    {onTagsSelected: tagsSelected }
            );
        }
    </script>
</head>
<body onload="initStags()">
    <div id="ts_text_field"></div>
</body>
</html>
```

Listing 6.4: *Tag Suggestion widget example*

Listing 6.4 contains an example how a tag suggestion widget can be embedded into a web page by using some lines of JavaScript code. A single global variable (JavaScript) is used for accessing the stags widget API. This is modeled after the popular module pattern – see [Rau14, chapter 31].[19]

6.4.2. Information Navigation

Relations between tags can be used for navigating an information space. This is helpful for people who are familiar with the content of a social tagging application as well as for people who simply want to gain an insight on its topics.

Figure 6.4 shows a screenshot displaying a navigational user interface. The right column contains filtering criteria for the source of the content[20] and as most important part a tree user interface component. This example displays the fourteen most frequently used tags for the current user – from “km” to “internet”. The node “web2.0” is expanded with its child nodes from “wiki” to “web2.0_applications”. For avoiding complexity, at most nine child tags are shown. The used color scheme for the related tags is the same as the one used for the thesaurus editor – see figure 6.7. Black tags are relations that come from co-occurrence

[19] also online `http://speakingjs.com/es5/ch31.html`

[20] In this screenshot only resources from the major social software applications inside the Siemens intranet are contained: TechnoWeb, Blogosphere, and Wikisphere.

analysis which are included if there are less than nine thesaurus relations present for the corresponding tag.

By clicking on a tag, a list of resources is returned in the left column. In this example the results 1-9 of 185 for the tag wiki are shown. The result set can be navigated through a user pagination interface design pattern[21]. An user can further restrict the result set by selecting other tags from the right column.

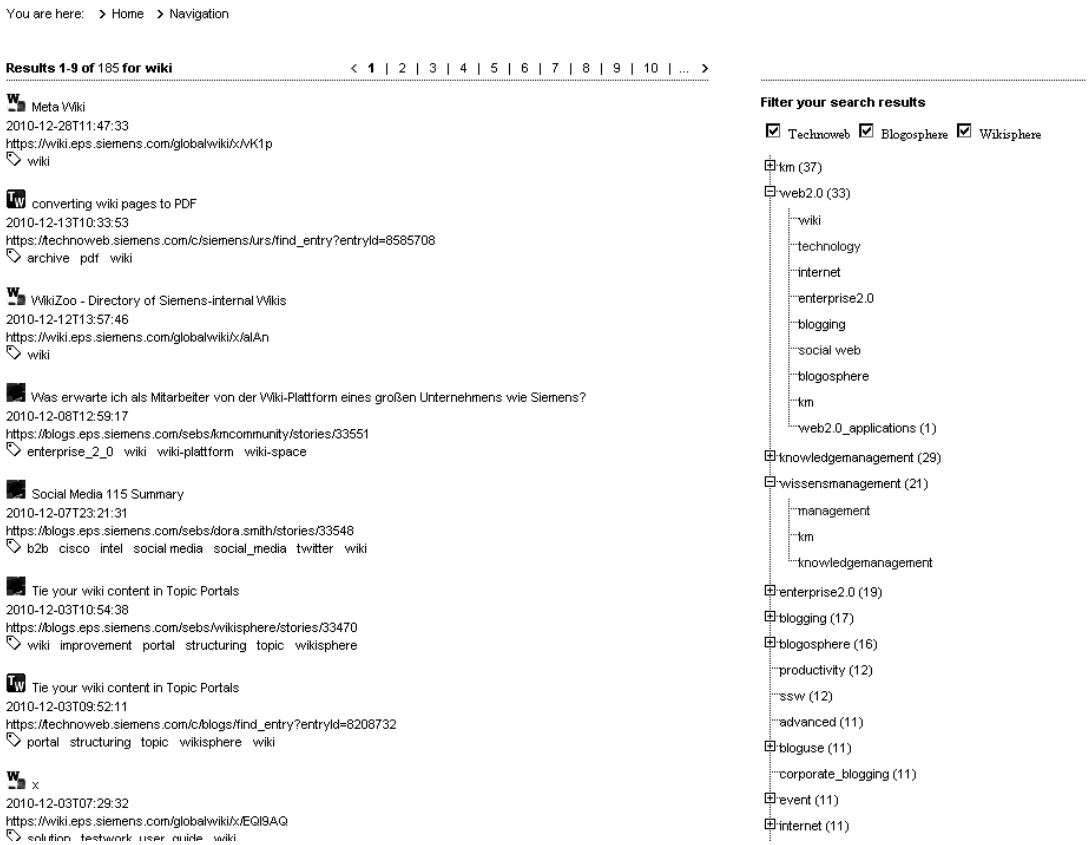


Figure 6.4: *Navigation user interface.*

The purpose of this user interface is to explore or navigate a collection of resources. In contrast to the search approach where a user knows exactly what he or she is looking for a navigational interface can also be used for discovering unknown content. Serendipity effects can occur when a user finds a piece of information that is useful, but that has not been the one that he or she was looking for in the first place.

6.4.3. Search

Search is a crucial part of every application that deals with a large amount of content. This is typically the case for every major site on the Internet. In the case of content that has been

[21] See `http://developer.yahoo.com/ypatterns/navigation/pagination/`

organized via social tagging, a mechanism for dealing with classical information retrieval problems (see chapter 1) is of significant importance – even when the number of resources does not reach the ones that can be observed in the Internet.

Figure 6.5 shows a screenshot for a search interface. On top one can see a typical input slot where an user can enter tags to search for. This is supported by a tag suggestion feature analogues to the one shown in figure 6.3.

Below the search input field, the current filter that is set to the tag “internet” is displayed. This filter can be removed by clicking on the red x. The filtered tag is highlighted in the result list in the left column. Via a pagination interface the result set can be navigated.

In the right column on the top a filter for restricting the sources of the displayed resources is provided – corresponding to the one shown in figure 6.4. Below that tags related to the currently selected tag “internet” are listed. These tags are grouped into several categories. “Your own relations” includes relations defined by the current user. “Everybodys relations” contains related tags without a restriction to a certain user. This view in the screenshot is collapsed. Below that the twenty most important co-tags are shown. On the bottom suggestions originating from a mapping of tags to DMOZ are contained. The mapping to DMOZ is error prone (see chapter 5, but in some cases it can lead to useful input.

When the user clicks on a tag in the right column the tag is added to the filter and the result set is updated. The left column containing the suggested tags is adopted accordingly.

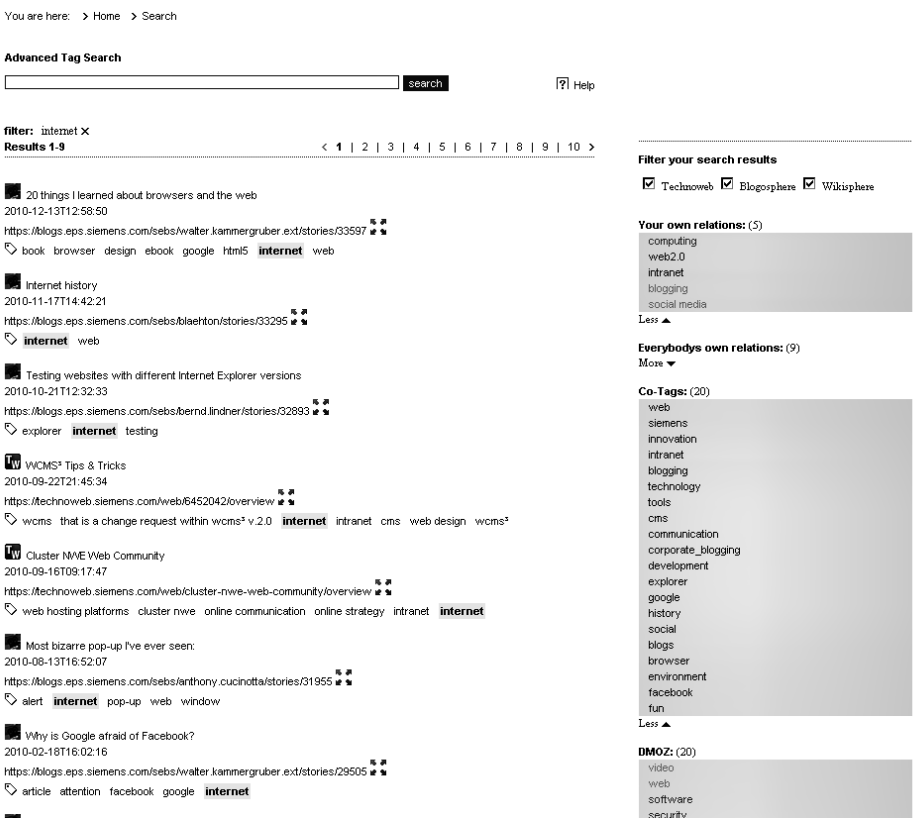


Figure 6.5: *Search with suggestions.*

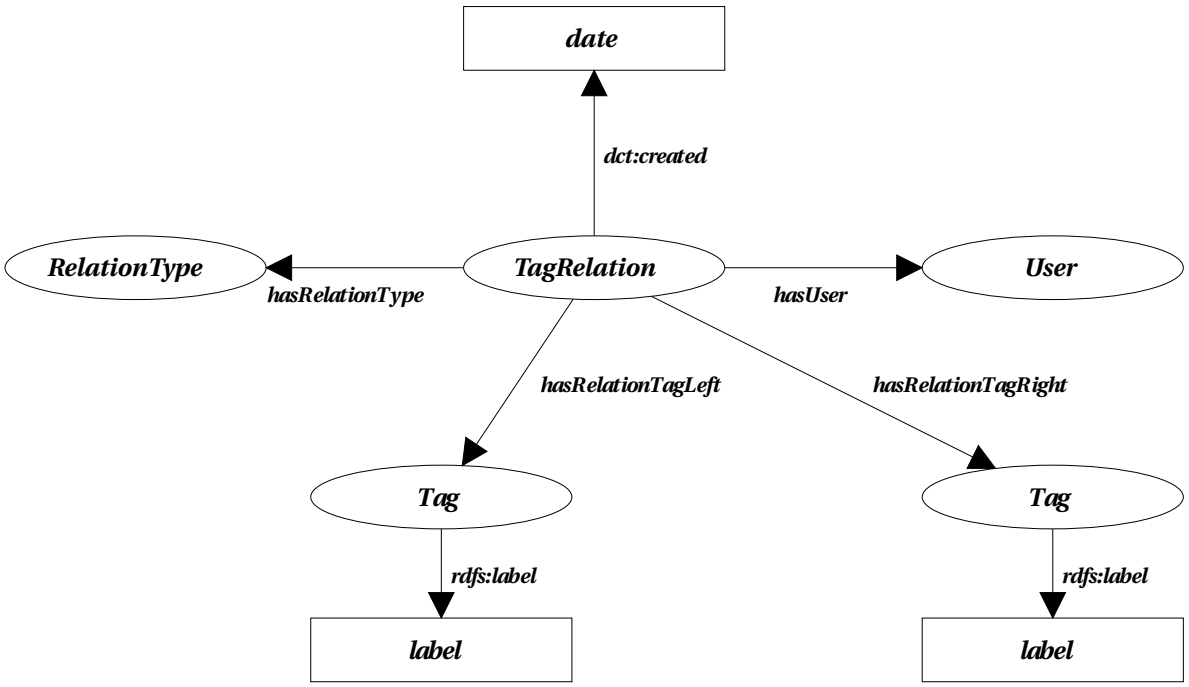


Figure 6.6: *Tag Relation Model: It is specified that a user defines a relation between two tags.*

6.5. Thesaurus Editor

The thesaurus editor described in chapter 5 has been implemented with a clean web interface. Since the dimensions of the user interface[22] for the thesaurus editor correspond to the dimensions of an application page, a loosely integration via a widget is a bad choice. Technically, it is possible to integrate the thesaurus through an IFrame or a similar mechanism, but overall user experience is likely to be negatively affected. Hence the thesaurus editor is preferable used as application for itself or relations are defined by using the RESTlike APi.

The thesaurus editor has a straight forward and meant to be intuitive user interface. An user can define relations per drag and drop. These relations are stored via AJAX in the background. As JavaScript library for developing the client side, Dojo[23] has been used. The relations are stored in the RDF store – see figure 6.1.

6.5.1. Data Model

In definition 5.5 in chapter 5, a tag thesaurus relation has been proposed. In this section an implementation via RDF is described.

Fig. 6.6 shows the underlying data model. It follows the popular N-array Relations Design Pattern[24]. Sometimes these kind of artificial classes are called reified relations. For

[22] The size it takes on the screen.
[23] `http://dojotoolkit.org/`
[24] `http://www.w3.org/TR/swbp-n-aryRelations/`

this case the TagRelation class is used to express a multinary relation between users having stated that two tags are associated by some kind of relation (synonym, broader, narrower, related or ignored) at some point in time. The relations are stored inside the *RDF DB* via the *RDF Wraper* (see figure 6.1). When no namespace is given a default namespace `https://stags.siemens.com/vocab` is assumed.

Additionally, the well-established vocabulary of Dublin Core[25] is reused for storing the time the user has created the relation between to tags (`dct:created`). Tags are fully qualified resources and therefore get their own URI as identifier[26]. The original tag as a string is linked to the tag URI by `rdfs:label`, a standard RDF Schema property[27].

6.5.2. User Interface Components

Fig. 6.7 shows a screenshot of the web interface. A user can define relations between tags via drag and drop. In the example the tag "knowledgemanagement" is selected (2). Selecting a tag can be done by double clicking a tag in any tag boxes. There is a simple filter mechanism for searching for tags (1). A user starts typing and the resulting tags are displayed according to entered letters. The resulting tags can be set as current tag or dragged into the relation boxes (3). The relation boxes are used as drop zones. A user can define a relation between the current tag and another tag by simply dragging the tag in to the desired box. The type of boxes match our chosen thesaurus relations enumerated above. If the user has already defined relations between the current tag and other tags, they are filled in to the relation boxes accordingly.

Utilizing the two main algorithmic approaches to find relations between tags, computed relations between the current tag and other tags are displayed in several suggestion boxes (4). In the first box tags that have a low string distance to the current tag are displayed. For the example "knowledgemanagement" the results are not optimal but the string distance can be used to find synonyms with spelling variants or singular/ plural. "event" and "events" or "web_2.0" and "web2.0" can be listed as examples. Another box contains the co-occurrence matches (in the example "km," "wissensmanagement" and further). In the two, far right-hand boxes suggestions generated by mapping a tag to terms in external structured input are displayed.

[25] `http://dublincore.org/documents/dc-rdf/`
[26] `http://www.w3.org/TR/webarch/#uri-benefits`
[27] `http://www.w3.org/TR/rdf-schema/`

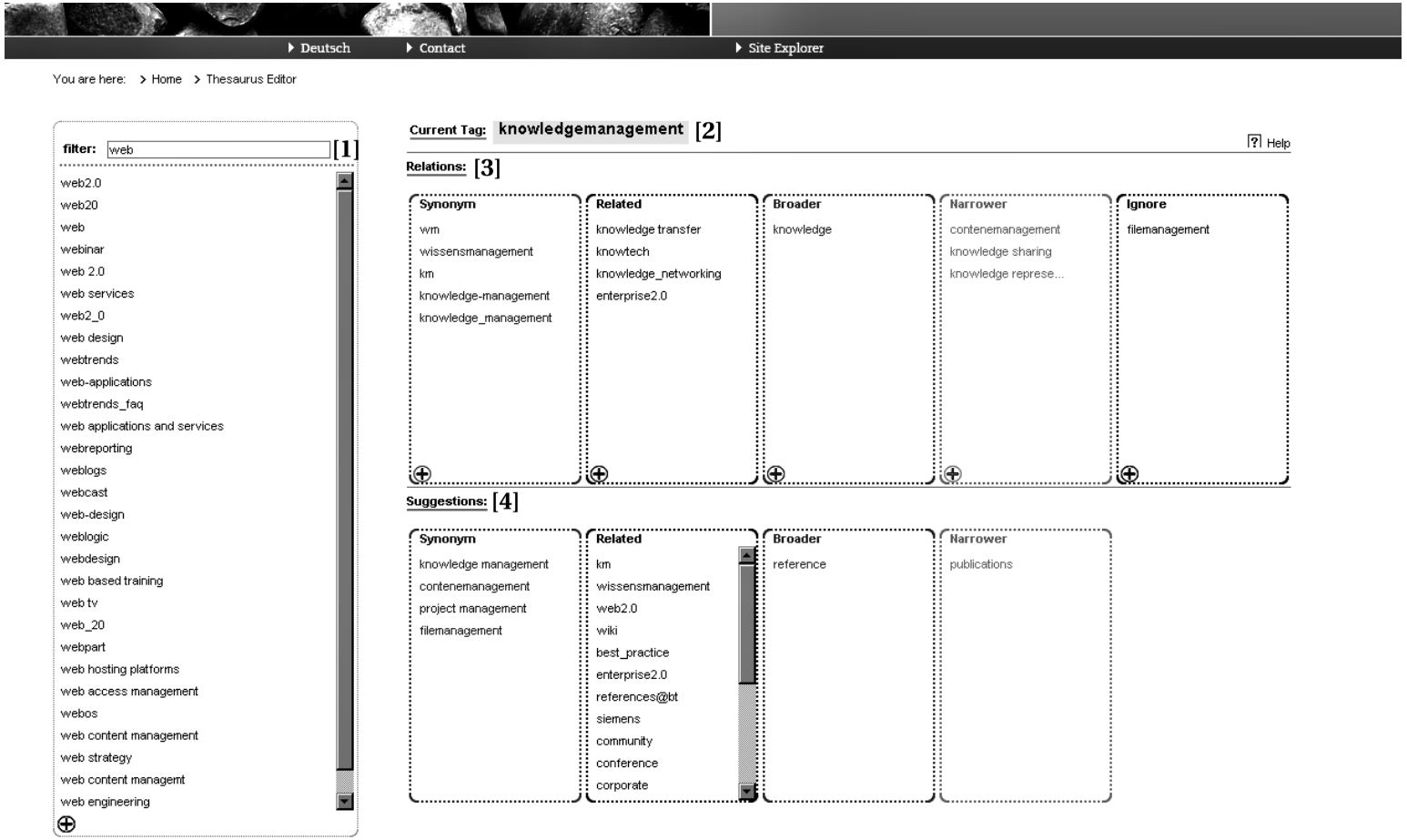


Figure 6.7: *Tag Thesaurus Editor example: An user can define relations via drag and drop to a relation box (synonym, narrower, broader, related, ignore (3)) between a selected tag (in this example "knowledgemanagement" (2)) and another tag. The already defined tag relations are displayed in the boxes. For the current tag, the user gets suggestions for possible related tags (4). Additionally, an user can search the folksonomy by applying a simple filter mechanism (1).*

6.6. Mapping between Requirement and Architecture Solutions

The architectural design has been developed based on the requirements that have been derived from the use case described in chapter 4. In table 4.5 the requirements are listed. Table 6.4 contains the mapping of the requirements to the decisions made for the implementation.

ID	Requirement Name	Architecture Solution and Notes
1	Cross application	The data aggregation has been implemented for various applications (sec. 6.3).
2	Personalization	The implementation of general and thesaurus use cases consider the individual user (sec. 6.4, 6.5).
3	Easy and loose integration	STAGS can be called via REST or can be embedded via widgets.
4	Type of tagging data agnostic algorithms	Implemented algorithms work on data exchange format. No text mining or other form of entity type specific algorithm is used.
5	Simple tagging data export format	A simple and generic data exchange format (sec. 6.3) has been defined.
6	Service Level Requirements (SLR)	Through Caching and usage of NoSQL databases a maintainable application has been developed.
7	Cross Site Integration	With JSONP and server to server communication via REST problems with same origin policies have been avoided.
8	Reasonable Response Time	The design has shown in productive environment to fulfill adequate response times.
9	Scalability	The implementation can be distributed among a cluster of servers when needed.

Table 6.4: *Mapping between requirements and solutions – for use cases see table 4.5.*

It is shown that the actual implementation matches the identified requirements. A version of STAGS is in use in a productive environment since November 2010. The implemented widgets are embedded in several internal platforms.

6.7. Evaluation and Experiences within Siemens

After the implementation of a prototype for the tagging framework, parts of the prototype have been transferred into an application (STAGS) that is used in production. Shown in this chapter is the validity of the architectural design and the usefulness of the service itself. The algorithms developed for analyzing social tagging data have been evaluated in chapter 5, where they were introduced and motivated. This approach is comparable to best practices in software testing. There first unit tests for small software modules are made and afterwards system testing is conducted. This process ensures that the overall systems (and especially the composition of the individual modules in a whole working environment) works as desired. The comparison with software testing is naturally limited. For example, there is no equivalent to integration testing and the system testing is typically made via a staging platform. On the other hand the analogy emphasizes the different granularities involved in the assessment of a design.

The evaluation of the implemented system follows the framework of the Updated DeLone and McLean information system (IS) success model (D&M model) [DM03]. This model is an extension and revised version of the original D&M model published in 1992 [DM92]. The original model from 1992 was developed after DeLone and McLean had reviewed existing work on the success of IS system. They tried to formulate a general and comprehensive definition of IS success. It covered different perspectives on the evaluation of information systems. They classified the found measure into six major categories. This lead to a multidimensional measuring model with different success categories that have dependences between each other. The updated model is the result of many attempts of other researchers to improve and extend the original model. In 2003 DeLone and McLean published their revised model. At the core the D&M model evaluates the effectiveness/ success of an IS with regard of its usage by stakeholders. This includes stakeholders' satisfaction with the system concerning interactions as well as the quality of the returned information.

6.7.1. Updated DeLone and McLean Information System Success Model

The D&M is designed as a general framework that has to be adapted to the specific context. Not each criterion defined by the D&M model is relevant for assessing the success of the implementation of the tagging framework. Other measurements are not testable with limited time and resources or other constraints, such as information security or privacy policies.

Figure 6.8 provides a graphical overview of the D&M model. The D&M comprises six theoretical dimensions:

- *System Quality:* It measures the IS itself. This includes more technical characteristics of an information system. Typical aspects are: Availability, reliability, adaptability, or response time.

- *Information Quality:* The content and content representation is the key issue for this characteristic. For an e-commerce site this includes personalization aspects (for example

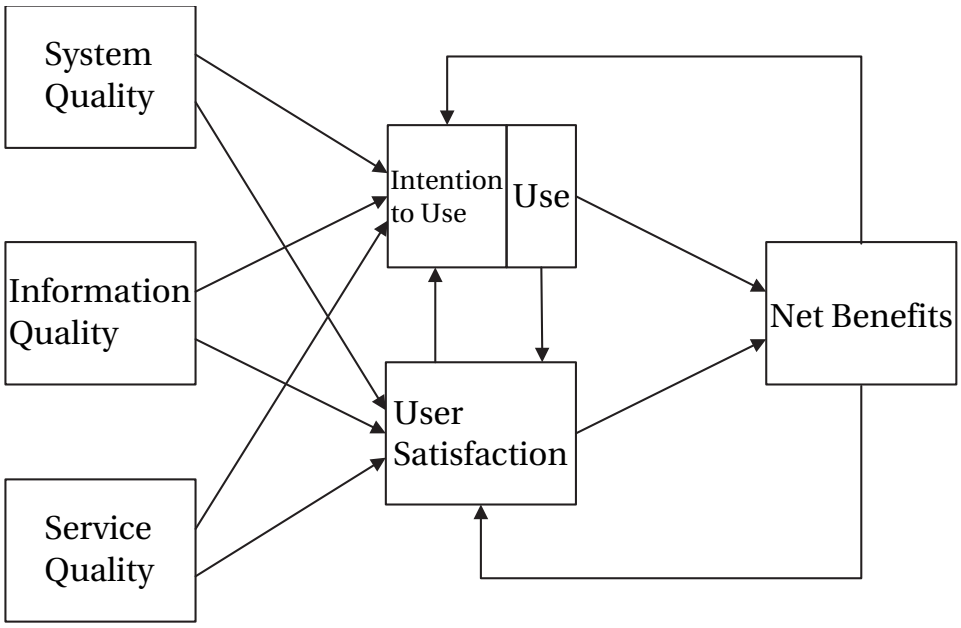


Figure 6.8: *Updated DeLone and McLean IS success model [DM03].*

recommendations what other items can be of interest for the current user), the relevancy of the presented information (an example search engines, where relevant content should be returned[28])

• *Service Quality:* This includes the support aspects. If problems or questions occur, how are these issues handled. Bad customer support leads to a loss of customers.

• *Intention to Use / Use:* The actual usage concerning the quality and quantity is important here. It is of interest how frequent a service is used and how users interact with the system.

• *User satisfaction:* This covers the overall user experience. Is the usability and the performance OK? Is the user satisfied with the outcomes of his or her interactions with the systems?

• *Net Benefits:* This is more some kind of a "feedback loop". Positive and negative outcomes are included. Does the usage have an impact on the overall efficiency of an organization or a general market?

As the arrows in figure 6.8 indicate the individual attributes are not independent. A slow performance (Service Quality) for a system has obviously a negative impact on the user satisfaction.[29] Voluntary used system (with alternatives) with low performance will see less persons employ it (Use).

[28] See chapter 1 for precision as popular measurement.
[29] As mentioned in chapter 4 performance can have a big impact on the usability of a system – see [Nie93]

The D&M model is not to be seen as the optimal and final framework for assessing information systems in regard of completeness and representativeness. Nevertheless, it provides useful indicators to chose from depending on the respective information system. Not all measures from the D&M model are of interest for the evaluation of STAGS. Some aspects are with limited resources and time not measurable – as already stated before. Other aspects are very hard to grasp or even to define. Applicable methods for evaluating what can be evaluated have to be selected. The assessment of different criteria defined in the D&M have been conducted with two different approaches. *First* by analyzing log files (see section 6.7.3) numbers on the actual usage of STAGS can be estimated. This gives insights on aspects that can be described in numbers. For example: "how many users had STAGS per day" or "what methods of STAGS were most frequently used". The log files can provide quantitative indicators. Qualitative Aspects of a system, such as usability aspects, can hardly be derived from log Files.

Hence, in a *second* step a questionnaire with people that are affiliated with applications using STAGS has been conducted (see section 6.7.4). Quality aspects of an information system cannot be assessed isolated from perception of individual users. Expert interviews is a method of choice to find out what non quantitative characteristics an information system has. This two-folded approach for the assessment of the resulting system ensures a coherent impression of the provided solution.

6.7.2. Applications inside Siemens Using STAGS

Currently, four different applications inside the Siemens intranet make use of STAGS as a service in production. Social Tagging Data is collected from five applications. Of the three major platforms described earlier (see chapter 5) TechnoWeb makes heavy use of STAGS. For Blogosphere and Wikisphere currently only proof of concept implementations exist. The data of these two major social software tools are aggregated from the live systems, though. There are plans for using STAGS inside Wikisphere and Blogosphere. They are simply not implemented at the time of this research efforts.

Further, there are other platforms using STAGS that have not been mentioned before:

- References+ [MS11]: A micro blogging service introduced in 2005. As of 2010 it had about 500 authors that have created around 2,600 posts. Currently the service has grown and is widely used inside the Intranet of Siemens Building Technologies.

- Communities for Competence (C4C): A collaboration platform around a Confluence wiki with SharePoint integration.

- Intranet Community Hub: An overview page of the activities that occur in Siemens social software applications. It is a sub page of the Siemens intranet portal. In this application there are no social tagging data generated.

Social tagging data of References+ and C4C are collected from the live systems. The Intranet Community Hub does not produce any social tagging data. Its purpose is to show the aggregated activity stream of all social software applications. This includes all mentioned social software applications. Each of them provides an API for the JSON export format.

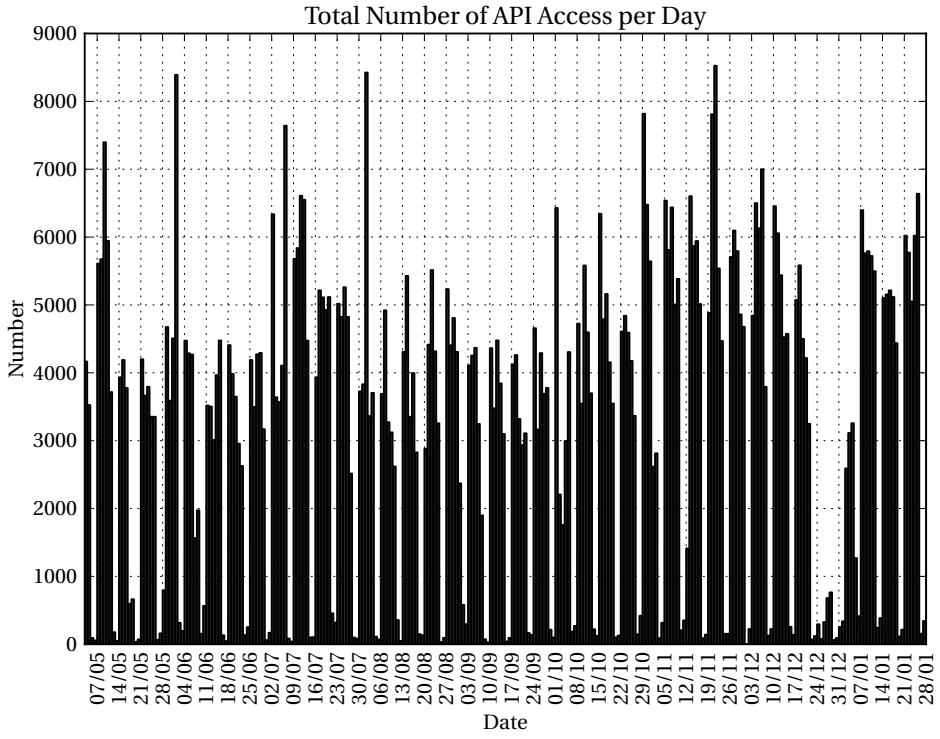


Figure 6.9: *Number of aggregated API access per day. Each HTTP request to the API is counted.*

6.7.3. Usage Statistics

In this section the empirical usage data of STAGS inside Siemens are investigated. The evaluation data has been extracted from log files. Each HTTP access is stored in the log files. The Log files from the third of May in 2012 to the twenty-eighth of January in 2013 have been analyzed. The data has been collected in anonymized form in order to respect privacy policies and to comply with laws concerning personally identifiable information (PIS). This relates especially to IP addresses. There were 57,186 different users (IPs) in total that have accessed STAGS in the nine months the log files cover. Only the actual API request returning data values, such as data serialized to JSON format, are considered. Other requests, such as requests returning HTML or CSS files, are excluded. Each single HTTP request has been logged and counted individually.

Figure 6.9 displays the number of HTTP request to the STAGS API. Each bar represents the total number of HTTP request to the STAGS API per day. Table 6.5 contains the statistical characteristics for HTTP requests to the STAGS API. In a) the API request for all days are collected. In b) only the values for work days[30] are considered. Typically, there are peaks at the beginning of the month. During Christmas holidays and New Year's Eve apparently there were less people working and therefore the numbers are lower. On average for work days, there were about 4000 requests to the STAGS API. If weekends are included, the num-

[30] Monday to Friday. Special Holidays are not excluded.

Characteristic	Value
Minimum	28
Maximum	8,424
Mean	2,981
Median	3,525
Standard deviation	2,121
Variance	4,497,108

(a)

Characteristic	Value
Minimum	596
Maximum	8,424
Mean	4,120
Median	4,108
Standard deviation	1,338
Variance	1,790,873

(b)

Table 6.5: *Statistics for all HTTP requests to the STAGS API: a) for all days, b) for work days (Monday to Friday)*

Characteristic	Value
Minimum	2
Maximum	1,538
Mean	389.3
Median	435
Standard deviation	298.43
Variance	89,059

(a)

Characteristic	Value
Minimum	16
Maximum	1,538
Mean	535.57
Median	481
Standard deviation	224.45
Variance	50,376

(b)

Table 6.6: *Statistics of unique users (IPs) for the STAGS API access : a) for all days, b) for work days (Monday to Friday)*

bers are lower.

Figure 6.10 shows the distribution of the API request to the respective applications. Note: the legend maps from left to right and from top to bottom for the chart. The application that makes most use of STAGS is TechnoWeb. References+ has become more popular over time. Hence, it has increased its percentage of the API access for the last months. Other applications seem to have not too much fluctuation in its access rates.

Figure 6.11 depicts the number of individual users per day. Table 6.6 contains the corresponding statistical measure (a) for all days, b) only work days). For work days there were on average about 500 unique users that had contact with the STAGS API. There is a peak at first of November were there was an article in a Siemens company magazine about TechnoWeb. Hence, there was an above average overall access to TechnoWeb and STAGS correspondingly.

Figure 6.12 shows a pie chart segmented according to the parts of the analysis modules (TagAnalysis, ResourceAnalysis, SystemAnalysis, and TagAssignmentAnalysis – see figure 6.1). Each analysis module is mapped to an URL part, such as `api/v1.2/json/tag/frequent` for the most frequent tags or

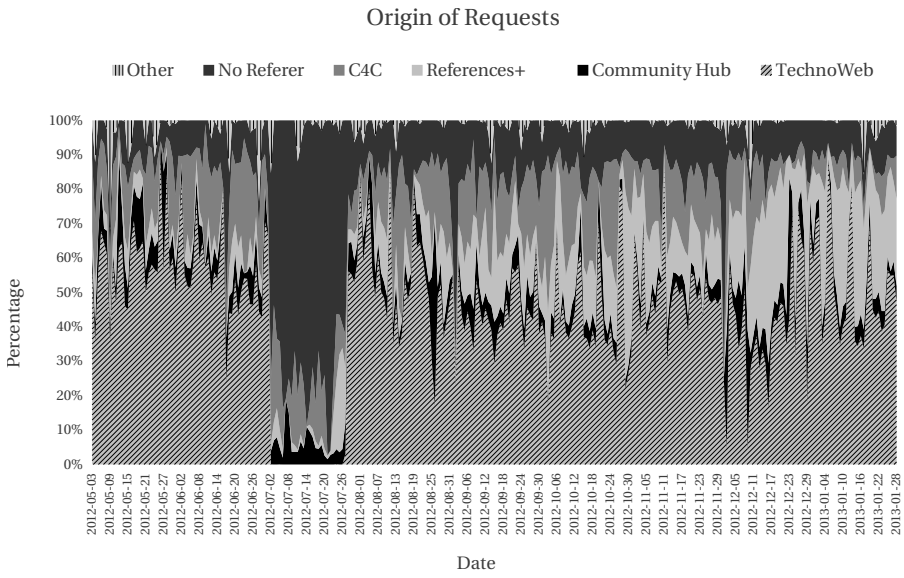


Figure 6.10: *Number of aggregated API access per day and origin. Each HTTP request to the API is counted. The referrer of the HTTP request is analyzed. Unfortunately in July, most of the HTTP referrers where logged as "-". The exact reason is unknown. This might have to do with a configuration policy for TechnoWeb that was tested in that period.*

`api/v1.2/json/user/findExperts` for retrieving people that are experts[31] for a certain topic. Each API methods has various parameters that are ignored in this aggregation. "Network" is a part of the API that has been introduced to deal with TechnoWeb specific requirements. It does not belong to the overall conceptual design of the tagging framework, but it is included in the chart because of its overall importance for the system used in production. Its basic functionality lies in recommending interesting networks for users according to their tags[32].

Each part of the analysis module is exposed as REST API. Methods regarding tagging systems, tag assignments (as a whole) are not totally utilized in production and some are only for debugging. Hence, they do not show up in the chart. Tag clouds and related methods are most widely used and this module has therefore the lion's share of the requests. The Resource API is used for displaying lists of tagged information items. In most scenarios, where the resource API is queried, also a tag cloud is used for filtering. Thus the views for filtering resources also contain a request to the tag module part. "User" refers to the module that is used for channeling Urgent Requests (see section 5.4.4). Its main purpose is to determine relevant users for a question that has been asked in TechnoWeb.

[31] Expert is here a general expression.
[32] In order to keep some compactness of this thesis, the details of the implementation are excluded.

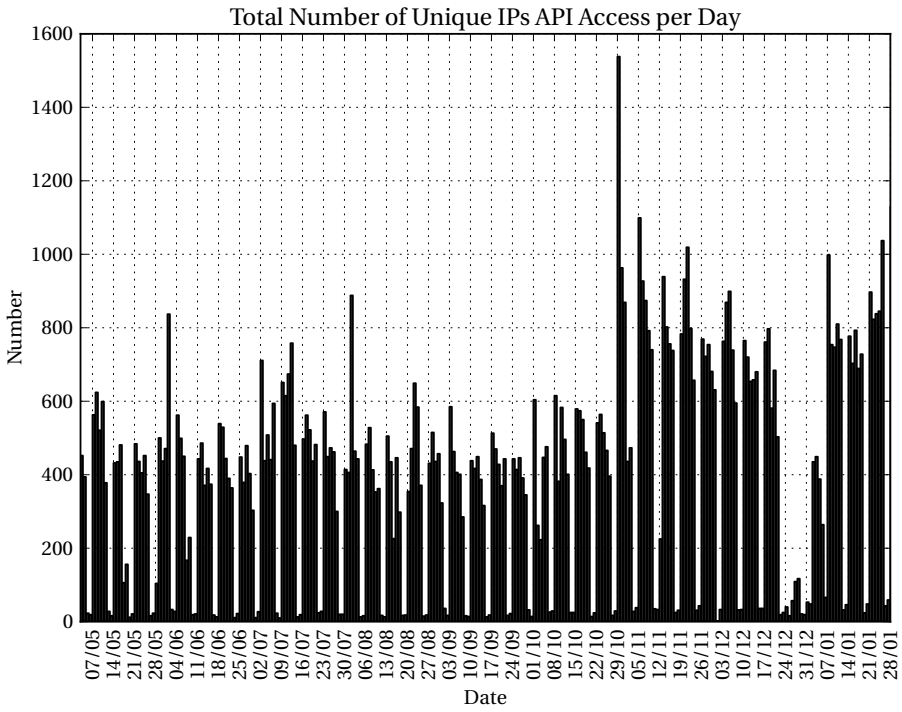


Figure 6.11: *Number of unique users (identified by their IPs) that have accessed the API per day. Each HTTP request to the API is counted.*

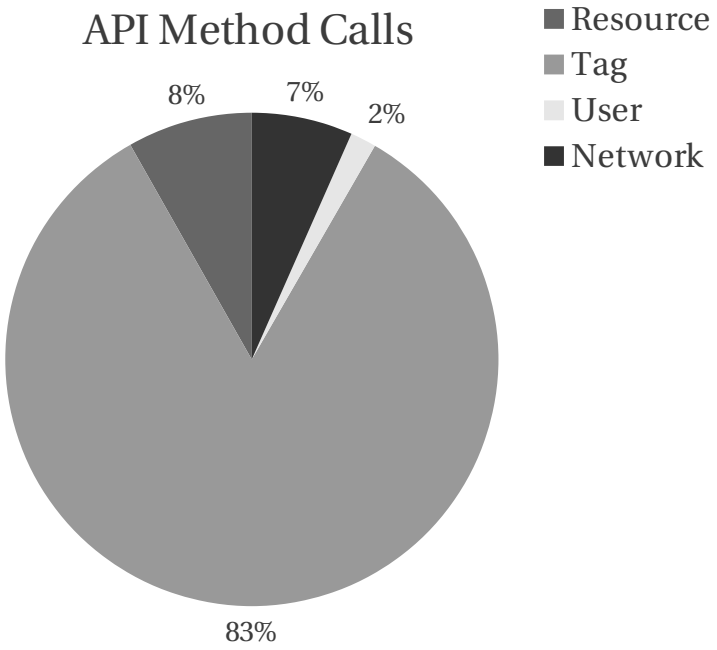


Figure 6.12: *Number of API request for each method type.*

6.7.4. Application Owner and Expert User Interviews

This section includes the summarized results of the interviews for various application owners and expert users. Each of the interviewed persons is a German native speaker. Hence, the interviews were conducted in German. For each application that is described in section 6.7.2, a qualified person was asked to participate in a questionnaire. This questionnaire aimed at getting an impression of the experiences that the people representing each application from an administrative – as well as user – a perspective which they made with STAGS. When individuals are asked there is always a problem with personal opinions that might differ from an objective investigation.

Five different persons were asked and there is an overall agreement that STAGS is a success for the respective use case. Hence, it is very unlikely that this is just a coincidence. The conclusions of the interviews seem to be indeed very valid. The provided actual numbers reflect more an intuition of the individual interviewees. For example, numbers such as availability are hard to guess without further investigation. The Rule of thumbs are only known to people with a background in a certain field. For example an availability of 90 % means a downtime of 36.5 days per year and 99 % means a downtime of 3.65 days per year. The real availability rate was in nearly all interviews underestimated. It was about 99 % on average for the two years since STAGS has been used in production with beginning of November 2010.

The interviews are divided into several parts reflecting the D&M model. At first the interviewee is asked about his personal background. This is to find out how reliable the answers to the individual questions are. For example, a person with no programming experience has limited insights on how easy a service can be embedded into an existing application. Even if the person gets feedback from a programmer, some tasks are not that easy to asses. On the other hand a programmer without the perspective of a product manager might not have the insights on the overall impact of an application.

In a second part questions that test different measurement characteristics defined in the D&M model are asked. For each characteristic, questions have been chosen that reflect its specific demands. The types of questions are adapted to the time constraints. It is not realistic to get in-depth answers for each aspect. The service quality characteristic is not applicable for STAGS. The context of STAGS is not comparable to for example with an Internet service. There are no customer inquiries that have to be dealt with in a scalable manner. Requests for changes and bug reports are treated via agile methods when they occur. Hence, the service quality aspect from a customer point of view is not relevant in this context. System quality and information quality are the most significant aspects that have to be addressed in the questions. For the cases where the actual usage and user satisfaction cannot be derived from the log files additional questions have been asked. Each interviewee is also an end user. The following sections summarize the findings of the interviews.

Blogosphere And Community Hub in Global Intranet Portal

The interview has been conducted on November 29th 2012 at 1.00 - 1.40 pm with Dr. K. E., Senior Researcher and Project Manager at Siemens Corporate Technologies. Dr. K. E. is

a project manager for the Community Hub and responsible for the Siemens Blogosphere. He uses applications where STAGS is included on a daily basis. He has experience in programming with older languages such as C and scripting languages. He had contact with the REST API in production and in prototypical implementations. From proof of concept point of view Dr. K. E. had also experience with provided tag suggestions and tag clouds. STAGS has been included in the Community Hub that is a part of the Global Intranet Portal of Siemens. For this use case the REST API of STAGS is queried for resources that are filtered on certain constraints. The resources are listed as activity stream in the Community Hub.

The efforts necessary to integrate STAGS with the application has been in the order of hours. Challenges for the integration were in the area of typical problems with software installations in an enterprise context, such as issues with HTTPS certificates. The actual integration was simple. STAGS' API is easy to understand as far as he is concerned. For his use case the API was flexible enough and his requirements were fulfilled. Widgets he had dealt with could be adapted via CSS adequately.

STAGS' estimated availability has been around 95 % or maybe even higher and hence sufficient. The returned results of STAGS are complete and relevant. The API returns the expected values. The performance (in the sense of reaction times) is not an issue for STAGS – even with increasing numbers of information items.

An end user is assumed to have no problems regarding the interaction with STAGS' UI elements – as far as he or she does not have issues with standard Intranet applications. STAGS is very useful in that matter that cross application services are provided. Having a mechanism to integrate various platforms is crucial. The potential of STAGS is not released completely – only about 20 %. Especially capabilities regarding semantics are yet to integrate.

The benefit of STAGS lay in the aggregated activity streams of the social software applications. STAGS achieved this benefit and the benefits were worth the effort. A real business impact is not measurable. STAGS has been used when the Community Hub was introduced from the beginning. Hence, a comparison of before and after is not possible.

Alternative solutions would have been in the adoption of interfaces provided by the Global Intranet search. This variant has been put aside because of the complexity of the offered interfaces and potential problems that is inherent with the nature of a search engine. Typically, results of a search engine are based on a full-text search. There are meta-data fields for tags available. The definition of these fields is not stringent and each application handles its meta-data differently. Therefore, the expected result of querying the Intranet search engine is very likely to produce less reliable and relevant results. STAGS with its REST API and the provided widgets seem to be a better approach for use cases such as his. In his role as project manager, Dr. K. E. gained insights in the prototypical implementation the thesaurus editor. He is convinced of the usefulness of this interface and that it is a needed component. Other features such as finding experts are known to him and he thinks those are exciting and innovative.

Community 4 Competences

The interview has been conducted on December 5th, 2012 at 2.00 - 2.35 pm with Florian Kuba, a Senior Developer at Siemens Industry Division (see appendix A.2). Mr. F. K. is a project manager and requirements engineer for the Community 4 Competence (C4C) platform. Being a computer scientist, he has a strong technical background in software development. He had contact with STAGS in the role of an end user. The REST API is known to him from a conceptual view and he wrote a small tool that queries the REST API.

C4C as platform consist of two different tools: a Microsoft SharePoint and an Atlassian Confluence. STAGS is used to build a bridge between the social tagging activities that occur in these tools. For C4C the integration of other systems, such as TechnoWeb or Wikisphere, is not important and to some point not desired. C4C uses the tag suggestion service of STAGS limited to tags that were used in C4C. This restriction has been made to ensure a consistent tagging practice in C4C independent from the tagging conventions in other systems. Additionally, in Confluence a view for filtering resource based on tags is included – also exclusively for resources from C4C.

The efforts for C4C to integrate STAGS were in the area of a week including documentation and tests. Since Wikisphere is also a Confluence installation the plugin for the tag export could be reused. Hence, the major challenge was in aggregating the social tagging data of SharePoint and Confluence into a single export format[33].

In general, the integration of STAGS into C4C was easy. Only the SharePoint integration needed some not straight-forward tricks. User interface elements of SharePoint are more demanding to adapt than for example the ones in Confluence.

For F. K., the API is easy to understand and use. To the aspect of flexibility, he cannot give a qualified answer since the functionality of the API was for him narrowed down to some given points. He also cannot say something of the ways the widgets can be styled, because the adaption of the widgets was provided. The results of the skinning of the widgets were acceptable for him.

The availability of STAGS is good and good enough for him. He guessed in the area of good 90 %. The reliability of STAGS is very good in his opinion. The relevance and the completeness of the returned results were sufficient.

For him, the UI elements are easy to use. He cannot make a general assessment, because older and less experienced persons might have another opinion. STAGS is useful in that sense that with tag suggestion support misspelled tags are avoided. Tagging an artifact is made easier and faster and hence, there are efficiency improvements for this use case. Cleaning up the tag base is supported by a tool that makes use of the STAGS API. There are some efficiency improvements noticeable. The actual amount of time that is saved is hard to estimate. The expected benefits of using STAGS were: an improved user experience, aggregation of the tags of the two applications, and saving time when cleaning the tag base. STAGS met these expectations.

The invested efforts were in a good relation to the benefits. An actual business impact is hard to quantify. Users may find needed information faster, but there is no feedback

[33] Note: This was a design decision made. It would also have been no problem for STAGS to collect the social data for a system via several exports

about that. From a service provide point of view cost savings were made in the area of tag cleaning. Numbers for this case are not easy to make explicitly as well. The popularity of the platform could not be increased in a measurable way. F. K. thinks that the introduced features are important and useful, but he cannot make any statements about the quantity without usage statistics. An alternative to using STAGS was to implement a proprietary solution for C4C. He doubts that this would have happened. In his general opinion there is a need to use small focused services instead of big monolithic applications. He remarks this as an aspect of STAGS.

He sees room for improvements in the area of administrative tools. There some more features for automated tests are needed. Another improvements is the need for a multi-client capability[34]. Additionally, he thinks that some form of recommendations may be useful, such as "recommend resources". Concluding he makes some other suggestions on how to improve STAGS, such as instead of making a tag export on a daily basis, an event based mechanism could be introduced.

Wikisphere And Landing Page Wikisphere in Global Intranet Portal

The interview has been conducted on December 6th, 2012 at 1.00 - 1.45 pm with B. L., a senior software developer and project manager at Siemens Corporate Technologies (see appendix A.3). Mr. B. L. is a solution designer for the landing page of the Wikisphere in the Global Intranet. He is responsible for the Siemens Wikisphere where he functions as project manager as well as developer for plugins. He is an experienced software developer (about twelve years). He is not a typical end user, but has some hands-on experience with STAGS. He is more familiar with STAGS from a technical point of view. He has implemented a Confluence Plugin for the social tagging data export[35].

He had contact with STAGS' REST API as project manager that explains the API to the agency that implements the solution. In the landing page the tag cloud widget has been used. The integration of the widget took about two days. This specific widget has been easy to integrate. Other widgets might be a little more challenging to deal with. In general, the API is undemanding to understand. Only simple HTML, CSS and some JavaScript knowledge is a requirement. Widgets are flexible to handle and can be extended – as far as he can tell.

STAGS availability has been around 95 %. For Mr. B. L. this availability is not sufficient. He expects a higher availability from a corporate service. He mentions that the down-times of STAGS are partly related to the hosting and not the application. The performance of STAGS is more than sufficient. The reliability is sufficient and the returned results have met the expectations on the API.

STAGS is useful for the narrow use case of the landing page, because it creates a more dynamic view on the content of the Wikisphere. As an end user B. L. mentions the TechnoWeb use case where STAGS is used for browsing resources. In that use case he is definitely con-

[34] Note: STAGS has been designed with the target to foster knowledge sharing. Some form of rights management system that enforces restrictions on the visibility of resource has therefore explicitly been not implemented.

[35] This is not literally said in the interview, but a known fact to the author.

vinced of the usefulness of STAGS. He describes the implementation of STAGS as state of the art. The productivity of the individual user is very likely increased. Some results might also have been achieved with a decent full-text search engine.

Benefits of STAGS lie in the aggregation of social tagging data from various social software applications. For him, as being responsible for the Wikisphere, this has the advantage that content from the Wikisphere can be displayed in other applications. This leads to more visibility of the platform and leads to more traffic and participation. This has been shown via statistical analysis of the traffic on his platform. Content listed on other platforms is more frequently visited. The invested efforts have therefore been justified. STAGS has a business impact for him in that manner that the target of the Wikisphere is to share information. With STAGS this mission is supported. An impact on other aspects, such as usability is not applicable for his area. Alternative solutions for STAGS were not available. A main issue was the cross application data aggregation and the access to this data via a public API. He sees the public API as a disadvantage of APIs with restricted access. Closed data sources cannot be integrated into STAGS. This might be important for some use cases.

TechnoWeb

This interview has been conducted on December 5th, 2012 at 11.00 - 11.35 am with Mr. T. M., a Senior Project Manager at Siemens Corporate Technologies (see appendix A.4). Mr. T. M. is a project and community manager for TechnoWeb. He has a background as a software developer using mainly Java and C++. Being an end user he can also provide insights on the quality of user interactions with STAGS. He has made some experiences with the STAGS REST API, but more from conceptual view then of the perspective of a developer. In TechnoWeb STAGS is used for Urgent Request Channeling, diverse Tag-Clouds, Tag-Suggestions based on the content of an Urgent Request and general Tag-Suggestions without the need of a text. Additionally filtered views on resources are merged with the results of an internal search. In TechnoWeb STAGS is embedded in various places: user profile pages, search views and whenever a resource can be tagged. Additionally STAGS is used in many other places. The costs to integrate STAGS into TechnoWeb have been small. He estimates the efforts in the area of hours for each used feature – summed for all features in the area of days.

He thinks that STAGS is easy to integrate into an existing application. Understanding the API is simple with the help of some common sense. The API follows usual conventions. Therefore, after someone has understood the concepts of the API it is easy to employ. In general the API is very flexible. Some use cases have a very narrow focus and hence the flexibility of these special API elements is stripped down to basic needs. This is not considered as a bad characteristic, though. The provided widgets are very easily to adapt to the style of an application.

STAGS' availability is around 99 %. In his opinion it would be desirable if it were at 99,9 %. Overall the availability is sufficient for TechnoWeb. Only when there are notifications for an Urgent Request to distribute a downtime of STAGS leads to essential problems.

STAGS' performance is excellent and therefore meets the requirements of TechnoWeb completely. The reliability of STAGS is sufficient. The returned results of the API in general

comply with the expectations. There might be some small bugs, but this is hard to check. With the exception of the tag suggestions for a textual content, the results of the API are very good and reasonable. For the tag-suggestions, there might be some way to manually manage the used tag corpus. If, for example, a typo in a tag is frequently made, this misspelled variant of tag also gets suggested. As far as he can tell, the problem lies not really within the algorithm but is more related to the data sources. For an algorithm that is based on frequencies, it is hard to say if a tag is good in the regard of content. There should be some way to make corrections manually. Except these problems the developed algorithms lead to sufficiently good results.

The provided user interface elements are easy to understand and use. From an end user perspective STAGS is useful. Learn effects for a user emerge when there is a good visualization of the contents of a platform. With decent tag visualizations a user can see that tags are beneficial to get information on certain topics – especially a user can stay informed on those topics.

Mr. T. M. cannot say too much about expected benefits of using STAGS before he became a project manager for TechnoWeb. One important feature that has been introduced since he joined the TechnoWeb team, is the Urgent Request Channeling. With the usage of this mechanism a big number of notifications could be saved. This helps significantly with scaling TechnoWeb's notification mechanism for an increasing user base. Hence the expected benefits were observable. The invested costs were justified be the achieved results.

For Siemens there exists certainly some business impact. For TechnoWeb there is no actual measurable business impact. The costs for maintaining the TechnoWeb platform stay the same. Only improvements in the general service and the community management tasks can be observed. Because of less SPAM emails (the sent emails are reduced by the Urgent Request Channeling algorithm) sent by TechnoWeb, the overall user experience with the platform has improved. Additionally, improvements in the user interactions with the TechnoWeb system, such as tag suggestions for a given content, lead to a better usability.

As an alternative solution he mentions that it would have been possible to implement an own solution for TechnoWeb. But then, data from other platforms would have been excluded. STAGS is a cross-platform service that offers synergy effects. A solution specifically tailored for TechnoWeb would have lacked these effects. STAGS being developed inside Siemens can be additionally adapted to the new use cases when they emerge.

References+

This interview has been conducted on November 30th, 2012 at 10.00-10.40 am with Dr. J. M., Senior Manager Knowledge Management at Siemens Building Technologies (see appendix A.5). Dr. J. M. is responsible for References+ as project and community manager as well as developer. He has a software programming background in C, C++, PHP and ASP, the technology References+ is implemented in.

In his daily work he uses the micro-blogging functionality of References+. Hence, he is also an end user and can assess usability aspects of STAGS. Main features used in References+ are Tag-Clouds and Tag-Suggestions. Tag-Cloud widgets are embedded at various locations with diverse configurations.

In References+ there are no direct queries to the REST API of STAGS. Only the provided widgets are used. These widgets were easily adapted to the various design contexts they occur. It took about a week to understand the concepts behind STAGS and to integrate the widgets into the diverse views they are used in. With a complete documentation, efforts might have been less. The documentation has been improved in this process. With the updated documentation using the STAGS' widgets seems to be easier to use. After someone knows the conventions used in STAGS the API is easy to use. The used widgets have many filter options that enables all customizations needed. With CSS the widgets can be easily skinned.

Dr. J. M. estimates that STAGS' availability is around 99,9 %. Therefore his requirements are totally fulfilled. The performance is excellent. In his daily usage he never observed any errors in the displayed data. Hence he guesses that the reliability is around 100 %.

The Tag-Cloud widget contains an effect, when while the mouse hovers a tag, co-tags to this tag are highlighted. This might confuse some users at first. He thinks that this might be an issue, but there were no complaints from users. In general, the widgets are intuitively to interact with.

STAGS is useful in the sense that with the Tag-Cloud overviews of topics in certain contexts are provided. Tag-Suggestions help reduce spelling mistakes and lead to a consistent tagging practice. Effects on the productivity are directly observable. Only through the consistent tag usage the effectiveness might be increased. Learning effects for users may occur in the sense that they might get improved overviews on the content of a system. The navigation of the content is enhanced.

Expected benefits were in an improved navigation and interlinkage of content and a better consistency in the tagging practice. These benefits were noticeable. An explicit business impact for References+ could not be observed. The popularity of the platform has been most likely been improved by the features of STAGS. Especially, the usability was enhanced.

Alternative solutions were not evaluated. First tests with STAGS have shown that it had useful features to offer. The main positive aspects of STAGS are that it can be easily integrated into an (web) application and very flexibly customized.

6.7.5. Summary of Evaluation

The six dimensions of the D&M model are: *System Quality*, *Information Quality*, *Service Quality*, *Intention to Use / Use*, *User satisfaction* and *Net Benefits*. Except service quality,[36] all criteria were evaluated in either the form of interview questions or in the form of quantitative numbers extracted from log files.

• *System Quality:* According to the conducted interviews, the system has sufficient performance, availability, reliability, and adaptability. Only Mr. B. L. remarked that the availability of the system is not adequate for his use case.

[36] This aspect is not that important for the evaluation of STAGS in the scope of this work. When (and if) STAGS is distributed as a product outside Siemens the service quality cannot be left out.

• *Information Quality:* Every interviewee stated that he was satisfied with the results the STAGS API returned. There is room for improvements, such as in the area of tag suggestions or test APIs. In general, the results meet the needs of the individual applications.

• *Intention to Use / Use:* That the system is used on a daily basis has been shown by the evaluation of the log files. More than fifty thousand persons have had a form of interaction with STAGS in one way or another.

• *User satisfaction:* According to the interviews usability improvements lead to a higher user satisfaction. If tasks are more efficient to perform with the help of a tool support the user satisfaction is evidently increased.

• *Net Benefits:* There has been no clear answer to this questions. There might be improvements in the way information items are found and shared. A general major impact is hard to grasp.

Overall one can say the implementation of the tagging framework has fulfilled the expectations and requirements. There is certainly room for improvements. However, the first findings have shown its value.

CHAPTER 7

Conclusions and Prospects

He who chooses the beginning of a road chooses the place it leads to. It is the means that determine the end.

— Harry Emerson Fosdick (1878 – 1969)

Contents

This chapter provides an overview of the contributions described in this thesis. Additionally it contains an outline of future work.

7.1. Summary of Contributions

In chapter 1 following research issues have been defined: *semantic challenge, hidden structure challenge,* and *orchestration challenge.* Table 7.1 contains a summary of the contributions to the individual identified challenges. These challenges have been addressed from a bird's eye view by the design and implementation of a tagging framework. Requirements

Challenge	Contribution
semantic challenge	Co-Occurrence Analysis (section 5.4.1)
semantic challenge	Association Rule Mining (section 5.4.2)
semantic challenge	Mapping of External Structured Sources (section 5.6)
semantic challenge	Semi-Automated Approach: Tag Thesaurus Editor (section 5.7)
hidden structure challenge	Discovering Communities of Interest (section 5.4.3)
hidden structure challenge	Urgent Request Channeling (section 5.4.4)
orchestration challenge	STAGS: Implementation of a Social Tagging Framework (chapter 6)

Table 7.1: *Contributions for solving identified challenges.*

needed for the implementation of a social tagging service have been formulated – based on several core use cases in which such as system can made use of.

Several approaches have been developed for enriching social tagging data with lightweight semantics. These approaches have been incorporated into a social tagging service. Relations between tags have been derived through three different kinds of approaches: (i) Data Mining and Statistical, (ii) Mapping of external structured sources, and (iii) Semi-Automated Approach: Tag Thesaurus Editor. These three approaches regard plain semantic relations. Additionally, weak hierarchical relations in form of tag bundles have been derived.

Beyond the semantic relations, other types of structures have been derived from social tagging data. A method for discovering communities of interest has been developed. An approach for message distribution mainly based on social tagging data has been created. Furthermore an algorithm for suggesting tags with a full text as input and in relation to an existing folksonomy has been introduced.

Various modules of the prototypical software has been transferred into a production ready state and gone live in November 2010. They are used within Siemens inside several social media applications and intranet pages. By analyzing log file and interviews with people that are affiliated with applications using STAGS the validity of the developed system has been shown. In the nine months the log files cover there were 57,186 different users in total that have accessed STAGS. For work days there were on average about 500 unique users that had contact with the STAGS API. On average for work days there were about 4000 HTTP requests to the STAGS API in total.

The message distribution algorithm has saved several *millions* of emails since it has been deployed. For TechnoWeb, the new distribution algorithm has no negative impact on the number of answers per asked question. Significantly fewer email notification lead to positive effects in the user acceptance of the platform. Additionally, each omitted email saves valuable time of an employee.

7.2. Potential Problems with the Chosen Overall Approach

In a perfect world every problem has an ideal solution. Unfortunately the world we live in differs in that point. The chosen approach in this thesis has shown its validity, but there are still some weak points to mention.

• As with every algorithm that is based on user generated content, there is a cold start problem. This means that if a system is newly introduced there is no data (created by users) available. Hence, if an algorithm relies on statistical, or similar, methods to create an output, for "empty" systems they fail. There is the need of some substitutional approach, such as returning random values or newest items. This "cold start" problem is observable for the majority of algorithms in chapter 5. If introduced into a newly created setup, they will deliver suboptimal results.

• Additionally to cold start problem, the algorithms in chapter 5 depend on the overall quality of the social tagging data. If the data input is messy (for example only few, general tags are used) the result of the algorithms tend to be less useful.

• In chapter 6 a daily bulk update approach has been selected to export the social tagging data. This method does not scale well for huge installations. For the case of the typical Intranet setups this is no problem. If it becomes one, then some kind of push mechanism, where an application notifies the tagging framework for changes (create, delete), has to be introduced. For sake of simplicity the bulk update has turned out to be a good choice. Also from a non-functional requirement – ease of implementation – the bulk update solution is a good choice.

Most of the evaluation methods seem to provide too less evidence from a strict scientific method[1]. This is due to the nature of the subject. Social tagging data typically have a heterogeneous character. Some scientific methods, suchs as a deductive approach, is not applicable here. From a practical perspective this is not a real issue. The experiences with the adoption of the developed framework have been proven its validity. In the "real world", in some cases a pragmatic approach turns out to be a prudent choice.

7.3. Future Work and Research Directions

The following sections contains suggestions in which the described implementation of the social tagging service can be improved. Then other usage scenarios are described and research questions not addressed by this thesis are explained.

[1] Discussions about falsifiability in the sense of Popper are not subject of this thesis – for some details see [Pop02]. Falsifiability is also a too strict requirement for most of the scientific fields.

7.3.1. Implementation Improvements

For performance reasons the used RDF store should be removed from the system. The implementation should solely rely on MongoDB and Ehcache. The described data mining approach for deriving tag bundles via association rule mining and clustering users by their tag usage are currently done offline. This means there is no real integration into the productive system. Re-computation of results has to be done manually. Integration into the running system will result into a decisive added value. Additionally alternative clustering and association rule mining approaches should be evaluated. One major problem is that association rule mining does not scale very well for big data sources. Increasing scalability for very large data sets can be achieved with map reduce [DG04] and horizontal scaling. Furthermore, an integration of thesaurus structures into the algorithms is desirable.

7.3.2. Applications of the Social Tagging System

The thesaurus editor can be extracted as standalone application and used in other contexts. For example, it can be adopted to create a concept based thesaurus that complements a commercial thesaurus used within an enterprise search engine. Social tagging data can improve dictionaries inside software spell checkers. Frequent tags not contained in a dictionary might be specialized terms used within a company or community. Utilizing thesaurus relations in the context of information channeling of information items, is an interesting application in the area of internal communication. Amongst other relations types, synonyms can be used for combining tagged resources from different sources. For example “sustainability” and “ethical_consumerism” can be treated as the same term. News tagged with either one of them can be merged into the same channel.

7.3.3. Tag Bundle Applications

In section 5.4.2, it has been shown that it is possible to compute tag bundles out of social tagging data by applying a popular association rule mining algorithm. For some users with a certain tagging gusto it was also not possible to determine tag bundles. Other association rule mining algorithms [HGN00] might deliver better results or results with other characteristics. There are plans to incorporate bundles into user interface design elements for STAGS. With the help of user feedback the bundling of tags can be investigated and improved.

Further Bundles can function as starting point for piled user interfaces – see [MSW92]. New forms of user interface metaphors may lead to new forms of addressing the problem of information overload as well as information scattering. Especially in the context of mobile devices with touch screen this can lead to major improvement in the overall user experience.

7.3.4. Social Tagging Data as Glue for Communities

It remains to be seen if a juxtaposition of communities of interest with social networks presents an expected congruence. By combining social network analysis with tagging communities, one is expected to be able to shape and aid the emergence of communities also exhibiting a high degree of centrality. For achieving real-time analysis of the data a number of issues remain to be addressed. Firstly, the runtime properties must be enhanced to a degree permitting on-line analysis. Secondly, the inclusion of social networking information (already part of many on-line information sharing platforms) offers a next step in the analysis. Having thesaurus relations, further improvements of the algorithm can very likely be achieved. By using synonym relations tags, can be grouped together in a pre-processing step.

7.3.5. Thesaurus Editor Usage Patterns

The actual usage patterns of the thesaurus editor have not been investigated in this thesis. Hence related to the semi-automatic approach following questions can be of interest:

1. What is the participation pattern of users in the constructions of the tag thesauri?

2. How do semi-automatically generated tag structures relate to automatically generated ones?

3. How useful do users perceive services that are mainly and directly based on a tag thesaurus?

4. Are the results of functions that model heterogeneity or homogeneity of competencies based on path lengths in the structure of these thesauri (as part of services) congruent with our expectations?

5. Is there a fixed point to which the evolution of the tag structure converges?

The first question can be evaluated by observing the generated structures and the quantity of unstructured tags left as “orphans” in the systems. For example, how many topic tags used in blog and forum entries have not been categorized in one of the existing topic tags? For the second question, it can be discussed for a narrowed down use case in the area of open innovation[2]. Finding people with the right competencies is an important part in the team creation process. The results of the thesaurus editor approach used in the context of competencies can be compared with the results of competence ontologies extracted from job advertisements [ZMH09] and the differences can be analyzed. The research question on homogeneity and heterogeneity, as well as the research question on the usefulness of services, can be addressed through interviews with lead users about the quality of the service results. The last research question can be investigated through systematic sampling and analyzing the structural dynamics of the thesaurus.

[2]For the concepts behind open innovation see [CVW06].

7.3.6. Information Distribution

As already stated in chapter 2 with the advent of the digital age information overload or flooding is a serious problem. As Rutherford Rogers is quoted – unsourced: "*We're drowning in information and starving for knowledge*". From a user perspective finding relevant information has to be fast, focused, and simple. Distribution of items relevant and personalized for the individual user is an important but likewise demanding task.

Using social tagging data can help with channeling information items. Filters based on the personal profile derived from tags related to the individual user can be a first step in this direction. Especially, in the enterprise context where no excessive user profiles, such as the ones Google and Facebook generate, are available.

Activity streams can be generated out of the generated user interest profile providing access to potentially relevant pieces of information and easing the access to information for an individual user. Prototypes for this approach have been developed, but have not been evaluated and refined yet.

Identification of experts in a company and improving collaboration amongst employees is a challenging knowledge management task. Sometimes the phrase "If we only knew, what we know" comes up in discussions on that topic.

List of Tables

List of Figures

Bibliography

[ABB+00] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet*, 25(1):25–29, May 2000.

[Ack06] M. Ackerlauer, H. Heiss. Breeding technologies within expert networks as a balanced technology management method. *WSEAS TRANSACTIONS ON BUSINESS AND ECONOMICS*, pages 245–252, 2006.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, New York, NY, USA, 1993. ACM Press.

[AKDG07] Hend S. Al-Khalifa, Hugh C. Davis, and Lester Gilbert. Creating structure from disorder: Using folksonomies to create semantic metadata. In *the 3rd International Conference on Web Information Systems and Technologies (WEBIST)*, 2007.

[ASSM07] S. Angeletou, M. Sabou, L. Specia, and E. Motta. Bridging the gap between folksonomies and the semantic web: An experience report. In *Workshop: Bridging the Gap between Semantic Web and Web*, volume 2, 2007.

[BD03] Bernd Bruegge and Allen H. Dutoit. *Object-Oriented Software Engineering: Using UML, Patterns and Java 2/E: International Edition.* Prentice Hall International, 2. a. international edition. edition, 10 2003.

[BHRS07] Matthias Baumgart, M. Yaser Houri, Thomas Rückstieß, and Frank Sehnke. Research methods in informatics and its applications: Design-oriented research. Talk in seminar Modern Aspects and Applications of Philosophy of Science at TUM, April 2007. `http://www14.informatik.tu-muenchen.de/personen/baumgart/download/public/presentation_CR.pdf`.

[BHS07] Thomas Bocek, Ela Hunt, and Burkhard Stiller. Fast Similarity Search in Large Dictionaries. Technical Report ifi-2007.02, Department of Informatics, University of Zurich, April 2007.

[BKN09] Robert C. Blattberg, Byung-Do Kim, and Scott A. Neslin. *Database Marketing: Analyzing and Managing Customers (International Series in Quantitative Marketing)*. Springer, 1 edition, 4 2009.

[BKS06] Grigory Begelman, Philipp Keller, and Frank Smadja. Automated tag clustering: Improving search and exploration in the tag space. In *Collaborative Web Tagging Workshop at WWW2006, Edinburgh, Scotland*, 2006.

[BLFM05] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986 (Standard), January 2005.

[BM05] Dan Brickley and Alistair Miles. SKOS core vocabulary specification. W3C working draft, W3C, November 2005. http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102.

[BM06] C.H. Brooks and N. Montanez. Improved Annotation of the Blogosphere via Autotagging and Hierarchical Clustering. In *Proceedings of the 15th international conference on World Wide Web*, pages 625–632. ACM New York, NY, USA, 2006.

[BS02] Kurt Bittner and Ian Spence. *Use Case Modeling*. Addison-Wesley Professional, 1 edition, 8 2002.

[BS09] Christian Bizer and Andreas Schultz. The Berlin SPARQL Benchmark. *International Journal On Semantic Web and Information Systems*, 5(2):1–24, 2009.

[BSWZ07] Simone Braun, Andreas Schmidt, Andreas Walter, and Valentin Zacharias. The ontology maturing approach to collaborative and work-integrated ontology development: Evaluation results and future directions. In *International Workshop on Emergent Semantics and Ontology Evolution (ESOE), 6th International Semantic Web Conference (ISWC 2007)*, 2007.

[CBB+10] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Paulo Merson, Robert Nord, and Judith Stafford. *Documenting Software Architectures: Views and Beyond (2nd Edition)*. Addison-Wesley Professional, 2 edition, 10 2010.

[CBHS08a] Ciro Cattuto, Dominik Benz, Andreas Hotho, and Gerd Stumme. Semantic analysis of tag similarity measures in collaborative tagging systems, May 2008.

[CBHS08b] Ciro Cattuto, Dominik Benz, Andreas Hotho, and Gerd Stumme. Semantic grounding of tag relatedness in social bookmarking systems. *The Semantic Web - ISWC 2008*, pages 615–631, 2008.

[CD04] Joel Cracraft and Michael J. Donoghue, editors. *Assembling the Tree of Life*. Oxford University Press, USA, 7 2004.

[CM08] Ed H. Chi and Todd Mytkowicz. Understanding the efficiency of social tagging systems using information theory. In *HT '08: Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, pages 81–88, New York, NY, USA, 2008. ACM.

[CMF08] Duen Horng Chau, Brad Myers, and Andrew Faulring. What to do when search fails: finding information by association. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 999–1008, New York, NY, USA, 2008. ACM.

[CMS09] Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines: Information Retrieval in Practice*. Addison Wesley, 1 edition, 2 2009.

[CR07] Neil A. Campbell and Jane B. Reece. *Biology with MasteringBiology (8th Edition)*. Benjamin Cummings, 8 edition, 12 2007.

[CSB+07] Ciro Cattuto, Christoph Schmitz, Andrea Baldassarri, Vito D. P. Servedio, Vittorio Loreto, Andreas Hotho, Miranda Grahl, and Gerd Stumme. Network Properties of Folksonomies. *AI Communications Journal, Special Issue on "Network Analysis in Natural Sciences and Engineering"*, 20(4):245 – 262, 2007.

[Csi02] Mihaly Csikszentmihalyi. *Flow: Das Geheimnis des Glücks*. Klett-Cotta, 1 2002.

[CV07] R.L. Cilibrasi and P.M.B. Vitanyi. The google similarity distance. *Knowledge and Data Engineering, IEEE Transactions on*, 19(3):370–383, 2007.

[CVW06] Henry Chesbrough, Wim Vanhaverbeke, and Joel West, editors. *Open Innovation: Researching a New Paradigm*. Oxford University Press, USA, 10 2006.

[Dav06] John Davis. *Measuring Marketing: 103 Key Metrics Every Marketer Needs*. Wiley, 1 edition, 11 2006.

[DG04] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI 2004*, pages 137–150, 2004.

[Dij82] Edsger W. Dijkstra. *Selected Writings on Computing: A Personal Perspective (Monographs in Computer Science)*. Springer, 1 edition, 10 1982.

[DM92] William H. DeLone and Ephraim R. McLean. Information Systems Success: The Quest for the Dependent Variable. *Information Systems Research*, 3(1):60, 1992.

[DM03] William H. DeLone and Ephraim R. McLean. The delone and mclean model of information systems success: A ten-year update. *J. Manage. Inf. Syst.*, 19(4):9–30, April 2003.

[DOS03] Michael C. Daconta, Leo J. Obrst, and Kevin T. Smith. *The Semantic Web: A Guide to the Future of XML, Web Services, and Knowledge Management*. Wiley, 5 2003.

[DPH10] Fabrizio De Pasquale and Michael Heiss. Harnessing the Global Innovation Potential of Siemens Workforce. *The Proceedings of The XXI ISPIM Conference 2010*, 21, 2010.

[Ehm08] Karsten Ehms. *Globale Mitarbeiter-Weblogs bei der Siemens AG.*, pages 199–209. Oldenbourg, München, 2008.

[Ehm10] Karsten Ehms. *Persönliche Weblogs in Organisationen – Spielzeug oder Werkzeug für ein zeitgemäßes Wissensmanagement?* PhD thesis, Universität Augsburg, 2010.

[EK09] Karsten Ehms and Walter Christian Kammergruber. Establishing of a semantic multilayer network. patent: US 0049179, Feb. 2009.

[EKSX96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. of 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.

[ES00] Martin Ester and Jörg Sander. *Knowledge Discovery in Databases. Techniken und Anwendungen.* Springer, Berlin, 2000.

[FBH00] Xiaobin Fu, Jay Budzik, and Kristian J. Hammond. Mining navigation history for recommendation. In *IUI '00: Proceedings of the 5th international conference on Intelligent user interfaces*, pages 106–112, New York, NY, USA, 2000. ACM.

[Fie00] Roy T. Fielding. *Architectural Styles and the Design of Network-based Software Architectures.* PhD thesis, University of California, Irvine, 2000.

[FJ09] Lee Feigenbaum and Simon Johnston. SPARQL 1.1 protocol for RDF. W3C working draft, W3C, October 2009. http://www.w3.org/TR/2009/WD-sparql11-protocol-20091022/.

[Fut05] Douglas J. Futuyma. *Evolution.* Sinauer Associates, 1 2005.

[Gas10] Oliver Gassmann. *Crowdsourcing: Innovationsmanagement mit Schwarmintelligenz: Interaktiv Ideen finden - Kollektives Wissen effektiv nutzen - Mit Fallbeispielen und Checklisten.* Carl Hanser Verlag GmbH & CO. KG, 9 2010.

[Gau05] Wilhelm Gaus. *Dokumentations- und Ordnungslehre: Theorie und Praxis des Information Retrieval.* Springer, Berlin, 5., überarb. a. edition, 4 2005.

[GH05] Scott Golder and Bernardo A. Huberman. The structure of collaborative tagging systems, Aug 2005.

[GH06] Scott A. Golder and Bernardo A. Huberman. Usage patterns of collaborative tagging systems. *J. Inf. Sci.*, 32(2):198–208, 2006.

[GHS07] Miranda Grahl, Andreas Hotho, and Gerd Stumme. Conceptual clustering of social bookmarking sites. In *7th International Conference on Knowledge Management (I-KNOW '07)*, pages 356–364, Graz, Austria, SEP 2007. Know-Center.

[GKL10] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. On bootstrapping recommender systems. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1805–1808, New York, NY, USA, 2010. ACM.

[GKVK08] Eirini Giannakidou, Vassiliki Koutsonikola, Athena Vakali, and Yiannis Kompatsiaris. Co-clustering tags and social data sources. In *WAIM '08: Proceedings of the 2008 The Ninth International Conference on Web-Age Information Management*, pages 317–324, Washington, DC, USA, 2008. IEEE Computer Society.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Gru05] Tom Gruber. Ontology of folksonomy: A mash-up of apples and oranges, November 2005. `http://tomgruber.org/writing/ontology-of-folksonomy.htm`.

[Gru09] Tom Gruber. *Ontology*, pages ?–? Springer, 1 edition, 12 2009.

[GT06] Marieke Guy and Emma Tonkin. Folksonomies: Tidying up tags? *D-Lib Magazine*, 12(1), January 2006.

[GTT+06] Benjamin M. Good, Erin M. Tranfield, Poh C. Tan, Marlene Shehata, Gurpreet K. Singhera, John Gosselink, Elena B. Okon, and Mark D. Wilkinson. Fast, cheap and out of control: A zero curation model for ontology development. In *Pacific Symposium on Biocomputing*, pages 128–139, 2006.

[HDM08] Andreas Heß, Philipp Dopichaj, and Christian Maaß. Multi-value classification of very short texts. In *31st Annual German Conference on Artificial Intelligence (KI 2008)*, Kaiserslautern, Germany, September 2008.

[HG08] J. Hendler and J. Golbeck. Metcalfe's law, Web 2.0, and the Semantic Web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1):14–20, 2008.

[HGN00] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining - a general survey and comparison. *SIGKDD Explor. Newsl.*, 2(1):58–64, 2000.

[Hin05] Peter G. Hinman. *Fundamentals of Mathematical Logic*. A K Peters/CRC Press, 2005.

[HJ01] Michael Heiss and J. Jankowsky. The technology tree concept – an evolutionary approach to technology management in a rapidly changing market. In *Change Management and the New Industrial Revolution, 200. IEMC'01 Proceedings.*, pages 37–43. IEEE, 2001.

[HJSS06a] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Emergent semantics in bibsonomy. In Christian Hochberger and Rüdiger Liskowsky, editors, *GI Jahrestagung (2)*, volume 94 of *LNI*, pages 305–312. GI, 2006.

[HJSS06b] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Folkrank: A ranking algorithm for folksonomies. In *Proc. FGIR 2006*, 2006.

[HJSS06c] Andreas Hotho, Robert Jäschke, Christoph Schmitz, and Gerd Stumme. Information Retrieval in Folksonomies: Search and Ranking. In *The Semantic Web: Research and Applications, 3rd European Semantic Web Conference, ESWC 2006, Budva, Montenegro*, volume 4011 of *Lecture Notes in Computer Science*, pages 411–426, Berlin/ Heidelberg, June 2006. Springer.

[HKP06] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 2 edition, 1 2006.

[HML08] Xian-Sheng Hua, Tao Mei, and Shipeng Li. When multimedia advertising meets the new internet era. In *MMSP*, pages 1–5. IEEE Signal Processing Society, 2008.

[HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *Management Information Systems Quarterly*, 28(1):75–106, 2004.

[HNP09] Alon Halevy, Peter Norvig, and Fernando Pereira. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[How09] Jeff Howe. *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*. Crown Business, unedited edition edition, 9 2009.

[HP98] Thomas Hofmann and Jan Puzicha. Statistical models for co-occurrence data. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, February 1998.

[HRGM08] Paul Heymann, Daniel Ramage, and Hector Garcia-Molina. Social tag prediction. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 531–538, New York, NY, USA, 2008. ACM.

[Hub05] Gerhard Hube. *Beitrag zur Beschreibung und Analyse von Wissensarbeit*. PhD thesis, Universität Stuttgart, 2005.

[IOF07] William H. Inmon, Bonnie O'Neil, and Lowell Fryman. *Business Metadata: Capturing Enterprise Knowledge*. Morgan Kaufmann, 10 2007.

[JBS08] Simon Jupp, Sean Bechhofer, and Robert Stevens. A flexible api and editor for skos. In *7th International Semantic Web Conference (ISWC2008)*, October 2008.

[JC94] Yufeng Jing and W. Bruce Croft. An association thesaurus for information retrieval. Technical report, University of Massachusetts, Amherst, MA, USA, 1994.

[JJF08] Akshay Java, Anupam Joshi, and Tim Finin. Detecting commmunities via simultaneous clustering of graphs and folksonomies. In *WebKDD 2008 Workshop on Web Mining and Web Usage Analysis*, August 2008.

[JPGB05] William Jones, Ammy J. Phuwanartnurak, Rajdeep Gill, and Harry Bruce. Don't Take My Folders Away! Organizing Personal Information to Get Things Done. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1505–1508, New York, NY, USA, 2005. ACM Press.

[KH09] Gerhard Käfer and Michael Heiss. Wissensnetze als Basis für Enterprise 2.0 - Ein Erfahrungsbericht der Siemens AG aus 10 Jahren Wissensvernetzung als Basis für die Einführung von Enterprise 2.0. *"Geteiltes Wissen ist doppeltes Wissen!" KnowTech 2009*, pages 201–205, 2009.

[Kli13] Gyula Klima. The medieval problem of universals. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Fall 2013 edition, 2013.

[Kru05] Steve Krug. *Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition*. New Riders Press, 2nd edition, 8 2005.

[KSB+08] Hak-Lae Kim, Simon Scerri, John Breslin, Stefan Decker, and Hong-Gee Kim. The state of the art in tag ontologies: A semantic model for tagging and folksonomies. In *International Conference on Dublin Core and Metadata Applications*, Berlin, Germany, 2008.

[KSHS08] Beate Krause, Christoph Schmitz, Andreas Hotho, and Gerd Stumme. The anti-social tagger - detecting spam in social bookmarking systems. In *Proc. of the Fourth International Workshop on Adversarial Information Retrieval on the Web*, 2008.

[KSP09] Joep J. M. Kierkels, Mohammad Soleymani, and Thierry Pun. Queries and tags in affect-based multimedia retrieval. In *ICME'09: Proceedings of the 2009 IEEE international conference on Multimedia and Expo*, pages 1436–1439, Piscataway, NJ, USA, 2009. IEEE Press.

[KVEL10] Walter Christian Kammergruber, Maximilian Viermetz, Karsten Ehms, and Manfred Langen. Using association rules for discovering tag bundles in social tagging data. *6th International Conference on Next Generation Web Services Practices (NWeSP 2010)*, 6, 2010.

[KVZ09] Walter Christian Kammergruber, Maximilian Viermetz, and Cai-Nicolas Ziegler. Discovering communities of interest in a tagged on-line environment. In *CASoN2009: Proceedings of the 1st International Conference on Computational Aspects of Social Networks*, 2009.

[KZ10] Walter Christian Kammergruber and Werner Zucker. Method and an apparatus for matching data network resources. patent: US 0059786, Oct. 2010.

[Lar04] Daniel T. Larose. *Discovering Knowledge in Data: An Introduction to Data Mining*. Wiley-Interscience, 1 edition, 11 2004.

[LEC07] David Laniado, Davide Eynard, and Marco Colombetti. Using WordNet to turn a folksonomy into a hierarchy of concepts. *Proceedings of SWAP 2007, the 4th Italian Semantic Web Workshop*, page 192, 2007.

[LGZ08] Xin Li, Lei Guo, and Yihong E. Zhao. Tag-based social interest discovery. In *Proceedings of the 17th International World Wide Web Conference*, pages 675–684. ACM, 2008.

[LH05] S. Lasser and M. Heiss. Collaboration maturity and the offshoring cost barrier: the tradeoff between flexibility in team composition and cross-site communication effort in geographically distributed development projects. In *Professional Communication Conference, 2005. IPCC 2005. Proceedings. International*, pages 718–728. IEEE, 2005.

[LHMK12] Manfred Langen, Michael Heiss, Thomas Mayerdorfer, and Walter Christian Kammergruber. Intelligent message distribution in corporate expert communities. In *Engineering, Technology and Innovation (ICE), 2012 18th International ICE Conference on*, pages 1–9, 2012.

[Lin08] Bernd Lindner. Der Einsatz von Wikis in der Siemens AG. I-KNOW, 2008.

[LJL+07] K.R. Lakhani, L.B. Jeppesen, P.A. Lohse, J.A. Panetta, and Harvard Business School. Division of Research. *The Value of Openess in Scientific Problem Solving*. Division of Research, Harvard Business School, 2007.

[LK08] Manfred Langen and Walter Christian Kammergruber. Tagging versus Ontologien? Informationsstrukturierung im Enterprise 2.0. In *KnowTech 2008 - Mehr Wissen-mehr Erfolg*, 2008.

[LKE11] Manfred Langen, Walter C. Kammergruber, and Karsten Ehms. Context-specific information distribution using the web 3l model. In *Next Generation Web Services Practices (NWeSP 2011)*, 2011.

[Mat04] Adam Mathes. Folksonomies - cooperative classification and communication through shared metadata. *Computer Mediated Communication - LIS590CMC*, December 2004.

[MB09] P. Mitra and K. Baid. Targeted Advertising for Online Social Networks. In *First International Conference on Networked Digital Technologies*, pages 366–372. IEEE, 2009.

[McA06] Andrew P. McAfee. "Enterprise 2.0: The Dawn of Emergent Collaboration" . reprint 47306. *MIT Sloan Management Review*, 47(3):21–28, 2006.

[MCM+09] Benjamin Markines, Ciro Cattuto, Filippo Menczer, Dominik Benz, Andreas Hotho, and Gerd Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *18th International World Wide Web Conference*, pages 641–641, April 2009.

[MEPG07] S. McCoy, A. Everard, P. Polak, and D.F. Galletta. The effects of online advertising. *Communications of the ACM*, 50(3):84–88, 2007.

[MHH06] V. Mikulovic, M. Heiss, and J.D. Herbsleb. Practices and Supporting Structures for Mature Inquiry Culture in Distributed Software Development Projects. In *Global Software Engineering, 2006. ICGSE'06. International Conference on*, pages 245–246. IEEE, 2006.

[Mic09] Microsoft Patterns & Practices Team. *Microsoft Application Architecture Guide (Patterns & Practices)*. Microsoft Press, second edition edition, 11 2009.

[Mil56] G.A. Miller. The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *Psychological review*, 63(2):81, 1956.

[Mil95] George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41, November 1995.

[Mis06] Gilad Mishne. Autotag: a collaborative approach to automated tag assignment for weblog posts. In *Proceedings of the 15th international conference on World Wide Web*, WWW '06, pages 953–954, New York, NY, USA, 2006. ACM.

[MM04] Frank Manola and Eric Miller. RDF Primer, W3C Recommendation. `http://www.w3.org/TR/rdf-primer/`, 2004.

[MM11] Perry Marshall and Thomas Meloche. *Ultimate Guide to Facebook Advertising: How to Access 600 Million Customers in 10 Minutes*. Entrepreneur Press, 1 edition, 9 2011.

[MNBD06] Cameron Marlow, Mor Naaman, Danah Boyd, and Marc Davis. Ht06, tagging paper, taxonomy, flickr, academic article, to read. In *HYPERTEXT '06: Proceedings of the seventeenth conference on Hypertext and hypermedia*, pages 31–40, New York, NY, USA, 2006. ACM Press.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schuetze. *Introduction to Information Retrieval*. Cambridge University Press, July 2008.

[MS11] Johannes Müller and Andreas Stocker. Enterprise microblogging for advanced knowledge sharing: the references@ bt case study. *Journal of Universal Computer Science*, 17(4):532–547, 2011.

[MSVV07] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22, 2007.

[MSW92] Richard Mander, Gitta Salomon, and Yin Yin Wong. A "pile" metaphor for supporting casual organization of information. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 627–634, New York, NY, USA, 1992. ACM.

[MTT98] Rila Mandala, Takenobu Tokunaga, and Hozumi Tanaka. The Use of WordNet in Information Retrieval. In Sanda Harabagiu, editor, *Use of WordNet in Natural Language Processing Systems: Proceedings of the Conference*, pages 31–37. Association for Computational Linguistics, 1998.

[Mul07] Michael J. Muller. Comparing tagging vocabularies among four enterprise tag-based services. In *GROUP '07: Proceedings of the 2007 international ACM conference on Supporting group work*, pages 341–350, New York, NY, USA, 2007. ACM.

[Nav01] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33:31–88, March 2001.

[Nel74] Theodor H. Nelson. *Computer Lib/Dream Machines*. Distributors, 6 1974.

[Nie93] Jakob Nielsen. *Usability Engineering*. Morgan Kaufmann, 1st edition, 9 1993.

[Nie99] Jakob Nielsen. *Designing Web Usability*. Peachpit Press, 1 edition, 12 1999.

[Pan06] Jiri Panyr. Thesauri, Semantische Netze, Frames, Topic Maps, Taxonomien, Ontologien – begriffliche Verwirrung oder konzeptionelle Vielfalt? *Information und Sprache. Festschrift für Harald H. Zimmermann*, pages 139–151, 2006.

[PB06] Tassilo Pellegrini and Andreas Blumauer, editors. *Semantic Web: Wege zur vernetzten Wissensgesellschaft*. Springer, Berlin, 1 edition, 5 2006.

[PBMW99] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[Pet09] Isabella Peters. *Folksonomies: Indexing and Retrieval in the Web 2.0 (Knowledge and Information)*. De Gruyter Saur, 10 2009.

[PH02] Jack Park and Sam Hunting. *XML Topic Maps: Creating and Using Topic Maps for the Web*. Addison-Wesley Professional, 7 2002.

[Pop02] Karl Popper. *The Logic of Scientific Discovery*. Routledge, 2 edition, 3 2002.

[Por80] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[PPM04] T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet:: Similarity-measuring the Relatedness of Concepts. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1024–1025. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2004.

[RAC+02] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. Getting to know you: Learning new user preferences in recommender systems. In *Proceedings of the 7th International Conference on Intelligent User Interfaces*, IUI '02, pages 127–134, New York, NY, USA, 2002. ACM.

[Rao48] C.R. Rao. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological)*, 10(2):159–203, 1948.

[Rau14] Axel Rauschmayer. *Speaking JavaScript*. O'Reilly Media, 1 edition, 3 2014.

[RR07] Leonard Richardson and Sam Ruby. *Restful Web Services*. O'Reilly Media, illustrated edition edition, 5 2007.

[SGMB08] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 259–266, New York, NY, USA, 2008. ACM.

[Shi05] Clay Shirky. Ontology is overrated: Categories, links, and tags, 2005. `http://www.shirky.com/writings/ontology_overrated.html`.

[SHJS06] Christoph Schmitz, Andreas Hotho, Robert Jäschke, and Gerd Stumme. Mining association rules in folksonomies. In *Data Science and Classification*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 261–270. Springer Berlin Heidelberg, 2006.

[Sin05] Rashmi Sinha. A cognitive analysis of tagging, September 2005. `http://www.rashmisinha.com/archives/05_09/tagging-cognitive.html`.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA, 2001. ACM.

[Spo11] Marshall Sponder. *Social Media Analytics: Effective Tools for Building, Interpreting, and Using Metrics*. McGraw-Hill, 1 edition, 7 2011.

[SS08] Wolfgang G. Stock and Mechtild Stock. *Wissensrepräsentation: Auswerten und Bereitstellen von Informationen*. Oldenbourg, 5 2008.

[SSB05] Ellen Spertus, Mehran Sahami, and Orkut Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, KDD '05, pages 678–684, New York, NY, USA, 2005. ACM.

[Sur05] James Surowiecki. *The Wisdom of Crowds*. Anchor, 8 2005.

[TDB+06] Tudhope, Douglas, Binding, Ceri, Blocks, Dorothee, Cunliffe, and Daniel. Query expansion via conceptual distance in thesaurus indexed collections. *Journal of Documentation*, 62(4):509–533, 2006.

[Ten06] Joseph T. Tennis. Social tagging and the next steps for indexing. In Jonathan Furner and Joseph T. Tennis, editors, *Proceedings 17th SIG/CR Classification Research Workshop*, 2006.

[Ton06] Emma Tonkin. Searching the long tail: Hidden structure in social tagging. *Proceedings of the 17th SIG Classification Research Workshop*, 2006.

[TSMM08] Jennifer Thom-Santelli, Michael J. Muller, and David R. Millen. Social tagging roles: publishers, evangelists, leaders. In *CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 1041–1044, New York, NY, USA, 2008. ACM.

[Van07a] Thomas Vander Wal. Explaining and Showing Broad and Narrow Folksonomies, 2007. `http://www.vanderwal.net/random/entrysel.php?blog=1635`.

[Van07b] Thomas Vander Wal. Folksonomy. Folksonomy Coinage and Definition, 2007. `http://vanderwal.net/folksonomy.html`.

[VDHS07] C. Van Damme, Martin Hepp, and K. Siorpaes. Folksontology: An integrated approach for turning folksonomies into ontologies. *Bridging the Gap between Semantic Web and Web*, 2:57–70, 2007.

[VH08] Asir S. Vedamuthu and Mary Holstege. W3C xml schema definition language (XSD): Component designators. Last call WD, W3C, November 2008. http://www.w3.org/TR/2008/WD-xmlschema-ref-20081117/.

[Vos07] Jakob Voss. Tagging, folksonomy & co - renaissance of manual indexing?, Jan 2007.

[WAH+12] Clemens Wiener, Isaac Newton Acquah, Michael Heiss, Thomas Mayerdorfer, Manfred Langen, and Walter Christian Kammergruber. Targeting the Right Crowd for Corporate Problem Solving – a Siemens Case Study with TechnoWeb 2.0. In *IEEE International Technology Management Conference*, 2012.

[WD08] Jian Wang and Brian D. Davison. Explorations in tag suggestion and query expansion. In *SSM '08: Proceeding of the 2008 ACM workshop on Search in social media*, pages 43–50, New York, NY, USA, 2008. ACM.

[Wei05] David Weinberger. Trees versus leaves, 2005. `http://www.hyperorg.com/backissues/joho-jan28-05.html`.

[Wei07] David Weinberger. *Everything Is Miscellaneous: The Power of the New Digital Disorder*. Times Books, May 2007.

[Wel07] Katrin Weller. Folksonomies and Ontologies. Two New Players in Indexing and Knowledge Representation. In H. Jezzard, editor, *Applying Web 2.0. Innovation, Impact and Implementation*, pages 108–115, 2007.

[Wel10] Katrin Weller. *Knowledge Representation in the Social Semantic Web (Knowledge and Information)*. K G Saur Verlag, 1 edition, 11 2010.

[Wer85] Gernot Wersig. *Thesaurus - Leitfaden. Eine Einführung in das Thesaurus-Prinzip in Theorie und Praxis*. Verlag Dokumentation Saur KG, 2., erg. a. edition, 4 1985.

[Wik12] Wikipedia. Apriori algorithm — Wikipedia, The Free Encyclopedia, 2012. `http://en.wikipedia.org/w/index.php?title=Apriori_algorithm&oldid=484551302` [Online; accessed 29-March-2012].

[WP08] Katrin Weller and Isabella Peters. Seeding, weeding, fertilizing. different tag gardening activities for folksonomy maintenance and enrichment. In Sören Auer, Sebastian Schaffert, and Tassilo Pellegrini, editors, *Proceedings of I-Semantics'08, International Conference on Semantic Systems. Graz, Austria, September 3-5*, pages 10–117, 2008.

[YLW+09] J. Yan, N. Liu, G. Wang, W. Zhang, Y. Jiang, and Z. Chen. How much can behavioral targeting help online advertising? In *Proceedings of the 18th international conference on World wide web*, pages 261–270. ACM, 2009.

[You08] G. Oliver Young. Global Enterprise Web 2.0 Market Forecast: 2007 To 2013. Forrester Research, Inc, 2008.

[ZB07] Valentin Zacharias and Simone Braun. Soboleo - social bookmarking and lightweight ontology engineering. In *Workshop on Social and Collaborative Construction of Structured Knowledge (CKC), 16th International World Wide Web Conference (WWW 2007)*, May 2007.

[ZC08] Valentina Zanardi and Licia Capra. Social Ranking: Uncovering Relevant Content Using Tag-based Recommender Systems. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 51–58, New York, NY, USA, 2008. ACM.

[ZMH09] Sabrina Ziebarth, Nils Malzahn, and H. Ulrich Hoppe. Using data mining techniques to support the creation of competence ontologies. In *Proceedings of the 14th International Conference on Artificial Intelligence in Education (AIED 2009)*, Brighton, England, July 2009.

[ZZ11] Dan Zarrella and Alison Zarrella. *The Facebook Marketing Book*. O'Reilly Media, 1 edition, 1 2011.

APPENDIX A

Interviews

The interviews conducted between 30th of November 2012 and 6th of December 2012. All interviews are guideline-based interviews with a predefined set of questions. Individual small adjustments based on the application and the course of the interview may have been made. The interviews A.2, A.5, and A.4 are telephone interviews. This is due to fact that the interviewee are distributed to different locations in Austria, Germany, and Switzerland. The interviews A.1 and A.3 where made in persona in Munich. Every interviewed person is a native German speaker. Therefore the interviews where held in German. Each interview took about 30 minutes in time.

A.1. Blogosphere And Community Hub in Global Intranet Portal

Interview based on the memory of the interviewer. There were technical issues with a used voice recording device. The interview has been checked for correctness by the interviewed person. The interview has been conducted on November 29th 2012 at 1.00 - 1.40 pm with Dr. K. E., Senior Researcher and Project Manager at Siemens Corporate Technologies.

Interviewer: Welche Rolle haben sie in Bezug auf ihre Applikation? Sind sie Project Manager oder Programmierer?

Interviewee: Ich bin eher Project Manager.

Interviewer: Sie haben aber auch eine gewisse Erfahrung mit Programmierung?

Interviewee: Könnte man so sagen.

Interviewer: Sind sie auch Endanwender in täglicher Arbeit und haben somit als normaler Anwender Kontakt mit STAGS?

Interviewee: Ich habe auch Erfahrung als Endanwender mit STAGS sammeln können.

Interviewer: Welche Software-Entwicklungserfahrung haben sie?

Interviewee: Ich hatte mit eher älteren Programmiersprachen zu tun, z.B. C oder ähnlichem, aber auch mit Skript-Sprachen.

Interviewer: An welchen Stellen wurde STAGS eingebaut?

Interviewee: Im Community Hub wurde STAGS zur Auflistung von Ressourcen in Form eines Activity-Streams über verschiedene Applikationen integriert.

Interviewer: Hatten sie Kontakt mit der STAGS REST API?

Interviewee: Ja, produktiv und prototypisch.

Interviewer: Welche Features von STAGS nutzen Sie?

Interviewee: Produktiv: Die Resource-API. Prototypisch habe ich Erfahrung mit Tag-Suggestions und der Tag-Cloud.

Interviewer: Wie hoch war der Aufwand bzgl. der Zeit zur Einbindung von STAGS in ihre Applikation?

Interviewee: Eher im Stundenbereich. Der Aufwand war eher gering.

Interviewer: Wie schätzen sie die Schwierigkeit der Integration in ihre Applikation ein?

Interviewee: Die Schwierigkeit der Einbindung ist eher gering. Es bestehen natürlich im die Herausforderung der Einbindung in einem Enterprise-Kontext, z.B. Zertifikate bei HTTPS.

Interviewer: Aber das dürfte man von einen normalen Entwickler, also keinen Experten, also jemand der über 10 Jahre Erfahrung hat, erwarten?

Interviewee: Ja, sofern jemand mit modernen Web-Technologien vertraut ist, dürfte das ohne große Probleme zu schaffen sein.

Interviewer: Wie schwierig ist die API zu verstehen?

Interviewee: Sie ist relativ intuitiv verstehbar.

Interviewer: Wie schwierig ist die API zu benutzen, d.h. nachdem man sie verstanden hat?

Interviewee: Es wird sich in der Regel keiner der Applikation-Owner mit der kompletten API beschäftigen. Die Teile mit denen sich jemand beschäftigt hat, dürften dann einfach zu benutzen sein.

Interviewer: Wie würden sie die Flexibilität der STAGS API einschätzen, also bzgl. der Setzung der Parameter?

Interviewee: Für die Fälle, mit denen ich zu tun hatte, war die Flexibilität ausreichend und die Modifizierbarkeit der Parameter hat meinen Anforderungen entsprochen.

Interviewer: Wie würden sie die Anpassbarkeit der bereitgestellten Widgets beurteilen?

Interviewee: Es geht, ja zunächst um die REST-API und die HTML-Widgets. Die Widgets dürften über CSS-Klassen, sofern sie adäquat gesetzt sind – das kann ich jetzt nicht genau sagen – relativ einfach zu gestalten sein.

Interviewer: Konkret für ihre Anwendung?

Interviewee: Da waren die Gestaltungsmöglichkeiten ausreichend.

Interviewer: Wie hoch schätzen sie die Verfügbarkeit von STAGS ein?

Interviewee: So ca. 95 % oder evtl. höher. Das kann ich nicht wirklich konkret ohne Zahlen sagen. Aber nachdem sie mich nach der Einschätzung gefragt haben: So im höheren 90-Prozent-Bereich.

Interviewer: Ist die Verfügbarkeit somit ausreichend für sie?

Interviewee: Natürlich ist jeder Ausfall unschön. Allerdings war das nicht wirklich schlimm.

Interviewer: Entspricht die Performance von STAGS ihren Bedürfnissen?

Interviewee: Ja, auf jeden Fall. Man merkt da keine Probleme bzgl. der Geschwindigkeit, wenn STAGS in eine Applikation eingebaut ist. Auch mit der steigenden Ressourcenmenge, es kommen ja ständig neue hinzu, gab es keine nennenswerten Geschwindigkeits-Einbußen.

Interviewer: Wie schätzen sie die Zuverlässigkeit von STAGS ein? Also im Sinne von: Wenn STAGS läuft, dass dann auch eine Antwort kommt.

Interviewee: Es ist da ein bisschen schwierig zu unterscheiden zwischen beiden Sachen: Zuverlässigkeit und Verfügbarkeit. Das ist nicht einfach einzuschätzen. Ich würde da ebenfalls sagen: Im höheren 90-Prozent-Bereich.

Interviewer: Ist die Zuverlässigkeit ausreichend?

Interviewee: Ja, wie gesagt: Es gab bis jetzt keine schlimmeren Auswirkung bei Ausfällen.

Interviewer: Wie relevant sind die gelieferten Daten der unterschiedlichen API-Elemente?

Interviewee: Der Relevanzbegriff ist ein bisschen schwierig zu deuten.

Interviewer: Ich meine, sind die Daten sinnvoll, bzw. brauchbar?

Interviewee: Ich würde sagen: Ja. Allerdings ist das nur ein Eindruck. Tatsächliche ausführliche Tests habe ich dazu nicht durchgeführt.

Interviewer: Wie vollständig sind die gelieferten Daten, bzgl. der Resource-API?

Interviewee: Das kann ich nur anhand von Stichproben sagen. D.h. wenn ich gewusst habe, dass ein Item existiert und ich mit entsprechenden Parametern danach gesucht habe, so wurden sie, soweit ich mich erinnern kann, immer in den jeweiligen Listen aufgeführt.

Interviewer: Entsprechen die Rückgabewerte den Erwartungen an die API?

Interviewee: Das ist schwierig zu verstehen.

Interviewer: Ich meine, liefert die API das zurück, was sie soll?

Interviewee: Im Prinzip ja. Ich konnte nichts Gegenteiliges feststellen.

Interviewer: Wie schwierig/einfach sind die UI-Elemente von STAGS zu benutzen?

Interviewee: Das ist natürlich sehr user-abhängig. Ich würde sagen, dass sie relativ einfach zu benutzen sind. Für jemand, der eine normale Intranet-Applikation benutzen kann, dürfte die Elemente einfach zu benutzen sein.

Interviewer: Wie schätzen sie die Nützlichkeit von STAGS für den Endanwender ein? Wird die individuelle Produktivität erhöht? Gibt es Lerneffekte für den Nutzer?

Interviewee: Ich denke die Applikation ist sehr nützlich. Vor allem der systemübergreifende Nutzen ist sehr entscheidend. Wir haben im Moment das Problem, dass Nutzer mit verschiedenen Applikationen arbeiten, es aber noch keine hinreichende Integration der verschiedenen Plattformen gibt. STAGS ist da sehr nützlich. Wir haben allerdings noch nicht das komplette Potential von STAGS ausgeschöpft. Ich würde eher sagen erst die ersten 20 % . Vor allem die semantischen Fähigkeiten sind sehr viel versprechend.

Interviewer: Welchen erwarteten Nutzen hatten sie?

Interviewee: Der erwartete Nutzen war die Zusammenführung von verschiedenen Systemen in einen Activitiy-Stream, der entsprechend gefiltert werden kann.

Interviewer: Ist der Nutzen eingetreten?

Interviewee: Ich würde sagen, ja.

Interviewer: War der Aufwand vertretbar für den Nutzen?

Interviewee: Definitiv, ja. Wirkliche Zahlen sind da immer schwer zu nennen.

Interviewer: Hat es sich somit gelohnt?

Interviewee: Ja.

Interviewer: Wie sieht der tatsächliche Nutzen aus? Hat STAGS für sie einen Business Impact?

Interviewee: Das ist schwierig zu sagen, da so was nicht wirklich ohne weiteres zu messen ist.

Interviewer: Hatten sie Kosteneinsparungen?

Interviewee: Das kann man so nicht sagen. Es gibt ja keinen Vorher-Naher-Vergleich. Das System wurde neu basierend auf STAGS eingeführt.

Interviewer: Hat sich die Beliebtheit der Plattform gesteigert?

Interviewee: Kann man somit auch nicht sagen.

Interviewer: Wurde die Usability verbessert?

Interviewee: So konkret auch nicht beantwortbar. Jedenfalls ist es jetzt möglich über verschiedene Applikationen hinweg Ressourcen zu finden.

Interviewer: Hatten sie alternative Lösungen?

Interviewee: Denkbar wäre die Nutzung einer Suchmaschine gewesen. Dabei ist die Nutzung der API wesentlich komplexer. Rückgabewerte sind zumeist in XML-Formaten. Die Anfragen sind in der Regel über Volltext-Suchen. Man kann zwar auf so genannte Keywords einschränken. Allerdings interpretiert dieses Feld jede Applikation anders. Auch das Ranking ist nicht so einfach zu beeinflußen. Im Endeffekt bekommt man keine tatsächlichen Activity-Streams und die Nutzung erfordert mehr „Verrenkungen“. Was natürlich eine Rolle spielt, sind andere Features von STAGS wie das Zusammenführen von Begriffen über semantische Relationen.

Interviewer: Sie hatten ja als Projektmanager die Gelegenheit den Thesaurus-Editor auszuprobieren und haben Demos dazu gesehen.

Interviewee: Dieses Interface ist natürlich einzigartig und so was habe ich so noch nirgends gesehen.

Interviewer: Diese Komponente ist ja im Moment noch prototypisch.

Interviewee: Ja, leider. Es besteht jedenfalls laut meiner Erfahrung Bedarf für eine solche Komponente. Auch die personenbezogenen Features, wie Suche nach Experten, finde ich sehr spannend und innovativ.

Interviewer: Nochmals zu den alternativen Lösungen. Wenn ja, welche und warum haben sie sich gegen diese entschieden? Das haben sie nun eh schon beantwortet. D.h. die Einfachheit war das wesentliche Kriterium.

Interviewee: Ja, so manches wäre wahrscheinlich über die Suchmaschine auch möglich gewesen. Eben aber nur nicht so einfach und möglicherweise auch nicht in der Art und Weise.

Interviewer: Wenn nein, welche Aspekte an STAGS sind alternativlos?

Interviewee: Die wirklich einfache Nutzung der API über JSONP und über die HTML-Widgets sind schon sehr entscheidend. Auch die anderen bereits erwähnten Komponenten.

A.2. Community 4 Competences

The interview has not been transcribed literally. The interview has been checked for correctness by the interviewed person. The interview has been conducted on December 5th, 2012 at 2.00 - 2.35 pm with Mr. F. K., a Senior Developer at Siemens Industry Division.

Interviewer: Welche Rolle haben sie in Bezug auf ihre Applikation? Sind sie Project Manager oder Programmierer?

Interviewee: Ich bin das teils, teils. Ich war vorher für die Weiterentwicklung verantwortlich. Da auch mehr technisch, aber nicht im Doing beschäftigt. Jetzt mache ich mehr die Requirements. Also, ich habe einen technischen Hintergrund, aber an C4C „rumgeschraubt“ habe ich nur an wenigen Stellen.

Interviewer: Sind sie auch Endanwender in täglicher Arbeit und haben somit als normaler Anwender Kontakt mit STAGS?

Interviewee: Ja, habe ich.

Interviewer: Welche Software-Entwicklungserfahrung haben sie?

Interviewee: Ich bin gelernter Informatiker. Ich bin reiner „technischer“ Informatiker – also keine Wirtschaftsinformatik oder ein Derivat.

Interviewer: An welchen Stellen wurde STAGS eingebaut?

Interviewee: Zunächst zwei Unterscheidungen: Einmal geht es darum, wie Tags aus dem System rauskommen und wie sie angelegt werden. Wir haben zwei Plattformen: Ein Confluence-Wiki und eine SharePoint-Installation. Da hat man nach einer Möglichkeit gesucht plattform-übergreifend zu taggen, d.h. in unserer Applikation aber plattform-übergreifend. Die Tag-Vorschläge

nutzen wir in beiden Systemen. Die werden auch aus beiden rausgeholt und an STAGS als eine große Liste geschickt. STAGS wird bei uns als „browse tags“ eingesetzt. Dazu verwenden wir ein Confluence-Plugin.

Interviewer: Mit dem Plugin meinen sie eine Filterung von Information-Items, also Wiki-Pages oder SharePoint-Items, mit Hilfe von Tags?

Interviewee: Genau. In Confluence heißen die Tags „Labels“. Wir nutzen die Tags in beiden System. Die Einbindung von „browse tags“ ist jedoch nur im Wiki eingebaut.

Interviewer: Hatten sie Kontakt mit der STAGS REST API?

Interviewee: Indirekt. Ich habe für C4C nichts selber entwickelt. Ich habe mir die API angeschaut. Ich habe ein Tool geschrieben, was auswertet welche Tags bei uns vergeben werden. Und ähnliche „Geschichten“. Auf der Ebene habe ich mich bewegt.

Interviewer: Welche Features von STAGS nutzen Sie?

Interviewee: Wir nutzen die Tag-Vorschläge und die Anzeige der Tags. Das sind die zwei Sachen, die wir nutzen. Als Anmerkung, für die Tag-Vorschläge nutzen wir nur Tags, die in einem von unseren beiden Plattformen vergeben wurden, also in SharePoint oder im Wiki. D.h. vereinfacht gesagt: Wir nutzen STAGS um unsere Plattformen miteinander abzugleichen. Hintergrund dazu – was evtl. für interessant sein dürfte: Wir verwenden zu gewissen Themen Standard-Tags, z.B. „project_management“ als Tag für project management. Das hat zum Teil historische Gründe. In anderen Plattformen wird das evtl. anders gehandhabt. Da sind vielleicht „project“ und „management“ eigene Tags. Daher haben wir uns, auch bzgl. unseres Style-Guides, dazu entschieden die Tag-Vorschläge auf *unsere* Tags einzuschränken.

Interviewer: Das ist natürlich von Plattform zu Plattform unterschiedlich, wie frei Tags vergeben werden dürfen.

Interviewee: Bei uns gibt es eben einen gewissen Style-Guide zur Tag-Vergabe.

Interviewer: Wie hoch war der Aufwand bzgl. der Zeit zur Einbindung von STAGS in ihre Applikation?

Interviewee: Für uns war der Aufwand nur an zwei Stellen. Zum einen das Einbauen der Tag-Vorschläge und der Export. Das Confluence-Plugin wurde uns bereitgestellt.

Interviewer: Der zeitliche Aufwand so ungefähr?

Interviewee: Der Export der Tags aus Confluence wurde ebenfalls durch ein bereitgestelltes Plugin gewährleistet. D.h. ich kann da nur was zu SharePoint sagen und wie aus den beiden Listen, d.h. SharePoint und Confluence, eine große Liste erzeugt wird. Ich würde mal sagen, der Aufwand war dazu eine Woche. Bei den ersten Versionen von STAGS war zunächst die Performance noch sehr schlecht. Da war der Test-Aufwand noch ein bisschen höher. Wir haben da mehrere Iterationen durchgeführt, bis das produktiv eingesetzt wurde. Wir hatten da das Problem eines Early-Adopters, der mit solchen

Kinderkrankheiten noch zu leben hat. Insgesamt war es, wie gesagt eine Woche, bis wir den Tag-Export mit Tests und Dokumentation fertig hatten. Der Hauptaufwand war beim Export. Für die Anzeige hatten wir weniger Aufwände. Die Aufwände für den Export aus dem Wiki ist da nicht mit eingerechnet.

Interviewer: Da gab es einen Synergieeffekt mit der Wikisphere. Das ist dieselbe Plattform.

Interviewee: Ja, genau.

Interviewer: Wie schätzen sie die Schwierigkeit der Integration in ihre Applikation ein?

Interviewee: Ich sage es mal so: Die Schnittstelle ist relativ einfach. Was ein bisschen „tricky“ war, ist der Einbau in SharePoint. Dazu war ein JavaScript-Hack nötig. Das liegt allerdings an der SharePoint-Plattform, die in diesem Bereich ein paar Schwächen vorweist. Das Problem liegt somit nicht an der Schnittstelle oder an STAGS. Es liegt vielmehr daran, dass viele Plattformen eine Integration von fremden Content gar nicht vorsehen. Die sind da zu „silo-artig“ aufgebaut.

Interviewer: D.h. für sie war die Schwierigkeit bei der Integration in SharePoint?

Interviewee: Ja, wir hatten ja nur diesen Teil zu implementieren. Somit kann ich das nur von der SharePoint-Seite beurteilen. Da einzubauen, dass die Tag-Vorschläge nicht aus dem Standard kommen, war eher schwierig. Unsere Lösung war nicht „quick and dirty“ sondern anders formuliert: Man musste sich schon ein bisschen „verbiegen“ . Also das war nicht so einfach. Bei anderen Plattformen tritt das Problem vielleicht gar nicht auf, weil da gewisse UI-Elemente leichter unter Kontrolle zu bringen sind und so was von der Plattform schon vorgesehen ist.

Interviewer: Wie schwierig ist die API zu verstehen?

Interviewee: Die API ist recht intuitiv. Das ist ja kein Hexenwerk. Wenn man weiß, worum es geht, dann bekommt man das auch hin.

Interviewer: Wie schwierig ist die API zu benutzen, d.h. nachdem man sie verstanden hat?

Interviewee: Ziemlich einfach. Die Schwierigkeiten lagen nicht daran sie zu verstehen sondern sie in das UI von SharePoint einzubinden.

Interviewer: Wie würden sie die Flexibilität der STAGS API einschätzen?

Interviewee: Wir haben da ein gewisses Set an Funktionalitäten vorgegeben gehabt. Darüber wurde dann auch nicht mehr allzu viel diskutiert. Ich persönlich habe das nicht implementiert und kann dazu nicht viel sagen.

Interviewer: Wie würden sie die Anpassbarkeit der bereitgestellten Widgets beurteilen?

Interviewee: Bzgl. der Widgets kann ich nichts sagen. Das hat ein Kollege von ihnen bei uns eingebunden.

Interviewer: OK. Aber würden sie sagen, dass es auffällt, dass ein fremdes System eingebunden ist?

Interviewee: Nein, wie gesagt: Bei uns ist es als Plugin bei Confluence integriert. Man sieht, dass es sich nicht komplett um den Confluence-Standard handelt, weil dort die Tags ein bisschen anders gehandhabt werden. Aber ich würde sagen: Das UI ist in Ordnung. Die Plattform Styleguides, z.B. die Farbe der Links konnte man parametrisieren. Insofern ist das in Ordnung.

Interviewer: Wie hoch schätzen sie die Verfügbarkeit von STAGS ein?

Interviewee: Jetzt, da STAGS einen Level erreicht hat, der ein gewissen Betriebszustand erreicht widerspiegelt, würde ich sagen: Gut. Je weniger ich mich damit beschäftigen muss, d.h. mit Problemen, desto besser ist es. Ich würde sagen: Da haben wir andere Module, die mehr Ärger machen. Mit STAGS haben wir relativ wenige Probleme.

Interviewer: Ist die Verfügbarkeit somit ausreichend für sie?

Interviewee: Ja, das passt.

Interviewer: Entspricht die Performance von STAGS ihren Bedürfnissen?

Interviewee: Ja, das ist schnell genug. Ganz am Anfang, wie erwähnt, gab es leichte Performance-Probleme. Das hat sich mittlerweile gelegt.

Interviewer: Wie schätzen sie die Zuverlässigkeit von STAGS ein?

Interviewee: Die Zuverlässigkeit ist auch gut. Da habe wir nichts auszusetzen.

Interviewer: Was würde das so ungefähr in Prozent sein? Eher 50, 80 oder 90 %?

Interviewee: Ich würde eher sagen so gute 90 %. Wir haben das auch nicht gemessen. Mit STAGS hatten wir wenige Probleme.

Interviewer: Ist die Zuverlässigkeit ausreichend?

Interviewee: Die Zuverlässigkeit ist gut genug. Also wenn sie da eine Wertung haben wollen, wie beispielsweise fünf Sterne von fünf möglichen, dann ist das so. Also, wie gesagt mit dem Modul haben wir keinen Ärger.

Interviewer: Wie relevant sind die gelieferten Daten der unterschiedlichen API-Elemente?

Interviewee: Da hätte ich gesagt, dass das passt. Wie das mit größeren Datenmengen aussieht, kann ich nicht beurteilen. Wir haben so ca. 15.000 Tags. Da ist das noch OK. Ähnliche Probleme hat auch Google.

Interviewer: Das bezieht sich jetzt auf die Resource-List?

Interviewee: Auf die Tag-Vorschläge und auch auf die „browse tags“. Bei größeren Datenmengen, denke ich, muss man manche Darstellung hinterfragen. Da wird es interessant. Aber im Moment passt alles.

Interviewer: Wie vollständig sind die gelieferten Daten, bzgl. der Resource-API?

Interviewee: Das passt. Da sind die Daten da. Man hat da auch den Use-Case: Finde ich das Ergebnis, das ich erwartet habe. Finde ich das auch. Das haben wir ein paar mal getestet und das funktioniert.

Interviewer: Entsprechen die Rückgabewerte den Erwartungen an die API?

Interviewee: Ja. Also wie gesagt: Man bekommt, das zurück, was man erwartet, wenn man anders an das System rangeht. Derjenige, der das benützt, weiß ja nicht was im System vorhanden ist. Er browst sich somit durch. Wenn man allerdings weiß, welche Inhalte vorhanden sind, und nach diesen sucht, so findet man die auch. Wir haben das auch tatsächlich getestet.

Interviewer: Wie schwierig/einfach sind die UI-Elemente von STAGS zu benutzen?

Interviewee: Für mich sind sie einfach zu verwenden. Wenn man eine ältere Person mit weniger IT-Hintergrund heranzieht, so kann ich das nicht beurteilen. Dazu müsste man eine Anfrage an diese Personengruppe stellen. Junge und IT-affine Personen dürften keine Probleme haben.

Interviewer: Wie schätzen sie die Nützlichkeit von STAGS für den Endanwender ein? Wird die individuelle Produktivität erhöht? Gibt es Lerneffekte für den Nutzer?

Interviewee: Kann ich so nicht sagen. Das müsste ich messen. Dazu fehlen mir leider die Statistiken.

Interviewer: Mir geht es jetzt nicht konkret um Nutzerzahlen. Vielmehr würde ich mich eher für Verbesserungen durch alternative oder bessere UI-Elemente interessieren?

Interviewee: Es ist bei der Tag-Vergabe sicherlich mehr Komfort vorhanden. Und so wenige Tags werden bei uns gar nicht vergeben, wenn man das in Vergleich zu den User-Anzahlen betrachtet. Das hat natürlich eine Vereinfachung zur Konsequenz. Dass man z.B. Tags, die es schon gab wieder vorgeschlagen bekommt. Unser Werkstudent freut sich auch. Dem habe ich ein Tool geschrieben, das ihm die vorhandenen Tags auflistet. Mit dem kann er Tippfehler rausfinden. Dadurch gab es von Betriebsseite her ein Einsparung. Bzgl. der User: Die haben zwar jetzt mehr Komfort. Wie viel Zeit, die jedoch dadurch eingespart haben, kann ich nicht beurteilen.

Interviewer: Welchen erwarteten Nutzen hatten sie?

Interviewee: Unser erwarteter Nutzen war: Mehr Komfort für die User, dass man plattformübergreifende Tags bekommt und dass wir beim Bereinigen der Tags Arbeit einsparen. Die Anforderungen wurden erfüllt.

Interviewer: D.h. der Nutzen ist eingetreten?

Interviewee: Ja, der Nutzen ist eingetreten. Wir haben auch keinen Ärger mit dem Modul. Jedenfalls in dem jetzigen Betriebsmodus läuft das Modul reibungslos und wir haben keine zusätzlichen Wartungsaufwände. Die Erwartungen wurden somit erfüllt.

Interviewer: War der Aufwand vertretbar für den Nutzen?

Interviewee: Ja. Für uns war das eine „low hanging fruit“.

Interviewer: Hat es sich somit gelohnt?

Interviewee: Ja.

Interviewer: Wie sieht der tatsächliche Nutzen aus? Hat STAGS für sie einen Business Impact? Hatten sie Kosteneinsparungen?

Interviewee: Evtl. finden Leute etwas schneller. Das kann man aber so nicht messen. Also eher nicht. Wenn jemand eine gesuchte Information schneller oder überhaupt findet und ein paar Tage Arbeit einspart, dann meldet er mir das nicht. So einfach ist das. So etwas ist allgemein schwer messbar. Das ist auch für eine allgemeine Plattform ein Problem: Zu zeigen, worin der konkrete Nutzen liegt; somit die Kosten eines Services zu rechtfertigen.

Interviewer: Hatten sie konkret als Betreiber Kosteneinsparungen?

Interviewee: Die Tag-Vorschläge wurden verbessert. Dadurch treten weniger Tippfehler auf und wir haben somit weniger Aufwand beim Säubern der Tags. Bei ersten Einsatz von dem Tool, das ich geschrieben habe, waren es so ein paar hundert Tags, die wir bereinigen mussten – also z.B. Tags die falsch geschrieben waren. Ich habe dazu über die Levenshtein-Distanz ähnliche Tags bestimmt und damit potentielle Kandidaten für die Bereinigung gefunden. Ein solches Modul wäre auch eine potentielle Erweiterung von STAGS.

Interviewer: Das sehe ich auch so und ist auch in Teilen schon umgesetzt. Dazu gibt es prototypische Implementierungen, die eben noch in den Produktiv-Status gehoben werden müssen.

Interviewee: Für mich, als Plattformbetreiber, sind Features für den Administrationsbereich natürlich sehr wesentlich. Da sollte man meiner Meinung nach noch nachlegen. Ein Plattformbetreiber will Statistiken und noch mehr Test-APIs. Letztere auch um Tests automatisiert durchführen zu können.

Interviewer: Hat sich die Beliebtheit der Plattform gesteigert?

Interviewee: Ich würde generell sagen. Es gab Verbesserungen bei der Usability. Das Problem ist speziell bei unserer Plattform, dass manche Nutzer, vor allem ältere, noch teilweise mit den Basics zu kämpfen haben. Da sind dann die Erweiterungen durch STAGS eher unwesentlich. So etwas wie STAGS werden die Nutzer dann eher in ein paar Jahren zu schätzen wissen. Im Moment ist eine normale Suche schon eine Herausforderung – um das übertrieben auszudrücken.

Interviewer: Konkret bzgl. der Beliebtheit?

Interviewee: Da kann ich keine signifikante Steigerung feststellen. Seit der Einführung von STAGS kann man keinen wesentlichen Zuwachs feststellen. Ich persönlich halte die Features für wichtig und nützlich. Bzgl. der Nutzer kann man das ohne Usage-Statistiken wenig sagen. Den Mehrwert durch die Tag-Vorschläge schätze ich höher ein als für die „browse tags“-Funktionalität.

Interviewer: Wurde die Usability verbessert?

Interviewee: Ja. Hatte ich ja bereits gesagt. Das ist besser geworden.

Interviewer: Hatten sie alternative Lösungen?

Interviewee: Die Alternative wäre gewesen auf das plattformübergreifende Tagging zu verzichten. Wirkliche andere Alternativen haben wir uns nicht angeschaut. Gibt es so was?

Interviewer: Also mir ist keine Alternative bekannt. Über einiges an Verrenkungen könnte man die Intranet-Suche dazu verwenden. Meiner Meinung gibt es da aber ein paar Stolpersteine und die Umsetzung ist nicht gerade trivial. Ich habe diese Frage nur gestellt, um einen Vergleich zu alternativen Produkten zu erhalten. In diesem Fall ist das aber nicht anwendbar.

Interviewee: Alternativ wäre für uns noch gewesen, selber etwas zu implementieren. So haben wir von ihrer Arbeit profitiert.

Interviewer: Wenn nein, welche Aspekte an STAGS sind alternativlos?

Interviewee: Meiner Meinung nach ist es der richtige Weg, kleine Services zu haben, die ihre Aufgabe gut erfüllen. Damit man von den Silos wegkommt. Ansonsten hätten wir selber etwas umsetzen müssen. Die Funktionalität die STAGS anbietet, hätten wir gebraucht. Dass wir allerdings selber etwas entwickelt hätten, glaube ich weniger. Für uns sind die plattformübergreifenden Tag-Vorschläge der Hauptaspekt. Die Anzeige der Ressourcen ist für uns nachgelagert und hat für uns zumindest im Moment noch nicht *den* Stellenwert.

Interviewer: Dann wäre ich mit meinen Fragen am Schluss. Sie wollten allgemein noch ein paar Bemerkungen machen?

Interviewee: Ich hatte bereits angedeutet, dass wir noch ein paar Features hätten, die für uns wichtig wären: Zum einen die Mandantenfähigkeit. Das andere wäre die Unterscheidung von Content-Töpfen, d.h. dass die Tag-Vorschläge auf Tags zu bestimmten Content-Typen eingeschränkt werden – also beispielweise Projektbeschreibungen. Damit könnte man auch kaskadierende Filter in „browse tags“ integrieren. Wie bereits angesprochen: Die administrativen Tools, d.h. dass man noch mehr an Daten über das System erhalten kann. Das schließt auch Statistiken und bessere Testbarkeit mit ein. Bzgl. des Exports: Der passiert ja im Moment einmal pro Tag. Ich persönlich bin ein Freund event-basierter Ansätze. Das ist an der Stelle eine Philosophie-Frage.

Interviewer: Im Moment gibt es eine Komponente, die über RSS-Feeds sich Aktualisierungen holt. Das ist zwar nicht event-basiert, liefert aber dennoch einigermaßen zeitnahe Ergebnisse. Das Problem ist auch die Verfügbarkeit von Eventbenachrichtigungen durch eine Anwendung. Es gibt im Web etwas genannt „Webhooks“. Das wird allerdings beispielsweise von den aktuellen Confluence-Versionen noch nicht unterstützt.

Interviewee: Genau, das ist meiner Meinung nach die Zukunft. Es reicht langfristig nicht, einmal pro Tag ein Update zu machen. Das Business wird immer schneller. Mit den event-basierten Ansätzen hat man da eine Lösung. Gut, man hat bei manchen Stellen auch wieder anderen Ärger. Man muss sich z.B. um

den Abgleich von den Daten kümmern. Durch die event-basierten Ansätze ist allerdings so manches einfacher zu handhaben. Vor allem ist das wichtig bei riesigen Datenmengen. Da hat man gar nicht die Zeit dazu diese alle zu importieren.

Interviewer: Es gibt auch von Google ein Protokoll mit dem Name PubSubHubbub, das unter Umständen auch relevant sein könnte.

Interviewee: Genau. Was ich mir noch gedacht habe sind so solche Angelegenheiten wie: „recommend content“, „recommend tags“, „recommend user“. Oder auch „recommend tags“ bzgl. eines Contents. D.h. man scannt den Inhalt und schlägt dazu relevante Tags vor.

Interviewer: Gewisse Funktionalitäten, die sie hier erwähnen, sind bereits produktiv. Z.B. sind im TechnoWeb Tagvorschläge basierend auf einem Volltext bereits integriert.

Interviewee: Eine Unterstützung des Taggings bei gewissen Standardvorgängen, wo beispielsweise eine Dokumentation entstehen soll, wäre auch noch interessant. Z.B. „lessons learned“ -Seiten sollten immer eine gewisse Reihe an Standardtagtypen aufweisen, also z.B. ein Tag, das ein Projekt referenziert. Für solche Fälle wäre eine Unterstützung noch interessant. Auch werden des Öfteren von manchen Leuten immer dieselben Tags miteinander verwendet, was dazu führt dass manche Tags mit wenigen unterschiedlichen Tags verwendet werden. Eine Suche nach Dokumenten über die Kombination von immer spezifischeren Tags ist da erschwert. Dazu könnte man sich auch noch etwas überlegen. Bei den Tag-Vorschlägen könnte man dem User, der unter Umständen gar nicht weiß, welche Tags er zur genaueren Einschränkung verwenden soll, noch mehr unterstützen. Das eine wäre eben Content-basierte Vorschläge, was es anscheinend schon gibt. Das andere wären eben alternative Auswahllisten. Aber generell was Social-Services betrifft: Es wird immer wichtiger viele kleine Tools zu haben, die ihre Sache gut machen, als monolithische Silo-Applikationen. Es ist wichtig, dass solche Tools plattformübergreifend sind. Bei den ganzen Social-Media-Themen handelt es sich um Querschnittsfunktionalitäten. Ich will ja einen Aktivitäten-Strom nicht nur von der einen Plattform, in der ich mich gerade befinde. Ich will einen umfassenden Aktivitäten-Strom. Da ist man mit STAGS auf dem richtigen Weg – in dem Sinne, wo die Reise hingehen muss mit internen Tools.

A.3. Wikisphere And Landing Page Wikisphere in Global Intranet Portal

The interview has not been transcribed literally. The interview has been checked for correctness by the interviewed person. The interview has been conducted on December 6th,

2012 at 1.00 - 1.45 pm with Mr. B. L., a senior software developer and project manager at Siemens Corporate Technologies.

Interviewer: Welche Rolle haben sie in Bezug auf ihre Applikation? Sind sie Project Manager oder Programmierer?

Interviewee: In diesem Fall beides. Für die Wikisphere bin ich sowohl Application-Manager als auch Entwickler. Für die spezielle Applikation (nicht die Wikisphere) war ich eher ein Solution-Designer.

Interviewer: Sind sie auch Endanwender in täglicher Arbeit und haben somit als normaler Anwender Kontakt mit STAGS?

Interviewee: Eher nicht so. Als Endanwender bin ich eher betroffen, als dass ich die Übersichtsseite des Öfteren besucht habe, um die „hot topics“ herauszufinden.

Interviewer: Welche Software-Entwicklungserfahrung haben sie?

Interviewee: Ein umfangreiche. So 12 Jahre.

Interviewer: An welchen Stellen wurde STAGS eingebaut?

Interviewee: Im Siemens Global Intranet auf einer Landing-Page für die Wikisphere. Der Datenexport aus der Wikisphere erfolgt über ein Confluence-Plugin. Darüber kenne ich auch die andere Seite von STAGS, also die technische.

Interviewer: Hatten sie Kontakt mit der STAGS REST API?

Interviewee: Nicht als Entwickler sondern als „Vermittler“. Ich habe die Applikation der ausführenden Agentur erklärt.

Interviewer: Welche Features von STAGS nutzen Sie?

Interviewee: Die Tag-Cloud-Darstellung gefiltert auf eine Applikation und einem zeitlichen Ausschnitt.

Interviewer: Wie hoch war der Aufwand bzgl. der Zeit zur Einbindung von STAGS in ihre Applikation?

Interviewee: So ca. 2 Tage. Es handelt sich dabei eine Schätzung, die müsste so ungefähr hingehen.

Interviewer: Wie schätzen sie die Schwierigkeit der Integration in ihre Applikation ein?

Interviewee: Sehr leicht für den speziellen Kontext. Bei komplexeren Applikationen kann es schon meiner Meinung nach „mittelleicht“ werden.

Interviewer: Wie schwierig ist die API zu verstehen?

Interviewee: Einfach.

Interviewer: Wie schwierig ist die API zu benutzen, d.h. nachdem man sie verstanden hat?

Interviewee: Einfach. Mit Grundlegenden HTML- und CSS-Kenntnissen. JavaScript und natürlich JSON sind auch Vorrausetzung.

Interviewer: Wie würden sie die Flexibilität der STAGS API einschätzen?

Interviewee: Sehr flexibel. Auch die nachträgliche Erweiterung ist möglich.

Interviewer: Wie würden sie die Anpassbarkeit der bereitgestellten Widgets beurteilen?

Interviewee: Ich kann da nichts Konkretes dazu sagen – mangels Erfahrung. Ich denke allerdings, dass eine Anpassung ohne große Schwierigkeiten möglich sein dürfte.

Interviewer: Wie hoch schätzen sie die Verfügbarkeit von STAGS ein?

Interviewee: So 95 % würde ich sagen. Es gibt immer wieder kleinere Ausfälle, die den Server betreffen.

Interviewer: Ist die Verfügbarkeit somit ausreichend für sie?

Interviewee: Nein, als Corporate-Service würde man sich eine höhere Verfügbarkeit wünschen. Das liegt aber nicht unbedingt an der Applikation selber, sondern teilweise auch am Hosting.

Interviewer: Entspricht die Performance von STAGS ihren Bedürfnissen?

Interviewee: Ja, STAGS ist „rasend“ schnell.

Interviewer: Wie schätzen sie die Zuverlässigkeit von STAGS ein?

Interviewee: Wenn STAGS läuft, ist der Service sehr zuverlässig.

Interviewer: Ist die Zuverlässigkeit ausreichend?

Interviewee: Ja, in diesem Fall schon.

Interviewer: Wie relevant sind die gelieferten Daten der unterschiedlichen API-Elemente?

Interviewee: Für meinen Fall sind die Daten relevant. Die Seite dient als Teaser. Die angezeigten Daten sind sehr relevant.

Interviewer: Wie vollständig sind die gelieferten Daten, z.B. bzgl. der Resource-API?

Interviewee: Die Daten sind vollständig entsprechend der Erwartungshaltung.

Interviewer: Entsprechen die Rückgabewerte den Erwartungen an die API?

Interviewee: Ja, damit beantwortet.

Interviewer: Wie schwierig/einfach sind die UI-Elemente von STAGS zu benutzen?

Interviewee: Das kann ich nicht einschätzen, da ich die UI-Elemente zu wenig kenne.

Interviewer: Wie schätzen sie die Nützlichkeit von STAGS für den Endanwender ein? Wird die individuelle Produktivität erhöht? Gibt es Lerneffekte für den Nutzer?

Interviewee: In dem speziellen Anwendungsfall ist das für den Endanwender durchaus nützlich, weil somit eine Einstiegsseite lebendiger gestalten kann. Es ist somit mehr Dynamik enthalten, statt nur statischen Content. Für den TechnoWeb-Cases bzgl. des Resource-Browsing, den ich als Endanwender kenne, würde ich STAGS definitiv als nützlich einschätzen. Von den Möglichkeiten und der Darstellung, glaube ich, dass es wirklich State of the Art ist. Die Produktivität wird wahrscheinlich etwas erhöht. Eine gute Volltext-Suche wird wahrscheinlich ähnliches liefern, wie gesagt bei einer „guten“ , bzgl. dem Finden von Ressourcen. Das ist allerdings eine Frage

des Maßstabs. Aus Endanwendersicht, also nicht für mich als jemand der Social-Media-affin ist, sind durchaus gewisse Aha-Effekte zu erwarten.

Interviewer: Welchen erwarteten Nutzen hatten sie?

Interviewee: STAGS aggregiert ja plattformübergreifend Daten. Der Benefit für mich als jemand der die Wikisphere verantwortet, dass die Inhalte der Wikisphere an verschiedenen Stellen angezeigt werden können. Dadurch erhöht sich die Sichtbarkeit und es entsteht somit ein sehr hoher Nutzen.

Interviewer: Ist der Nutzen eingetreten?

Interviewee: Ja. Ich kann an statistischen Auswertungen nachvollziehen, dass anderswo gelistete Inhalte häufiger aufgerufen werden.

Interviewer: War der Aufwand vertretbar für den Nutzen?

Interviewee: Ja, war er.

Interviewer: Hat es sich somit gelohnt?

Interviewee: Ja.

Interviewer: Wie sieht der tatsächliche Nutzen aus?

Interviewee: Der tatsächliche Nutzen, ist durch die erwähnten Zugriffszahlen eingetreten. Die Wikisphere hat auch eine höhere Sichtbarkeit, wenn sie beispielsweise als Filter irgendwo auftaucht.

Interviewer: Hat STAGS für sie einen Business Impact?

Interviewee: Die Inhalte der Wikisphere werden über STAGS weiter verbreitet. Dadurch wird das Teilen von Wissen erleichtert und das ist die ureigentliche Aufgabe oder das Ziel der Plattform.

Interviewer: Hatten sie Kosteneinsparungen?

Interviewee: Wenn ich z.B. die Tag-Cloud-API zur Einbettung im Global Intranet nicht von STAGS nutzen könnte, hätte ich die Implementierung selber durchführen müssen. Und ich habe somit im gewissen Sinne eine Einsparung.

Interviewer: Hat sich die Beliebtheit der Plattform gesteigert?

Interviewee: Ja, durch die höhere Sichtbarkeit.

Interviewer: Wurde die Usability verbessert?

Interviewee: Nein, da im Moment in der Wikisphere noch keine UI-Elemente von STAGS genutzt werden. Bei der Landing-Page, kann man nicht von einer Verbesserung der Usability sprechen. Es handelt sich schließlich nur um einen Teaser.

Interviewer: Hatten sie alternative Lösungen?

Interviewee: Nicht zur Integration über eine API.

Interviewer: Wenn ja, welche und warum haben sie sich gegen diese entschieden? Wenn nein, welche Aspekte an STAGS sind alternativlos?

Interviewee: Die plattformübergreifende Aggregation von Inhalten. Ein großer Vorteil, liegt darin, dass man die Inhalte auch anonym erhalten kann. Das ist Zugleich eine Einschränkung, da dadurch die Anbindung von „geschlossenen Datentöpfen“ nicht möglich ist.

A.4. TechnoWeb

The interview has not been transcribed literally. The interview has been checked for correctness by the interviewed person. This interview has been conducted on December 5th, 2012 at 11.00 - 11.35 am with Mr. T. M., a Senior Project Manager at Siemens Corporate Technologies.

Interviewer: Welche Rolle haben sie in Bezug auf ihre Applikation? Sind sie Project Manager oder Programmierer?

Interviewee: Eher Project Manager.

Interviewer: Sind sie auch Endanwender in täglicher Arbeit und haben somit als normaler Anwender Kontakt mit STAGS?

Interviewee: Als normaler Anwender habe ich über TechnoWeb Kontakt.

Interviewer: Welche Software-Entwicklungserfahrung haben sie?

Interviewee: Ich habe mehrere Jahre Software-Entwicklungserfahrung in Java und C++ im Bereich Radiologie-Informationssysteme – bei Siemens hauptsächlich (2006 - 2010). Davor während des Studiums, und bei Infineon im Rahmen meiner Diplomarbeit hatte ich Kontakt mit Matlab.

Interviewer: An welchen Stellen wurde STAGS eingebaut?

Interviewee: Wir nutzen STAGS beim Urgent Request Channeling, für diverse Tag-Clouds, für Tag-Suggestions zu einem Inhalt und einem Titel und für das Auto-Completion, bei dem man ein Tag eingeben will und bereits zu den ersten Buchstaben Vorschläge in einer Drop-Down-Liste bekommt. Das sind die Stellen, die mir einfallen. Bei der tag-basierten Search bin ich mir nicht sicher, ob wir das ausschließlich über unsere Datenbank machen oder ob da STAGS involviert ist.

Interviewer: Es gibt an der Stelle die Möglichkeit externe System miteinzubeziehen. Das erfolgt ebenfalls mittels STAGS. Da wird die Resource-API von STAGS über Sever-to-Server-Kommunikation verwendet, um Treffer außerhalb von TechnoWeb miteinmischen zu können.

Interviewee: Ah, OK.

Interviewer: Das nur der Vollständigkeit halber.

Interviewer: Hatten sie Kontakt mit der STAGS REST API?

Interviewee: Oberflächlich, aber ich habe sie nicht in der Form eines Softwareentwickler verwendet. Ich habe die API mir angesehen. Ich habe über sie diskutiert. Also, ich hatte mit ihr zu tun.

Interviewer: Welche Features von STAGS nutzen Sie? Das wurde eigentlich vorher schon beantwortet. Interessant ist aber noch, an welchen Stellen im TechnoWeb STAGS eingebaut wurde.

Interviewee: Wir haben beim Dashboard eines Benutzers eine Tag-Cloud. Es gibt bei der Tag-Search eine Tag-Cloud. Es gibt bei der Profilseite eines Nutzers eine individuelle Tag-Cloud. Immer wenn man ein Item erstellt, also Requests, News, Diskussion, Poll oder Urgent Request, gibt es die Möglichkeit zu taggen. In diesem Rahmen wird STAGS verwendet. Die Verwendung von STAGS ist sehr breitgestreut und STAGS ist an sehr vielen Stellen integriert.

Interviewer: Wie hoch war der Aufwand bzgl. der Zeit zur Einbindung von STAGS in ihre Applikation?

Interviewee: Relativ gering. Pro Feature würde ich sagen im Stundenbereich. Wir nutzen viele Features. Somit wird der Aufwand im Bereich von Tagen liegen. In Summe ist der Aufwand aber nicht hoch.

Interviewer: Wie schätzen sie die Schwierigkeit der Integration in ihre Applikation ein?

Interviewee: Geringe Schwierigkeit.

Interviewer: Wie schwierig ist die API zu verstehen?

Interviewee: Mit „Hausverstand“ kann man die API schon verstehen.

Interviewer: Wie schwierig ist die API zu benutzen, d.h. nachdem man sie verstanden hat?

Interviewee: Wenn man die API verstanden hat, dann ist die Benutzung leicht. Die API orientiert sich an gängigen Standards, bzw. Konventionen. Wenn man diese verstanden hat, dann ist die Nutzung der API einfach.

Interviewer: Wie würden sie die Flexibilität der STAGS API einschätzen?

Interviewee: Man kann die API recht flexibel einsetzen. Gewisse Teile der API haben einen speziellen Anwendungsfall für den sie geschaffen wurden. Es macht da inhaltlich keinen Sinn diese anders zu verwenden. Die API ist nicht hundert Prozent auf Flexibilität ausgerichtet sondern hat einen bestimmten Zweck zu erfüllen. Potentiell ist die API dennoch sehr flexibel, sie wurde aber an manchen Stellen für recht konkrete Anwendungsfälle zugeschnitten und hat an diesen Stellen nicht allzu viele freiwählbare Parameter, die großartig viel am Verhalten ändern würden. Das ist allerdings nichts Schlechtes. Das ergibt sich dort aus der Natur der Sache.

Interviewer: Wie würden sie die Anpassbarkeit der bereitgestellten Widgets beurteilen?

Interviewee: Sehr leicht. Also wir haben STAGS in TechnoWeb integriert und es fällt nicht auf, dass da was Fremdes ist.

Interviewer: Wie hoch schätzen sie die Verfügbarkeit von STAGS ein?

Interviewee: Relativ hoch, könnte aber besser sein. Es gab hin und wieder mal ein Problem mit der Datenbank. Auch war einmal ein administrativ-technisches Problem. Das ist aber nicht direkt ein Fehler von STAGS. In der Regel ist ein kurzer Ausfall auch nicht so schlimm. Ein einziges Problem ist, wenn

gerade ein Urgent Request versendet werden soll und da STAGS nicht verfügbar ist und somit dieser nicht versendet wird. Daher würde ich sagen die Verfügbarkeit ist gut, könnte aber besser sein.

Interviewer: Was würden sie als geschätzten Prozentwert meinen?

Interviewee: So 99 % ist die Verfügbarkeit. Sie könnte aber ruhig 99,9 % sein.

Interviewer: Ist die Verfügbarkeit somit ausreichend für sie?

Interviewee: Generell ist sie ausreichend. Nur beim Urgent Request Channeling, wo ein Ausfall gravierender ist, wäre eine höhere Verfügbarkeit schon besser.

Interviewer: Entspricht die Performance von STAGS ihren Bedürfnissen?

Interviewee: Ja, die ist super. Die ist in Ordnung. Die Performance wird immer schneller und besser. So wie ich das sehe ist die völlig ausreichend.

Interviewer: Wie schätzen sie die Zuverlässigkeit von STAGS ein?

Interviewee: Die ist sehr hoch. Mir wäre noch nie aufgefallen, dass wenn es läuft, ein „Blödsinn“ rausgekommen wäre.

Interviewer: Ist die Zuverlässigkeit ausreichend?

Interviewee: Sie ist hoch und ausreichend. Es ist durchaus möglich, dass irgendwo ein kleiner Bug ist. Das ist unter Umständen auch schwierig nachzuvollziehen. Generell kommt das raus, was man sich erwartet. Die Schnittstelle liefert, das was sie liefern soll. Inhaltlich kann es sein, dass man sich manchmal leicht etwas anderes erwartet hätte.

Interviewer: Wie relevant sind die gelieferten Daten der unterschiedlichen API-Elemente?

Interviewee: Für das Urgent-Request-Channeling sind die Daten sehr gut, sehr relevant und brauchbar. Genauso wie die Recommendations of Network Members. Die ist auch sehr gut. Die Tag-Suggestions, die auf den Inhalten und den Titeln eines Items basieren, die könnten besser sein. Und zwar liegt das meiner Meinung nach daran, dass der Tag-Corpus nicht bereinigt, bzw. gewartet, wird und dann ab und zu Tags mit Tippfehlern auftauchen, bzw. ab zu Tags, die keinen Wert haben, vorkommen und die auch vorgeschlagen werden. Der Automatismus im Hintergrund, der Tags-Suggestion auswählt, wählt sie wahrscheinlich auf Häufigkeiten und Ähnlichem aus. Der erkennt ja nicht, ob ein Tag inhaltlich schlecht ist. Ab und zu wird dann das richtige Tag neben dem mit dem Tippfehler vorgeschlagen. Meiner Meinung nach wäre es dann perfekt, wenn es irgendwie einen manuellen Weg gäbe, das zu bereinigen – dass Tag-Suggestions mehr nur saubere Tags vorschlagen, also dass man quasi „böse“ Tags oder wie man die auch nennen mag, als „böse“ markiert und die in den Tag-Suggestions nicht mehr auftauchen. Damit soll verhindert werden, dass sich solche Tags dann weiter verbreiten; dass diese nicht vorgeschlagen werden. Ein Nutzer clickt auf ein Tag und liest es vielleicht nicht gescheit. Somit hat man dann ein Tag mit schlechter Qualität.

Interviewer: Und die anderen beiden Features, also die Tag-Suggestions in dem Drop-Down-Menu und die tag-basierte Suche?

Interviewee: Die passen. Die sind ja recht „straight forward“ in den Implementierungen. Allerdings werden bei den Vorschlägen auch Tags mit Tippfehlern vorgeschlagen – wenn man beim Tippen einen Tippfehler macht.

Interviewer: Ja, das ist technisch bedingt. Da könnte man evtl. noch etwas verbessern. Wie sie gesagt haben, sollte man da noch manuell, was machen können.

Interviewee: Das wollte ich damit sagen. Mit dem was es liefert, bin ich sehr zufrieden. Der Algorithmus, die Art und Weise, wie er arbeitet, ist in allen Fällen gut genug, d.h. ausreichend gut, so wie es jetzt ist – natürlich kann technisch immer, was optimiert werden. Nur wäre es gut, wenn man eine manuelle Möglichkeit hätte den Tag-Corpus zu bereinigen. Und wenn man auch nur zehn Minuten in der Woche verbringt. Da schafft man schon die gröbsten Dinge, die einem auffallen, loszuwerden. Aber so etwas fehlt, habe ich den Eindruck.

Interviewer: Das sind Überlegungen, die ich hatte. Dafür habe ich auch prototypische Implementierungen. Allerdings ist das noch nicht in einen Produktivstand überführt. Wie vollständig sind die gelieferten Daten, bzgl. der Resource-API?

Interviewee: Ja, ich denke die sind vollständig. So vollständig wie sie sein können, wenn man bedenkt, dass man einmal pro Tag die Daten abgleicht. Es kann sein, dass man neue Ressourcen, die noch nicht exportiert worden sind, nicht findet. Das ist aber auch klar, und das ist auch akzeptabel.

Interviewer: Entsprechen die Rückgabewerte den Erwartungen an die API?

Interviewee: Das tun sie mit Ausnahme der besagten Tippfehler bei den Tags. Aber ansonsten ist immer das zurückgekommen, was man sich erwartet hat.

Interviewer: Wie schwierig/einfach sind die UI-Elemente von STAGS zu benutzen?

Interviewee: Sehr leicht. Die sind ganz einfach zu verstehen.

Interviewer: Wie schätzen sie die Nützlichkeit von STAGS für den Endanwender ein? Wird die individuelle Produktivität erhöht? Gibt es Lerneffekte für den Nutzer?

Interviewee: Ich bin der Meinung, dass die Produktivität erhöht wird und dass es sehr nützlich ist für den Benutzer. In manchen Fällen hängt es natürlich davon ab, für was man es verwendet und über welches Feature man redet. Aber generell finde ich die Sachen nützlich und brauchbar. Ein Beispiel ist das Follow-Tag im TechnoWeb. Da ist es sehr nützlich, dass man über bestimmte Themen am laufenden gehalten wird – über Sachen, die man ansonsten vielleicht einfach übersehen würde. Wo einem Wissen fehlt für eine gute Lösung, hatte man über die Mechanismen ausreichend Wissen für eine bessere Lösung. Die Lerneffekte sind insofern da: Wenn man die Tags visualisiert und der Nutzen der Tags klarmacht, dass man über Tags Dinge wiederfinden kann, dann lernt das der Nutzer eben und versteht besser mit den Tool umzugehen. Ein anderer Lerneffekt ist eben, dass der Nutzer herausfindet, dass er über die Tags zu bestimmten Themen informiert bleiben

kann. Das hängt auch davon ab, wie man Lerneffekt definiert. Das kann man auch so sehen, dass wenn ein Nutzer zu bestimmten Themen informiert bleibt, eben auf dem Gebiet, was dazulernt.

Interviewer: Welchen erwarteten Nutzen hatten sie?

Interviewee: Das ist für mich persönlich schwierig zu beantworten, da vieles schon vor meiner Zeit gestartet ist. Ich denke der erwartete Nutzen damals war, dass man ein Tagging-Framework für mehrere Applikationen hat, dass man es schafft applikationsübergreifend Informationen auszutauschen – was bzgl. der Wikisphere und Blogosphere sehr gut funktioniert, auch über den Follow-Tag-Mechanismus.

Interviewer: Und wie schaut es aus mit so Themen, wie Urgent-Request-Channeling?

Interviewee: Urgent-Request-Channeling ist wieder ein eigenes, aber gutes Thema. Man hat sich da ja erwartet, dass man über die Einbeziehung des Tagging-Frameworks in das Urgent-Request-Channeling, wie wir das ja jetzt einsetzen, sich sehr viele Emails, sehr viele Notifications, erspart – dass man die richtigen Leute erreicht, ohne dabei die falschen zu vergraulen. Man hat sich das erwartet, bevor man mit der Umsetzung begonnen hat. Das ist dann auch erfüllt worden. Es werden ungefähr dieselbe Anzahl an Urgent Request beantwortet wie früher, nur ist die Anzahl der verschickten Emails ein Bruchteil von früher – sowohl absolut als auch relativ bzgl. der Anzahl der verschickten Emails. Was das angeht, war da zwar ein gewisser Aufwand dahinter, das zu entwickeln, aber es hat sich definitiv gelohnt. So wie das früher mit den neun Kategorien war, das war nicht skalierbar. Das hat funktioniert mit zwei/drei tausend Leuten. Das funktioniert aber mit über zwanzig tausend Leuten nicht. Wenn dann jeder Urgent Request an über zwanzig tausend Leute verschickt wird, das würde nicht funktionieren. Also was das angeht, hat STAGS einen wertvollen Beitrag geliefert.

Interviewer: Ist der Nutzen eingetreten? Das haben sie eigentlich schon beantwortet.

Interviewee: Der ist definitiv eingetreten.

Interviewer: War der Aufwand vertretbar für den Nutzen? Das ist eigentlich auch schon beantwortet.

Interviewee: Der Aufwand war vertretbar

Interviewer: Dasselbe gilt, ob es sich somit gelohnt hat?

Interviewee: Ja.

Interviewer: Wie sieht der tatsächliche Nutzen aus? Hat STAGS für sie einen Business-Impact?

Interviewee: Für Siemens hat es definitiv einen Business-Impact. Für uns als Applikation hat es insofern keinen Business-Impact, da wir kein Business haben. Wir werden finanziert, aber die Benutzer machen das gratis. Wenn die Nutzer für einen Urgent-Request zahlen müssten, aus welchem Grund auch immer, dann hätten wir so was wie einen Business-Impact. Also es lässt sich die Frage schwer beantworten.

Interviewer: Hatten sie Kosteneinsparungen?

Interviewee: In demselben Sinne hatten wir auch keine Kosteneinsparungen, weil z.B. ich so oder so in Vollzeit beauftragt bin. Jetzt ist das zwar so, dass ich weniger Aufwand habe Leute zu beschwichtigen, die sich über SPAM-Mails beschweren. Anderseits führt das nicht dazu, dass ich weniger beauftragt werde oder arbeite. Es spart nicht Kosten. Es spart Zeit. Sagen wir mal so.

Interviewer: OK, d.h. das würde man dann als Effizienzsteigerung betrachten. Sie können mehr in ihrer Zeit erledigen.

Interviewee: Genau.

Interviewer: Hat sich die Beliebtheit der Plattform gesteigert?

Interviewee: Ich denke schon. Ich denke, dass für manche das schon ein Frustpotential war, mit Urgent-Requests zugespammt zu werden. Der Umkehrschluss wäre, wenn das nicht mehr so passiert und es passiert nicht mehr so, dass dann die Beliebtheit steigt. Und auch die Angelegenheiten, die das Tagging betreffen, sind ja schön präsentiert – auch bzgl. der Usability. Die Usability ist definitiv verbessert worden. Z.b. Tag-Suggestions: Jemand hat einen Inhalt geschrieben, also Titel und Text. Es werden passende Tags vorgeschlagen. Dann ist das definitiv eine Usability-Verbesserung. Die Leute müssen nicht mehr großartig tippen sondern können einfach clicken. Das tun sie gern und lieber und das macht es das Tool auch leichter zu verwenden. Zum einen wird die Beliebtheit gesteigert zum anderen die Usability. Das ist also ein positiver Effekt.

Interviewer: Usability und Beliebtheit sind meisten ja korreliert. Die beiden Themen kann man nicht unbedingt so einfach trennen. Wurde die Usability verbessert? Das haben sie eh schon positiv beantwortet.

Interviewer: Hatten sie alternative Lösungen?

Interviewee: Nachdem das vor meiner Zeit war, kann ich das nicht so genau sagen. Ich denke mal, eine Alternative wäre gewesen, alles selber zu stricken. Das kann man immer machen. Ich vermute, dass das eine Alternative war, die im Raum gestanden ist. Ich denke mal man hat sich deshalb dagegen entschieden: Wenn jede Plattform das für sich selber strickt, hat man wieder keine Synergie-Effekte – wenn man das so sagen will. So wie wir jetzt mit dem Tagging-Framework, als gemeinsamen Service, die Wikisphere, Blogosphere, C4C, ReferencesPlus, was auch immer miteinander verbinden, das hat einfach Vorteile, die man erhält, im Unterschied dazu, wenn man alles selber macht. Als bereichsübergreifender Service ist es unschlagbar. Und nachdem es ein „Siemens-Lösung“ , also keine zugekaufte Lösung ist, hat man sehr leicht die Möglichkeit etwas anzupassen – sofern es notwendig ist, man kann sehr leicht die Schnittstelle erweitern. Das sind auch wieder Vorteile, die dafür sprechen.

Interviewer: D.h. die zwei Aspekte: Plattformübergreifender Service und Anpassbarkeit?

Interviewee: Ja, es ist auch keine kommerzielle Lösung. Bei einer kommerziellen Lösung gibt es auch wieder Angelegenheiten, wie ein Vertrag. Auch wären Anpassungen wieder schwieriger. Man müsste sich auch wieder um Lizenzen für jede Plattform kümmern. Das sind so Sachen, die man nicht so leicht lösen kann.

A.5. References+

This interview is not a literal transcript. The interview has been checked for correctness by the interviewed person. This interview has been conducted on November 30th, 2012 at 10.00-10.40 am with Dr. J. M., Senior Manager Knowledge Management at Siemens Building Technologies.

Interviewer: Welche Rolle haben Sie in Bezug auf ihre Applikation? Sind Sie Project Manager oder Programmierer?

Interviewee: Im bin im Endeffekt alles, also für beides zuständig. Ich habe die Gesamtverantwortung für References+. Das betrifft sowohl die IT als auch Themen wie Content-Strukturierung und eine Moderationsfunktion für die Community. Allerdings habe ich auch weite Teile der Software – so ca. 80-90 % — selbst programmiert. Das mache ich in Personalunion.

Interviewer: Sind Sie auch Endanwender in täglicher Arbeit und haben somit als normaler Anwender Kontakt mit STAGS?

Interviewee: Die Endanwender sind die Nutzer. Aber ich nutze das System natürlich selbst auch. STAGS wurde ja im Microblog-Bereich eingebaut. Ich blogge auch gerne und viel. Also jedes Mal wenn ich einen Micropost schreibe, dann verwende ich STAGS. Ich sehe die Tag-Cloud in References+ und benutze die Tag-Vervollständigung. Ich sehe auch die Tag-Clouds in anderen Widgets. Das sind Boxen, wie man Sie von Facebook oder Twitter im Internet kennt. Diese werden in anderen Intranet-Seiten eingebunden. Dabei werden Blogpostings kontextabhängig passend zu einer Intranetseite dargestellt. Das kann der Seiten-Admin einstellen. Das funktioniert über WCMS-Applikation-Funktionalität oder eine spezielle SharePoint-Funktion oder, wenn es gar nicht anders klappt, über einen IFrame. Dazu wird auch eine Tag-Cloud angezeigt, passend zu dem jeweiligen Intranet-Seiten-Thema, also auch außerhalb von References+. Diese Funktionalität stammt ebenfalls von STAGS. Die Tag-Clouds kommen in References+ von STAGS, sei es für alle Blog-Postings oder sei es für die Blogpostings zu einem bestimmten Thema, sprich die Co-Tags zu einem bestimmten Tag. Auch gibt es eine Ansicht mit Blogposts von einem gewissen Nutzer mit gefilterter Tag-Cloud. Wie gesagt, die Tag-Cloud taucht auch auf anderen Intranet-Seiten auf. Allerdings wird dabei alles von References+ als Web-Service zur Verfügung gestellt.

Interviewer: Sie sind somit also auch Endanwender?

Interviewee: Ja, bin ich.

Interviewer: Welche Software-Entwicklungserfahrung haben sie?

Interviewee: References+ ist in ASP, also Classic ASP, programmiert mit Anbindung an eine MS SQL-Datenbank. Ich selbst habe auch noch andere Erfahrungen, d.h. früher habe ich was in C und C++ gemacht. Privat habe ich eine Webanwendung in PHP und MySQL geschrieben. Ein bisschen Java-Ahnung habe ich auch noch.

Interviewer: An welchen Stellen wurde STAGS eingebaut?

Interviewee: Ich fasse nochmals zusammen: Ein Auto-Completion-Field kommt bei der Eingabe neuer Blogpostings. Für Blogpostings in References+ ist es zwingend erforderlich mindestens einen Tag anzugeben. Tags sind somit nicht wie bei Twitter optional. Bei References+ muss man taggen. Es wird eine Tag-Cloud angezeigt in verschiedenen Filtermechanismus, wie ich bereits erwähnt habe. Und es kommt eine Tag-Cloud in verschiedenen Intranet-Seiten vor, sofern ein Widget von References+ eingebaut wurde.

Interviewer: D.h. Sie verwenden die Widgets, die als JavaScript-Libraries zur Verfügung gestellt werden?

Interviewee: Genau, ich binde die JavaScript-Files ein und verwende die verschiedenen Features.

Interviewer: Hatten Sie Kontakt mit der STAGS REST API?

Interviewee: Nein, ich benutze nur die JavaScript-Libraries und nebenbei erwähnt: Es kommt in verschiedenen Layout-Kontexten im Intranet vor und es schaut überall ansprechend aus.

Interviewer: Welche Features von STAGS nutzen Sie?

Interviewee: Die Tag-Cloud und die Tag-Auto-Completions.

Interviewer: Wie hoch war der Aufwand bzgl. der Zeit zur Einbindung von STAGS in ihre Applikation?

Interviewee: Schwierig zu sagen. Alles in allem werde ich mich schon ein Woche damit beschäftigt haben. Das besteht auch aus dem Verstehen der Konzepte dahinter und den Überlegungen für welche Use-Cases STAGS am besten eingesetzt werden kann. Natürlich musste ich noch selbst die Sachen implementieren. Ich hatte auch noch ein paar kleine Anpassungswünsche. Also insgesamt so ca. eine Woche bei mir und eine Woche bei Siemens CT.

Interviewer: Wie schätzen Sie die Schwierigkeit der Integration in ihre Applikation ein?

Interviewee: Die Einbindung, die der Programmierer vornimmt, um STAGS zu nutzen?

Interviewer: Ja.

Interviewee: Die Einbindung an und für sich ist sehr einfach. Voraussetzung ist jedoch, die nicht ganz vollständige Dokumentation zu verstehen. Ich habe dazu ein paar Vorschläge gemacht. Mittlerweile, glaube ich, ist das besser. Ich würde somit sagen: leicht bis mittel. Ganz trivial ist mir das am Anfang nicht erschienen. Es ist sicherlich nicht sehr kompliziert, das einzubauen.

Man bindet das JavaScript ein und nutzt ein paar Funktionen. Also ich finde das sehr straight-forward.

Interviewer: Wie schwierig ist die API zu verstehen?

Interviewee: Der Einbau ist leicht und die API ist mittel. Das ist jetzt meine subjektive Einschätzung.

Interviewer: Wie schwierig ist die API zu benutzen, d.h. nachdem man Sie verstanden hat?

Interviewee: Wenn man die API verstanden hat, ist die Benutzung wirklich sehr einfach.

Interviewer: Wie würden Sie die Flexibilität der STAGS API einschätzen?

Interviewee: Da wüsste ich auf Anhieb keinen Verbesserungsvorschlag. Man kann ganz viele Parameter einstellen. Man kann nach Nutzern filtern, nach Tags, nach Co-Tags, nach Systemen. Ich kann sogar die Tags auf eine Auswahl bestimmter Systeme einschränken, also beispielsweise References+ und TechnoWeb. Ich wüsste nichts, was mir dazu fehlen würde. Ich finde, das ist sehr flexibel einstellbar.

Interviewer: Wie würden Sie die Anpassbarkeit der bereitgestellten Widgets beurteilen?

Interviewee: Ich habe mir die CSS-Klassen angeschaut und diese den Standards der Seiten angepasst. Ich weiß nicht, ob das so richtig ist.

Interviewer: Das ist genau der gedachte und richtige Weg.

Interviewee: Bei der Tag-Cloud gibt es eine gewisse Anzahl an Klassen. Ich habe die entsprechend angepasst. Das dürfte so zehn Minuten gedauert haben, also überhaupt kein Problem. Das habe ich einfach selber gemacht und das funktioniert bestens. Ich finde das sehr gut. Die eine Applikation will z.B. die Sachen in blau, die andere in grün und die nächste in einer anderen Farbe. Ich habe z.B. in einer Applikation weniger Platz und die Größe der Tags in der Tag-Cloud zwischen der vierten und fünften Klasse gleichgesetzt, um in das Layout zu passen. Da war die Dynamik ein bisschen zu stark und da machte eine solche Änderung Sinn. Ich finde die Verwendung der CSS-Klassen hier sehr gut.

Interviewer: Wie hoch schätzen Sie die Verfügbarkeit von STAGS ein?

Interviewee: 99,9 % ist meine Erfahrung. D.h. ich habe so gut wie keine Downtime feststellen können.

Interviewer: Ist die Verfügbarkeit somit ausreichend für sie?

Interviewee: Ich finde hier „ausreichend“ nicht sehr passend. Das klingt nach Schulnote „vier“ . Die Verfügbarkeit empfinde ich als sehr gut.

Interviewer: D.h. die Verfügbarkeit ist somit zu ihrer Zufriedenheit?

Interviewee: Ja, 100 %. Zur vollsten Zufriedenheit. Ein Server fällt schon mal aus, aber da kann ja die Applikation nichts dafür. Es gab ja mal einen kleinen Netzausfall vor kurzem.

Interviewer: Entspricht die Performance von STAGS ihren Bedürfnissen?

Interviewee: Zur vollsten Zufriedenheit. Das geht sehr performant. Die Seite wird aufgebaut und sobald die Seite steht, ist die Tag-Cloud da.

Interviewer: Wie schätzen Sie die Zuverlässigkeit von STAGS ein?

Interviewee: Mir ist noch nie ein Fehler aufgefallen. D.h. ich würde sagen bei 100 %. Ich verwende das System täglich. Ich blogge auch und ich konnte noch nie Probleme feststellen. Auch nicht in anderen Systemen.

Interviewer: Ist die Zuverlässigkeit ausreichend?

Interviewee: Somit auch hier: Zur vollsten Zufriedenheit.

Interviewer: Wie relevant sind die gelieferten Daten der unterschiedlichen API-Elemente? Also, auch brauchbar, bzw. plausibel.

Interviewee: Sehr brauchbar. Sonst würde ich das ja auch nicht einbauen. Ich bekomme da auch sehr positives Nutzerfeedback, bzgl. der Tag-Vorschläge. Das wird sehr geschätzt.

Interviewer: Wie vollständig sind die gelieferten Daten?

Interviewee: Die sind vollständig. Mir ist noch nie aufgefallen, dass beispielsweise ein Tag, den ich erwartet hätte, nicht da wäre. Natürlich hinterfrage ich das nicht in alle Einzelheiten. Dazu habe ich gar nicht die Zeit.

Interviewer: Natürlich. Es geht hier nur um eine Einschätzung. Etwas anderes ist ohne größeren Aufwand schwierig zu auszusagen.

Interviewer: Entsprechen die Rückgabewerte den Erwartungen an die API?

Interviewee: Ja, das tun sie.

Interviewer: Wie schwierig/einfach sind die UI-Elemente von STAGS zu benutzen?

Interviewee: Ich könnte mir vorstellen, dass manchmal die unterschiedliche Einfärbung von Tags und Co-Tags so manchen User verwirren könnte. Das ist allerdings nur ein Gefühl und es hat sich noch niemand beschwert, bzw. nachgefragt. Grundsätzlich wer blogged und sich mit diesem Medium beschäftigt, weiß, was eine Tag-Cloud ist oder wie eine Auto-Completion-Liste funktioniert – dass wenn ein/zwei Buchstaben eingegeben wurden, die Elemente in der angezeigten Liste eine Ergänzung sind. Solch ein System-Verhalten wird ja schon quasi erwartet. Intuitiver geht es kaum mehr. Das einzige, was eventuell nicht offensichtlich ist, ist eben die farbliche Hinterlegung von Co-Tags. Das könnte ich aber auch selber meinen Usern irgendwo erklären, was es damit auf sich hat. Es hat aber, wie gesagt, noch keine Beschwerden von Nutzern gegeben, die von dem Verhalten zu sehr verwirrt waren. Ich sehe somit da auch keinen Verbesserungsbedarf.

Interviewer: Wie schätzen Sie die Nützlichkeit von STAGS für den Endanwender ein?

Interviewee: STAGS finde ich sehr nützlich. Man weiß gleich, wenn man sich in einem gewissen Kontext befindet, welche Themen da gebloggt werden. Das Auto-Vervollständigen hilft die Tag-Menge überschaubar zu halten, Tippfehler zu vermeiden und hilft dabei dem Nutzer dazu anzuleiten die vorhandene

Tag-Menge zu verwenden – ohne ihm die Freiheit zu nehmen neue Tags zu vergeben. Also ich finde das sehr nützlich.

Interviewer: Wird die individuelle Produktivität erhöht?

Interviewee: Das hört sich sehr „hochgestochen“ an. Wenn Sie so wollen, kann man das evtl. so ausdrücken: Die Produktivität wird in dem Sinne erhöht, dass die Tags einigermaßen konsistent verwendet werden.

Interviewer: Gibt es Lerneffekte für den Nutzer?

Interviewee: Der Nutzer sieht den Kontext, in dem eine Person blogt oder auch den Kontext ,in welchem Tags zu einem anderen Thema bereits vergeben wurden. Auch sieht er, welche Tags schon existieren. Man kann über die Tags und auch damit deren Co-Tags zu den Postings bzgl. eines Themas informieren. Man findet dadurch auch die relevanten Personen, die zu einem Thema Bescheid wissen. Darin sehe ich die Lerneffekte. D.h. in der Community sind relevanter Content und Experten leichter zu identifizieren.

Interviewer: Also, die Navigation, die durch STAGS unterstützt wird?

Interviewee: Ja, genau. Dabei entsteht ein gewisser Lerneffekt für den Nutzer.

Interviewer: Welchen erwarteten Nutzen hatten sie?

Interviewee: Ich hatte mir erwartet: Eine bessere Darstellung von geblogten Inhalten, bessere Querverlinkung von Inhalten, leichteres Auffinden von relevanten Inhalten. Ich habe mir eine konsistentere Vergabepraxis von Tags, auch auf eigene neue Postings, erwartet.

Interviewer: Ist der Nutzen eingetreten?

Interviewee: Ja, auf jeden Fall. Vor der Einführung wurden viele Tippfehler gemacht. Z.B. wurde „access control“ von manchen mit einem „s“ geschrieben oder auch „siemens“ ohne „ie“ , also ein klassischer Tippfehler. Das hat sich mittlerweile gebessert. Ich habe den Eindruck die Leute sind grundsätzlich eher bereit, Tags zu vergeben. Ich mache keine Einschränkung auf eine bestimmte Sprache, in der geblogged werden soll. Allerdings sind die Leute jetzt motivierter, englische Schlagwörter zu verwenden – auch wenn sie beispielsweise in Deutsch posten. Das finde sehr nützlich. Die Tags sind somit einheitlicher. Dieser Effekt ist für mich als Admin durchaus spürbar.

Interviewer: War der Aufwand vertretbar für den Nutzen?

Interviewee: Ja, klar. Die eine Woche ist durchaus vertretbar.

Interviewer: Hat es sich somit gelohnt?

Interviewee: Ja, durchaus.

Interviewer: Wie sieht der tatsächliche Nutzen aus? Hat STAGS für Sie einen Business Impact?

Interviewee: Das kann ich nicht quantifizieren, da ich generell nichts zum Business Impact von Microblogging sagen kann. Das in konkrete Zahlen zu packen ist eher schwierig. Ich hatte eine Umfrage zur Nützlichkeit von References+ durchgeführt. Allerdings keine allein zum Microblogging oder auch dem

Nutzen von STAGS. Ich denke, dass ein zumindest subjektiver Nutzen von Blogs besteht. Sollte kein solcher bestehen, dann würde niemand den Dienst nutzen. Auf die Microblogs werden zugegriffen. Das sehe ich anhand von Statistiken. Und Blogs werden geschrieben, was für jeden Nutzer sichtbar ist. Es muss somit irgendeine Form von Nutzen geben. Wie weit sich der Nutzen in geschäftlichen Zahlen niederschlägt, kann ich nicht sagen.

Interviewer: Das ist generell ein Problem von Social Software: Wie kann man den Nutzen in Zahlen messen?

Interviewee: Ich mache das normalerweise über Nutzerumfragen. Ich müsste dazu eine konkrete Frage stellen. Speziell für Microblogs hatte ich das bis jetzt noch nicht.

Interviewer: Hatten Sie Kosteneinsparungen?

Interviewee: Ich kann da keinen quantitativen Betrag nennen, da es sich dabei um ein Feature handelt, das ja zunächst Kosten verursacht hat. Ob ich stattdessen selber eine Lösung implementiert hätte oder eine Lösung gekauft hätte, kann ich so nicht sagen. Es handelt sich um ein tolles Feature und sie erfüllt definitiv ihren Zweck. Ich kann sagen: Das Preis-/Leistungs-verhältnis ist sehr gut. Einen konkreten Betrag, den wir gespart haben, kann ich allerdings nicht nennen.

Interviewer: Hat sich die Beliebtheit der Plattform gesteigert?

Interviewee: Ich hoffe es. Es werden von jeder Plattform zeitgemäße Features erwartet. Ich sehe das mit den Tag-Cloud und der Autovervollständigung durchaus als solche zeitgemäßen Features. Ich erhielt Feedback von Seiten-Ownern im Intranet. Diese sind immer sehr begeistert. Sie bauen eine simple URL in ihre Seite ein und bekommen passende Postings und eine kontextabhängige Tag-Cloud geliefert. Da bekomme ich immer sehr positives Feedback. Die Seiten-Owner schätzen den Service sehr. Wenn sie das einmal integriert haben, wollen sie das nicht mehr missen.

Interviewer: Wurde die Usability verbessert?

Interviewee: Ja, auf jeden Fall. Die Eingabe der Tags wurde sehr erleichtert. Das merke ich persönlich. Man gibt zwei Buchstaben ein und bekommt den Rest ergänzt. Ich persönlich habe als Hauptthemen für meine Microposts: „knowledge management“ , „social media“ , „web 2.0“ und „references+“ . Ich gebe die ersten Buchstaben ein und erhalte den Rest vervollständigt. Das ist definitiv sehr nützlich.

Interviewer: Hatten Sie alternative Lösungen?

Interviewee: Ich habe nicht aktiv nach Alternativen gesucht. Ein Kollege machte mich auf STAGS und die Integration im TechnoWeb aufmerksam. Ich habe dann das ausprobiert, und alles hat sehr gut geklappt. Von daher habe ich keine andere Lösung evaluiert. Ich weiß somit nicht, ob es eine gibt.

Interviewer: Welche Aspekte an STAGS sind alternativlos, bzw. was denken Sie ist besonders an STAGS?

Interviewee: Folgende Aspekte finde ich hervorragend: Wenn man die Dokumentation verstanden hat, ist die Integration in die eigene Applikation sehr leicht. Die Widgets sind sehr einfach an das jeweilige Seiten-Layout anzupassen. STAGS bietet eine verbesserte Darstellung von Inhalten. Das sehe ich als die Hauptvorteile. Ich kann mich nur wiederholen. Nachdem man eine JavaScript-Datei einbindet, ein paar Parameter anpasst und ein bisschen CSS-Styling vornimmt, funktioniert beispielweise die Tag-Cloud in wenigen Minuten.