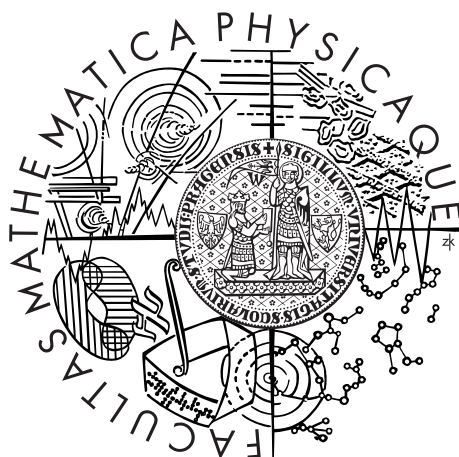


Charles University in Prague

Faculty of Mathematics and Physics

MASTER THESIS



Bc. Michal Folk

Shared Personal Knowledge Base

Department of Software Engineering

Supervisor of the master thesis: Mgr. Martin Nečaský Ph.D.

Study programme: Informatics

Specialization: Software Systems

Prague 2013

I wish to thank Mgr. Martin Nečaský Ph.D., supervisor of my diploma thesis project, for his guidance during the development of the project, suggestions about the thesis, for his patience and time he spent with me. Also, I would like to thank everyone with whom I consulted the project.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague, August 02, 2013

Název práce: Sdílená osobní databáze znalostí

Autor: Bc. Michal Folk

Katedra: Katedra softwarového inženýrství

Vedoucí diplomové práce: Mgr. Martin Nečaský Ph.D.

Abstrakt: Cílem práce je navrhnout řešení problému neefektivity opětovného vyhledávání již jednou nalezené informace. Navrhované řešení je založeno na použití osobní databáze znalostí postavené na stávajících technologiích, přizpůsobené potřebám běžných uživatelů. Diplomová práce se obzvláště zaměřuje na vyhledávání uložené informace na základě sémantické podobnosti mezi tagy. Pro určování podobnosti se využívají kolektivní znalosti velké skupiny uživatelů.

V první části je problém představen pomocí scénářů ze skutečného života. Druhá část práce tento problém analyzuje z pohledu osobní sdílené databáze znalostí. Třetí část vysvětluje navržené řešení postavené na službě Delicious a DBpedii. Navržené řešení je implementováno v podobě prototypu. V závěrečné části je prototyp testován a získané výsledky jsou vyhodnoceny.

Výsledky testování naznačují, že navržené řešení může vést ke zjednodušení opětovného vyhledávání informací, ale zároveň odhalují výkonnostní problémy, které implementovaná metoda přináší. Práce doporučuje úpravy, které by mohly vést k urychlení vyhledávání, což by umožnilo testování prototypu ve větším rozsahu.

Klíčová slova: opakované vyhledávání informací, kolektivní vědomosti, tagy, sémantická podobnost, Delicious, DBpedia, ontologie

Title: Shared Personal Knowledge Base

Author: Bc. Michal Folk

Department: Department of Software Engineering

Supervisor: Mgr. Martin Nečaský Ph.D.

Abstract: The goal of the paper is to suggest a solution of an inefficiency in searching for previously searched and found information. The suggested solution is based on the use of a personal knowledge base built upon existing technologies and adapted to needs of common users. The thesis is focused especially to the search based on semantic similarities between tags. Collective knowledge is used for finding the similarities.

The first part of the paper introduces the repetitive search problem by a few real world scenarios. In the second part the problem is analyzed from the personal knowledge base point of view. The third part explains the suggested solution that is built upon Delicious, a bookmarking service and DBpedia. The suggested solution is implemented as a prototype. In the final part the prototype is tested and evaluated.

The test results suggest that the presented solution can make the repetitive search easier, but at the same time it exposes some performance issues that the proposed method brings up. The paper recommends modifications that could improve the performance and allow more extensive prototype testing.

Keywords: repetitive search, collective knowledge, tags, semantic similarity, Delicious, DBpedia, ontology

Contents

1	Introduction	2
1.1	Motivation	2
1.2	Delimitation of the thesis	3
2	Repetitive search for information	5
2.1	Real life scenarios	5
2.1.1	Scenario 1 - Curiosity rover article	5
2.1.1.1	Common solutions	6
2.1.1.2	Advantages of SPKB over the common solutions	7
2.1.2	Scenario 2 - Tax declaration	8
2.1.2.1	Common solutions	8
2.1.2.2	Advantages of SPKB over the common solutions	8
2.2	Use cases	9
2.2.1	UC 1 - Do not search knowledge of others	9
2.2.2	UC 2 - I do not tag, others tag	10
2.2.3	UC 3 - Support of similarities between tags	10
2.3	Goal of the thesis	11
3	Repetitive search as a part of SPKB	12
3.1	Analysis with respect to the goals	12
3.1.1	Personal knowledge base requirements	12
3.1.1.1	Maintenance and administration	12
3.1.1.2	Knowledge domain	13
3.1.1.3	Habits and needs	14
3.1.1.4	Technology	15
3.1.2	Utilizing collective knowledge	15
3.1.2.1	Popular tags	16
3.1.2.2	Common tags vocabulary	19
3.1.3	Similarity of tags	21
3.1.3.1	Definition	21
3.1.3.2	Similar tags vocabularies	23
3.1.3.3	Similarity relations mining	27
3.2	SPKB components	30
3.2.1	Article Storage	30

3.2.2	Tag Vocabulary	33
3.3	Suggested solution	33
3.3.1	Selected approach	34
3.3.2	Addressing the SPKB issues	35
3.3.2.1	Maintenance and administration	35
3.3.2.2	Respect to habits and needs of the users	35
3.3.2.3	Use of existing technologies	36
3.3.2.4	Ambiguous tags	36
3.3.2.5	Synonymous tags	36
3.3.2.6	Interoperability	37
3.3.2.7	License	38
3.3.2.8	Multi-domain content	39
3.3.2.9	Tags vocabulary	39
3.3.2.10	Tags/relations mining	39
3.4	Similar projects	40
3.4.1	Faviki	40
3.4.2	Zemanta	41
3.4.3	Full-text search engines	43
4	DBpedia as Tag Vocabulary	44
4.1	The SPKB relation schemes	44
4.1.1	Synonyms	44
4.1.2	Specification/generalization	46
4.1.3	Related	47
4.2	SPKB relations mapping	48
4.2.1	Ontologies	48
4.2.1.1	SKOS	49
4.2.1.2	DBpedia Ontology	50
4.2.1.3	Dublin Core	51
4.2.2	SPKB relations mapping overview	52
4.3	Querying DBpedia data sets	52
5	Prototype implementation	54
5.1	Platform	54
5.2	Architecture	54
5.3	Design	57
5.3.1	Endpoint	57
5.3.2	Articles Storage and Tags Extractor	58
5.3.3	Tags Similarity Checker	58

5.3.4	Tags Vocabulary	59
5.4	Build and installation	60
6	Prototype evaluation	61
6.1	Testing	61
6.1.1	Performance issues	61
6.1.1.1	DBpedia	61
6.1.1.2	SPKB performance measures	62
6.1.2	Single article search test	63
6.1.2.1	Testing method	63
6.1.2.2	Search results	65
6.1.2.3	Examining results	65
6.1.2.4	Comparison with Delicious search	69
6.1.3	Precision and recall test	70
6.1.3.1	Testing method	71
6.1.3.2	Results	72
6.2	Drawbacks and possible improvements	75
6.2.1	Long taking search	75
6.2.2	Too few popular tags	76
6.2.3	No popular tags	77
	Conclusion	78
	Bibliography	80
	Attachments	82

1. Introduction

1.1 Motivation

The idea of Shared Personal Knowledge Base, in terms of the thesis, is motivated by an elimination of a repetitive search of already searched and found information. From time to time, regardless to a profession, people face a problem which they already met. A solution that they had found may be too complex to remember, or it may be simply forgotten. Such cases imply the necessity of repetitive search of the information. It may take a lot of time until they find again the information they need. It would have been more efficient if they had saved the information for a future usage, especially when its finding was difficult, or if there is a chance that the given piece of information may be often requested.

Each individual can decide independently whether given information is sufficiently valuable for him, whether it is reasonable to save the information for a future usage. If saving the information makes sense for somebody, that person will save the information. This happens every day in various forms, starting at paper notes and ending at a specialized knowledge management software. The latter option is common especially in companies where maintaining an organized database of knowledge is necessary for the business. In personal world, such a strong necessity for organizing knowledge usually does not exist. Regardless of a preferred manner in which information are stored, step by step, a personal knowledge database is being created.

It is important to realize that all information saved in a knowledge base are already processed. They are processed in a way that a user needs to. There still exist original information sources, which the user had used and processed. In comparison to the original sources, however, the information saved in the knowledge base has an added value. The added value is a custom processing of the original information. The information is processed in a way that brings a specific benefit to a specific individual. Furthermore, it is rather reasonable to assume that if a piece of information is valuable for one man, the same piece of information can be valuable for others too. Therefore, having the saved information publicly accessible may save time of others, when instead of reinventing the wheel they use the provided information.

It is not a revolutionary idea. It is simply what the mankind does all the time - reusing current knowledge.

In the world of individuals, unlike the world of companies, it is not so common to manage knowledge base. At least, not in a manner to be able to share knowledge between people, especially when those people are not related to each other. It does not necessarily mean though, that individuals do not have knowledge that is worth of sharing. Moreover, knowledge sharing is usually not the primary goal why should individuals maintain a knowledge base. If information stored in a personal knowledge base suppose to be high of quality, then these information should have been stored on account of a real need of their owner. After that, as a secondary goal, information from the knowledge base may be provided to others. I believe that motivation for sharing knowledge is, at least partially, hidden in a human selfishness. The selfishness in the best sense. If information saved in a knowledge base bring the owner to other people interested in the same area and to knowledge of these people, it could be a good reason for many people to share their knowledge. Having such an application, people would process content because, in the first place, the processing brings some benefits for them, and the outcome of their work could bring them another benefits on top of that. This process may be understood as a social network, where knowledge of its users are in the first place. It is like a knowledge driven social network.

1.2 Delimitation of the thesis

The previous section describes a vision and a broad context within which the idea of Shared Personal Knowledge Base (SPKB) is introduced. The description outlines and implies several main principles or key characteristics of SPKB, such as:

- Simplifying a repetitive search of information
- Private organization of knowledge
- Knowledge sharing and knowledge based relationships
- Public access to knowledge

It is quite obvious that only the listed characteristics cover a broad range of problems that need to be solved in order to realize the SPKB

idea. SPKB as a whole is much beyond the possibilities of this thesis. For that reason the thesis focuses primarily on *simplifying the repetitive search of information*. It is not possible, however to work on the selected part in an isolation, without touching any other parts of the system. Conversely, the thesis focuses on a solution of the problem which interacts also with the other SPKB principles, but these are considered only in a minimal range.

2. Repetitive search for information

This chapter introduces a few real-world scenarios that explain what is meant by the *repetitive* search for information. For each scenario there are presented commonly used methods for realization of the scenario, and disadvantages of the method are pointed out. SPKB use cases based on the scenarios are formed afterward. Finally, the goals of the thesis are defined.

2.1 Real life scenarios

The first scenario, Scenario 1 - Curiosity rover article, presents a situation when a user by chance, without explicit searching, comes across an interesting article. The article contains information that the user finds out to be useful, but this finding happens after a long period of time since he read the article. In the second scenario, Scenario 2 - Tax declaration, the user repetitively search for a very specific information.

2.1.1 Scenario 1 - Curiosity rover article

User is interested in the space, especially the NASA space program. As he reads some new articles on this subject, he accidentally comes across an article which he likes more than others, because of many technical information about the Curiosity rover. In spite of he has read a lot of similar articles, he has not found details like this elsewhere. The user does not need any of these technical information for any specific purpose at the moment. The information only seem to be interesting for him. Six months later when the user does not have a clue of the server where he had found the article, or its title, and even less of the link to the article, he realizes that the article contains Curiosity rover technical details, which could improve the quality of his upcoming school presentation on this subject. Fortunately, the user has saved the article into his SPKB. Now, he is trying to find the article in SPKB by a few words which he vaguely can remember or guess from content of the article. SPKB returns a link to the article as one of the found results.

2.1.1.1 Common solutions

There are several commonly used methods for getting to the article without using SPKB:

- Bookmarking the article in a web browser
- Bookmarking the article outside of a web browser
- Search for the article in a browser's history
- Search for the article using a full-text search engine

Bookmarking the article in a web browser. The first drawback of this option is the fact that the bookmark is accessible only from the browser. Loss of the locally stored data (for example OS re-installation or hardware failure), using a new device, or using several different devices simultaneously may lead to an inability to get to the stored link to the article.

Browser bookmarks may be considered as a too heavy weight tool for storing something that maybe wont be needed at all in the future. Storing every piece of information with a potential of being used in the future would lead to a huge amount of bookmarks in the browser. Moreover, it brings another issues related to searching the bookmarks.

Bookmarking the article outside of a web browser. There exist online bookmarking services which remove some of web browser bookmarking drawbacks. At least one drawback still remains, though. It is the necessity to know what exact tags the user has assigned to an article what he is looking for the article. Some of the services provides at least partial full-text searching support that may help, but it does not help too much with similarities between tags. For example a similarity between *automobile* and *car*.

Another drawback is that existing online bookmarking services do not use collective knowledge during searching in users' bookmarks. If they did that it could be easier to find a bookmarked article when the user did not assign any tag to the article, but other users did.

Search for the article in a browser's history. Searching for the article in the history of visited pages after six months may be tricky. Especially, if the user uses more than a single device for browsing the

web. Since history data are stored locally, this option suffers with the similar problems as when the article is bookmarked in a browser.

Search for the article using a full-text search engine. This is probably the most preferred option. Search engines can really help most of the times, but in situations when the user does not remember a title of the article and knows only a few rather commonly used words from content of the article, such as “Mars” or “Curiosity”, he probably gets a huge results set. In such cases, the user needs to be lucky. Otherwise it may take a long time to find the requested article. The main problem related to using a full-text search engine is that it searches for the article within the entire web, instead of searching only in the user knowledge base. Features like differentiation of visited links from those unvisited does not always help. For example, if the user does not use the search engine to find the article at the first time the differentiation of visited links is useless.

2.1.1.2 Advantages of SPKB over the common solutions

SPKB suggests a different approach in a few points, specifically:

- Search based on similarity between tags
- Utilizing collective knowledge

We believe these may help with repetitive searching for information.

Search based on similarity between tags could help in situations when a user does not remember the tags which he has used for organizing a specific piece of content. For example, if he used *Automobiles*, then *Automobile* should have lead to equal search results. Another example are synonyms, such as *Car* and *Automobile*, or even a broader similarity between tags, for example *Automobile* and *Truck*.

Collective knowledge may find a use in a situation when a user stores an article in his knowledge base, but he does not assign any tag to the article. If there are other users who have stored the same article, there is a chance that at least some of them have used the same set of tags for describing the article. In a case like this, it makes sense to consider common tags as popular tags of the article. The popular tags may be found useful when the user is looking for the article to which he have not assigned any tag.

2.1.2 Scenario 2 - Tax declaration

A user has an obligation to submit his tax declaration. Because his incomes come from several countries and he is still a student, he is subjected to different rules than most of the people. The user has read relevant laws but he has still some difficulties to understand to some of their parts. Therefore, he is looking for some good advice in various articles and forums related to the subject on Internet. He is using a full-text search engine and clicks on links, one by one, in the results. Finally, after couple of hours he found the required information. The user uses the information and finishes his tax declaration. A few weeks later, he accidentally falls into a discussion on this topic, when he is confronted with a different opinion related to the very same situation as he had to face earlier. Therefore, the user needs to find his source of the information again.

2.1.2.1 Common solutions

There are more or less the same options for repetitive search of the requested information source, as were described in the first scenario in section 2.1.1.1. However, this example provides a better example of the search engine helplessness in suggesting an appropriate link by a differentiation of the visited links. Almost all links in the search results are visited, but most of them did not lead to the requested information. Using the browser history leads to the very same problem - too many records in the history but it is hard to choose the right one.

2.1.2.2 Advantages of SPKB over the common solutions

The major advantage over the commons solutions that SPKB provides in this case is the fact, that it stores only those information which were evaluated as useful. This evaluation was made by a human being, by the user who had needed those information and eventually he processed them appropriately. SPKB should be a tool that encourages its users to store information even if the user is not sure whether he will need the information in the future or not. In order to do that, storing information in SPKB has to be quick and painless, without forcing the user to organize stored information. The user is the one who is responsible for making the decision whether the information is so specific and worth of that that it needs to be organized, or whether

Table 2.1 – UC 1 - Do not search knowledge of others

MY KB		KB OF OTHERS		SEARCH BY TAGS	SEARCH RESULTS
ARTICLE	ASSIGNED TAGS	ARTICLE	ASSIGNED TAGS		
Mars ex- ploration has started	Mars	Curiosity Rover is First To Witness Martian Streambed	Mars	Mars	Mars ex- ploration has started

there is an assumption that other people will do organize the information for him.

2.2 Use cases

There are several use cases of SPKB which may be extracted from the presented scenarios. The following use cases are going to be realized in form of a prototype within the thesis:

- UC 1 - Do not search knowledge of others
- UC 2 - I do not tag, others tag
- UC 3 - Support of similarities between tags

There are still some ambiguities in specification of these use cases, such as a vague definition of the similarity between tags, but it is intended to be a part of the thesis to suggest a proper solution.

2.2.1 UC 1 - Do not search knowledge of others

The purpose of this use case is to show (see Table 2.1) that only those information stored in the user's own knowledge base are relevant. This fact significantly reduces a set of information that SPKB searches through. The opposite of this approach is a full-text search through the entire Internet.

Some users have stored also the *Curiosity Rover is First To Witness Martian Streambed* article, but this article is not included in the search results, even though it has assigned the searched *Mars* tag. The reason is that the user who is searching for Mars related articles has not stored the article in his knowledge base.

Table 2.2 – UC 2 - I do not tag, others tag

MY KB		KB OF OTHERS		SEARCH BY TAGS	SEARCH RESULTS
ARTICLE	ASSIGNED TAGS	ARTICLE	POPULAR TAGS		
Curiosity Rover is First To Witness Martian Streambed		Curiosity Rover is First To Witness Martian Streambed	Mars, space, NASA	Mars	Curiosity Rover is First To Witness Martian Streambed

2.2.2 UC 2 - I do not tag, others tag

Although articles of other users are not considered when a search is performed, tags assigned by other users to the articles are utilized, as Table 2.2 shows.

This use case is based on the assumption that it is beneficial to use the most popular tags for a given article, especially when the user has not assigned any tags to the article, but other people have. The other people can categorize the article instead of the user:

“First, we show that tagging distributions of heavily tagged resources tend to stabilize into power law distributions...We see the emergence of stable power law distributions as an aspect of what may be seen as collective consensus around the categorization of information driven by tagging behaviours.”[6]

The collective knowledge, popular tags, should be considered only when the user did not tag an article by himself. Otherwise, when the user did tag the article, it should be assumed that he categorized the article exactly as he needs to and his own categorization has a higher value than a categorization of others.

In the example described by the table, the user and also other users have stored the *Curiosity Rover is First To Witness Martian Streambed* article. The user, unlike others, has not assigned a tag to the article. Despite that he can still find the article in his knowledge base by some of the most popular tags of the article.

2.2.3 UC 3 - Support of similarities between tags

A search based on a similarity between searched (entered) tags and assigned tags is a crucial part of SPKB. Table 2.3 shows an example

Table 2.3 – UC 3 - Similarity between tags

MY KB		KB OF OTHERS		SEARCH BY TAGS	SEARCH RESULTS
ARTICLE	ASSIGNED TAGS	ARTICLE	POPULAR TAGS		
Curiosity Rover is First To Witness Martian Streambed	Mars			Martian	Curiosity Rover is First To Witness Martian Streambed

how is the feature expected to work. The user does not need to remember which exact tags he assigned to the article, as far as he knows closely related tags to them.

The example shows that the user is able to find the article even if the searched tag does not match exactly the assigned tag. Both tags are closely related to each other, though.

2.3 Goal of the thesis

The goal is to propose a solution that makes the repetitive search of information easier by:

- (a) using a personal knowledge base for storing information
- (b) utilizing public knowledge in organizing stored information
- (c) incorporating a similarity between tags when is searching for stored information

In other words, relying on a cooperation between people, even if the cooperation is indirect or unaware, is preferred over automatic content description generators.

The proposed solution will be verified by a prototype. Instead of creating a new technology, existing and widely used technologies¹ are going to be used.

¹In next parts of the document, the word *technology* is used often and it covers not only technologies in the strict sense (e.g. Ethernet), but also applications (e.g. Delicious) or frameworks (e.g. RDF).

3. Repetitive search as a part of SPKB

In this chapter, the repetitive search is analyzed from various aspects. Issues related to the SPKB repetitive search support are introduced and potential solutions of the issues are outlined. Finally, a final solution is suggested.

3.1 Analysis with respect to the goals

In order to suggest a solution fulfilling the goals, it is necessary to know what problems the solution needs to solve. The problems are revealed by analyzing the use cases and requirements coming out of the use cases.

3.1.1 Personal knowledge base requirements

Personal knowledge base systems are specific in some aspects, especially in comparison with knowledge base systems used in companies. The differences need to be considered when a personal knowledge base system is being designed. This section introduces some requirements specific for usage of systems for storing knowledge in a world of individuals.

3.1.1.1 Maintenance and administration

Comparing a personal knowledge base, regardless of a specific definition of *personal knowledge base*, with a knowledge base systems widely used in companies, two major differences emerge. The first one is a motivation that drives storing knowledge in these two very distinct environments. The second difference is an availability of hardware and administration skills. Both of the afore differences play a crucial role in designing a personal knowledge base system.

In companies, there are employees who are responsible for storing and maintaining a knowledge base. There may be employees devoted to this task, or it may be the responsibility of all employees across the company. In both cases, maintaining a knowledge base is a part of their job, they are paid for doing that, or they are penalized for not doing that. Optionally, they may be inspected whether they do it

properly. At the opposite end of the spectrum, in a world of individuals, no direct reward, inspection or punishment exist. Individual users need to understand why it is beneficial for them to store their knowledge. They usually maintain a knowledge base voluntarily as a free time activity.

Hardware and software infrastructure in companies is incomparable to capabilities in a world of individuals. Even more important than an equipment imbalance in these two worlds is a lack of administering skills of most of individual computer users. Therefore, a knowledge base system that requires a difficult installation or maintenance, regardless of how good the system is, the most likely won't become popular between users.

3.1.1.2 Knowledge domain

Companies typically need to store knowledge related to a specific domain, the domain related to their business. Various domain sets mean various requirements for the knowledge base system. Knowledge base systems may be adjusted to a specific domain. In general, the adjustment to a domain allows a better interoperability between various systems that operate within the same domain. For example, let's have a system for categorizing books. The adjustment of the system to the domain may be represented by a form which contains domain specific information that a user needs to fulfill for each book entered into the system. The form may contain fields specific for the MARC 21² format, such as Abbreviated Title or Translation of Title by Cataloging Agency. It is obvious that these fields do not bring too much benefits in other domains (e.g. chemistry and medicine). However, exchanging information between knowledge base systems that operate with books is easier if the systems use the same format for storing the information. The same approach, a use of specific forms or whatever other adjustment to a domain is difficult to apply for a personal knowledge base system. Personal knowledge base systems need to be as flexible as possible. Time consuming operations, such as fulfilling a domain specific form, need to be eliminated. Moreover, having a specific form, if there are oodles of domains and multiple well known formats or standards for each domain, is not very realistic and of-

²MARC 21 are communication formats, widely used standards for the representation and exchange of bibliographic, authority, holdings, classification, and community information data in machine-readable form. Source: <http://www.loc.gov/marc/bibliographic/bdintro.html>

ten even unnecessary. For example, if a user wants to maintain a To Read list within his personal knowledge base system, he probably does not want to fill the Translation of Title by Cataloging Agency field. He probably does not want to follow any specific format or standard either. The user rather want to enter a title of the book, in whatever format which is understandable for him, and then to add the To Read label to the created piece of information. Or, he may want just append a title of the book to an already existing record. This is a significant difference between a company knowledge base system and a personal knowledge base system. It is crucial to consider this different approach to storing knowledge when a personal knowledge base system is being designed.

On the other hand, although a personal knowledge base should support a wide range of domains, a single user usually stores information belonging to only a few domains. One user is usually interested only in a few subjects in such a way that he wills to maintain a knowledge base with the subjects related information. For example, if a user is an avid RC modeler (radio-controlled models) and at the same time he is interested in the universe and also in some other subjects, then a major part of his knowledge base is probably related to the RC models and only a small part of records, if any, is related to those other subjects. It is important to be aware of this premise. Although, it is only an assumption, but rather reasonable, based on common sense that one individual is not able to be deeply involved into too many projects. The assumption could help solve some problems, such as an ambiguity between tags which are being used by other users (for example Jaguar as an animal and Jaguar as a car brand). The ambiguity between tags is analyzed later in this chapter.

3.1.1.3 Habits and needs

As a user, so a company, both have own habits, customized processes and needs for saving knowledge. In both cases, there is a possibility to adjust the habits and processes to a knowledge base system. Actually, users often adjust processes to a software that they use. The better option, however, is adjusting the system to needs and habits of its users. In order to do that in context of a personal knowledge base system, the knowledge base system should be as flexible as possible. Ideally, the personal knowledge base system should not force the users to change their habits. It should be built upon their habits and use

outcome of their work, in this case it is an added value in a form of human processing of content, as a benefit provided for the users. In other words, let a user to work as he wants to, because he knows the best what he needs, adjust the personal knowledge base system to his needs, and then try to convert outcome of his work to another benefits, either for him, or for other users.

3.1.1.4 Technology

From a user point of view, it is better using existing technologies than creating new ones. There is a chance that a user is already familiar with a technology that already exists on the market and therefore also familiarizing with a personal knowledge base that uses or is based on the technology may be faster. A user could appreciate even more if he can use the same technologies as he has already been using. This approach does not require him to change his habits. Advantages resulting therefrom were already described.

One of benefits of using existing technologies is that it is necessary to implement only those parts of SPKB that do not exist yet, or that bring an added value over or solve some problems of the existing technologies. In some cases, this may be done transparently, without awareness of users. For example, an implementation of SPKB may extends an existing knowledge base system. If the existing knowledge base system provided a client application that is very popular among users, then they may still use the same client application alongside with the newly implemented SPKB. On the other hand, using a scenario like this is often not possible because of license issues.

In a relation to the required flexibility and a wide range of potential users it could be effective to build SPKB from independent services. Independent services allow to replace one service for another. By this way the result system may be customized.

3.1.2 Utilizing collective knowledge

Tagging has been proven to be a popular method for organizing and describing content. Users of a wide spectrum of applications are accustomed to using tags. Because of this and because of the simplicity and flexibility that tags provide over other methods (e.g. directory structure), tagging has been chosen to be used by SPKB.

3.1.2.1 Popular tags

Having a large user base with a potential that more than one user may store the same piece of information, then it is quite reasonable to assume that besides the information itself, also a way how the information is organized could repeat. In other words, there is a chance that various users may add the same tags to the same piece of information. As a proof we can use the [7] research, which observes a convergence rather than divergence in tag choice proportions:

“One might expect that individuals’ varying tag collections and personal preferences, compounded by an ever-increasing number of users, would yield a chaotic pattern of tags as time goes on. However, it turns out that the combined tags of many users’ bookmarks give rise to a stable pattern in which the proportions of each tag are nearly fixed.”[7]

One could still argue that if there is an increasing number of users, a relatively stable set of tags, then these tags may be used for labeling different resources. In other words, besides the increasing number of users, there is also an increasing number of tagged resources, and in an extreme case there would be no two users who stored and tagged the same resource. In this case, having a stable set of tags does not imply the same organization of the same piece of information among the users, simply because there does not exist any resource shared between the users. This objection, however, shows as incorrect when we explore, in addition to the afore research, whether it is common practice or not to bookmark the same content between various users. For example, Popacular³, a tool for creating a list of the most bookmarked resources from Delicious⁴ (that is the same bookmarking service as was used in the afore research), emerges that there are a lot of URLs bookmarked multiple times by different users. For example, in time of writing this text, FlatUI⁵ homepage was bookmarked 308 times in period of the last month.

On the basis of the fact that various people tend to organize the same piece of information similarly, it makes sense to extract popular tags and use them as the collective knowledge, or a collective agreement on the description of the information. Having a collectively

³<http://popacular.com/>

⁴<http://delicious.com/>

⁵<http://designmodo.github.com/Flat-UI/>

agreed description is useful when a user looks for a specific information within his knowledge base, especially in a case like was already described by the use case in section 2.2.2.

Ambiguous tags. Simplicity of tags brings, among others things, an inability to distinguish two homonyms⁶ from each other. For example, one user may use *Jaguar* for describing content related to the vehicle brand, while another users uses the very same tag for describing content related to the animal species.

At first glance it looks like a serious problem in context of a knowledge base, especially regarding the process of content searching. However, UC 1 - Do not search knowledge of others considerably alleviates the seriousness of the problem, since the only content taken into account when a user is searching for a specific information is the content stored within the user's knowledge base. It is not the same as when a user is searching for new information. For example, if a user wants to get to know something about the jaguar animal species, he can use either full-text search or a search tool of other kind for getting to the requested information. The tool that is being used is going to search through the entire information base of its own and returns those results that match the entered "Jaguar" keyword/tag. The result set probably contains articles related to both, the cars and also animals. After the user finds some of the provided articles interesting, then he can decide to store the article in his personal knowledge base and assign the "Jaguar" tag to it. Later, when he needs to get to the same article again, now the article is stored in his knowledge base, and he starts to search for the article by the tag "Jaguar", he won't get other articles besides those stored within his own knowledge base. That is the difference. As far as he is not interested into Jaguar brand cars and jaguar animal species simultaneously he won't observe the ambiguous tag problem when is working with his knowledge base.

There is a chance that a user is really interested in both domains. In this case, there is still an option for him to choose a different tag for at least one of the domains. Among other things, because of the existence of this alternative option, this paper does not address the ambiguity of homonymous tags issue. One solution of the issue is

⁶Definition of homonyms: One of two or more words that have the same sound and often the same spelling but differ in meaning, such as bank (embankment) and bank (place where money is kept). Source of the definition: <http://www.thefreedictionary.com/homonym>

employing more advanced tags which contain unique identifiers. For example, each tag can be represented by a pair of a label and an identifier. The identifier is used for distinguishing two tags, regardless whether the label parts of the tags equal to each other or not. The very same principle is used in well-known Resource Description Framework⁷ (RDF). RDF uses URLs as identifiers of resources. The resource is whatever that needs to be described. Thus, a tag is also a resource. In fact, there are projects, for example Common Tags⁸, that utilize RDF for addressing the issue of ambiguous tags. Using such methods, however, almost always complicates using the whole system. For example, a user needs to select an appropriate context into which a just assigning tag belongs. Therefore, as long as possible, there is an effort to keep tags flat, without additional information. On the other hand, the option to incorporate more advanced techniques or technologies, such as RDF, is not excluded, especially if it can help with another issues related to a realization of defined use cases.

Synonymous tags. There is also another form of a tags ambiguity. The previously described case was about using homonymous tags in two different context. This time, the ambiguity is about using synonymous⁹ tags within the same context. In other words, suppose there is an article about an upcoming car model. For describing the article, various people may use different tags, such as *Car*, *Automobile*, *Vehicle*, *Automotive*. All of these convey basically the same thing. SPKB should know there is a relation between these tags. Otherwise, if a group of users use *Car* and this tag happens to be a popular tag for the article, then it can happen that the article is not found if somebody tries to find the article by *Automobile*, even though it is rather obvious that both tags are very similar.

Using a common tag vocabulary could eliminate the problem. For example, the common tag vocabulary could define the “Automobile” tag as a tag which is going to be used for describing all content related to automobiles. No further using the “Car”, “Vehicle” or “Automotive” tags.

⁷<http://www.w3.org/RDF/>

⁸<http://commontag.org/>

⁹Definition of synonym: a word or phrase that means exactly or nearly the same as another word or phrase in the same language, for example shut is a synonym of close; Source: <http://oxforddictionaries.com/definition/english/synonym>

3.1.2.2 Common tags vocabulary

The first thing that needs to be taken into account in order to use a common tag vocabulary is the necessity to have one. As a solution to the synonymous tags problem it would be enough to have a thesaurus. How exactly the vocabulary is used depends on a specific implementation of SPKB, as well as on options of the vocabulary that is being used. For example, a main word that represents a synonymous group can be selected and users are allowed to use only the selected main words for tagging. Another example how to incorporate the vocabulary into SPKB is allowing the users to use arbitrary tags, but chosen tags are translated to the main tags internally, if such a transformation is required. These very briefly presented examples outline only a few options how to resolve the synonymous tags problem. There are also others, but it is not crucial to analyze them at this level of the paper.

Interoperability. Even though, there exist many online thesauri, they provide different, often quite limited, if any, options for using their data. Primarily because of interoperability issues. For instance, there is a very popular service The Free Dictionary¹⁰, which includes a thesaurus containing a synonym database, but no official API exists for getting the data.

One step further than not having an API, but still not an optimal option, is having a proprietary API. In other words, having an API which is not in comply with any well-known standard. A cooperation with such services brings complications related to a future extension of SPKB, or to replacing one implementation of a component with another. For example, if SPKB allowed its users to use whatever thesaurus they want then their preferred thesauri would have to comply with a selected standard which defines rules for exchanging data. Otherwise, if different thesauri used different protocols, easy switching between them would not be possible.

License. Even if an existing vocabulary with a public standardized API is found, another complication in form of license or policy issues may appear. As an example, there is Dictionary.com¹¹ that never-

¹⁰<http://www.thefreedictionary.com/>

¹¹<http://dictionary.reference.com/>

theless provides API, it is provided only to selected partners, as the service declares on the web page:

“We are selective about our API partners and approve use as well as develop terms on a case-by-case basis.”[8]

Similar license issues relate also to other services analyzed in this paper. In case that a service is going to be used for the purpose of implementing the SPKB prototype, but there are potential complications regarding the license, this fact is explicitly stated.

Multi-domain content. Ideally, a vocabulary with a support of a well-known standard for exchanging data should be used. From the SPKB point of view, an appropriate standard is SKOS¹². SKOS provides a standardized method for organizing knowledge by defining a set of relations which express how two pieces of information relate to each other. The SKOS standard allows knowledge to be stored in distributed and decentralized applications and makes transferring of the knowledge between these applications easier. At this point, the main reason for using SKOS is that it is a standard. Other important reasons why it is beneficial to employ a vocabulary supporting SKOS will appear and be analyzed in a section related to the similarity between tags.

Having a proper standard does not necessarily mean that there is a wide range of vocabularies supporting the standard. Typically, there are specific domain oriented SKOS data sources. From the nature of SPKB, as was described in above sections, it is required to have a multi-domain vocabulary in order to cover different groups of users. For example, some of well-known SKOS data sources are the following: STW Thesaurus for Economics¹³, Country Codes¹⁴, UK Public sector vocabularies¹⁵ or Polythematic Structured Subject Heading System (NTK PSH)¹⁶. Even though that the last listed data source is an exception and it contains subjects across domains, there are still missing many commonly used tags. Only a few examples of the missing tags are: Ubuntu, Star-wars, vegan, video, start-up, humor. In case of the other data sources the situation is even worse.

¹²<http://www.w3.org/2004/02/skos/intro>

¹³<http://zbw.eu/stw/versions/latest/about>

¹⁴<http://eulerssharp.sourceforge.net/2003/03swap/countries#>

¹⁵<http://standards.esd.org.uk/>

¹⁶<http://psh.ntkcz.cz/skos/home/html/en>

They focus on specific domains and therefore words or phrases from different domains are not included therein.

It is rather common that existing SKOS data sources contain sort of artificial phrases. For example, NTK PSH contains “rock music” instead of more common “rock”, or “young people” or “teenagers” instead of “teen”. It is more difficult to use such formal forms of the words and phrases in an application that is ought to be targeted to common users.

3.1.3 Similarity of tags

3.1.3.1 Definition

SPKB relations. Two tags should be considered similar if there is one of the following relations between them:

- synonym
- generalization
- specification
- related

Figure 3.1 shows an example of the listed relations. It is important to note, that the example does not present the only and right way of organizing the given group of tags. There are other alternatives. Figure 3.2 shows one of them. How are tags eventually organized, what type of relations are assigned between them, it depends on various factors, such as a vocabulary that is being used and what features the vocabulary supports. For example a thesaurus usually defines only the synonym and related relations. Had a vocabulary with a collective knowledge support then the users may decide which way of organizing the tags is the most appropriate.

There may be many others similarity relations than those listed above, some of them may capture specifics of different contexts, others may be of universal use. The listed relations are generic and self-explaining enough to cover probably all types of similarities required by SPKB and its users. The list was created by analyzing different words generally considered to be similar (step #1). Found types of similarity were described and named (step #2). Subsequently, these named similarity relations were grouping together until a few groups

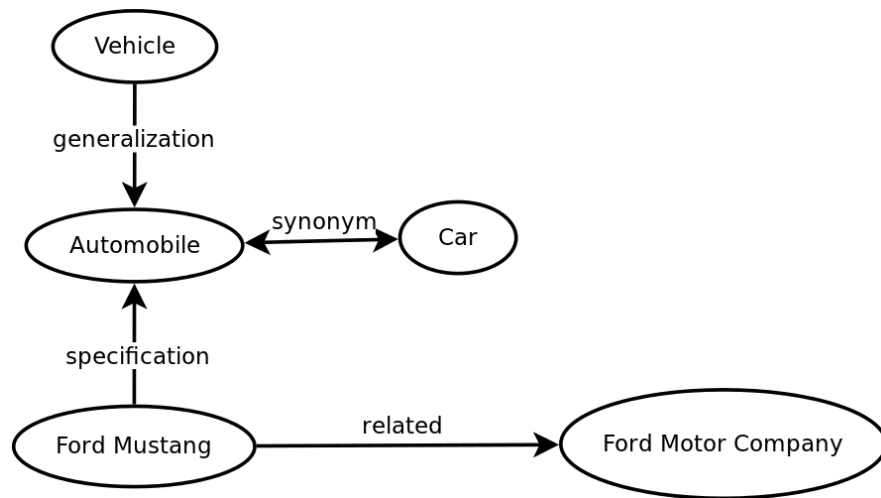


Figure 3.1 – SPKB relations - example #1

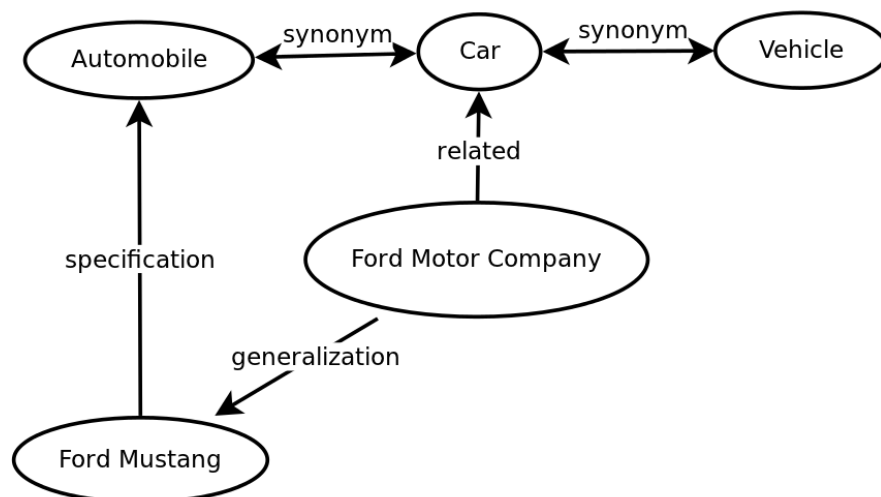


Figure 3.2 – SPKB relations - an alternative to example #1

were created. Each group was named (step #3) and at a certain point in the process it was found out that the names given to each of the final groups are generic enough for describing all relations belonging into the group, and these names also cover practically all possible relation types which may occur between words. Table 3.1 shows this process on a small set of tag pairs.

Table 3.1 – A sample of the process of defining SPKB relations

STEP #1		STEP #2	STEP #3
TAG 1	TAG 2	TAGS RELATION	RELATION TYPE
Alfa Romeo	Automobile	product (of)	generalization
Tesla Motors	Automobile	producer (of)	
A Floating City	Jules Verne	written by	
Apple	Apple Cake	ingredient (of)	

SKOS coverage. A comparison of the SPKB relations with the relations defined by SKOS shows that even though SKOS names its relations differently, the SPKB relations are basically a subset of the SKOS relations. For each SPKB relation, it is possible to find a SKOS relation which is lexically equal to the SPKB relation. Table 3.2 shows the mapping of the SPKB relations to the SKOS relations.

Table 3.2 – Mapping of SPKB relations to SKOS

SPKB RELATION	SKOS RELATION
synonym	skos:exactMatch
	skos:closeMatch
generalization	skos:broader
specification	skos:narrower
related	skos:related

3.1.3.2 Similar tags vocabularies

SKOS versus multi-domain vocabularies. As already suggested, it is beneficial to use a SKOS based tag vocabulary. The major benefit is the direct mapping from SKOS to the SPKB relations. On the other hand, as was said too, no multi-domain SKOS vocabulary was found. Since the multi-domain nature of knowledge plays a crucial role in the context of SPKB, preferring the SKOS support at the cost of restricting supported knowledge domains is not an option.

There are basically two approaches how to overcome the SKOS support versus supported domain range problem:

- joining several specific domain SKOS based vocabularies into one
- an explicit creation of the SKOS relations in a non-SKOS wide domain range vocabulary

Joining different SKOS vocabularies together could create a new multi-domain vocabulary. However, even though there exist many various SKOS vocabularies, there is still a chance that the created multi-domain vocabulary does not cover all domains required by the users during their every day use of SPKB. Moreover, different vocabularies that are maintained independently bring different problems. For example, a license issues, or different ways of accessing the data (such as an online accessible endpoint with a search support versus downloading a raw dataset as a whole in a specified format).

The second approach is based on the idea of using a multi-domain data source, although without the SKOS support. Content of the data source is processed when SPKB creates the required SKOS relations on its own. Thus the generated SKOS data are not included in the original data source. This approach requires SPKB to be able to transform relations between words, that the original data source defines, to the SPKB or SKOS relations. Such a task is far away from being trivial, however. Especially when various data sources use different ways of storing the data. It may happen that there are data sources which make such a transformation easier because they use a method of storing the data that is more compatible with SPKB than methods used by other data sources. For example, if a wide range domains data source defines word relations that are semantically close to the SPKB relations, let's say that only their names differ, but these similar relations are not implemented in the SKOS format, then their transformation either to the SPKB or SKOS relations may be relatively easy. Though, in reality this usually is usually not the case.

DBpedia as a similar tags vocabulary. Searching for a public multi-domain information data source almost inevitably starts or ends at Wikipedia¹⁷. It is only hardly possible to find a bigger information

¹⁷<http://www.wikipedia.org/>

source, especially if inclusion of collective knowledge is desired. Investigating Wikipedia’s data shows a big potential of this data source for the purpose of SPKB. Here are the main virtues:

- content is based on public knowledge
- multi-domain content
- partially structured information
- categorized content

Wikipedia contains most likely information related to all parts of human knowledge. There is a general agreement on information published at Wikipedia. This characteristic is very convenient for extracting popular tags that is part of the SPKB’s collective knowledge feature, described in the section 3.1.2.

Many information published at Wikipedia are structured and therefore they can be extracted from articles easier. For example, such structured information may be: capital, area, population, date of birth, price, timezone, number of wheels, etc. It depends on a subject of the article what type of structured information it presents.

Besides the structured data, Wikipedia defines categories and articles may be assigned into the categories. By this way, articles are organized hierarchically. For example, Figure 3.3 shows a few categories, represented by ovals, which may be found at Wikipedia. The figure defines the *History of Australia* category as a subcategory of *Australia*. Moreover, if indirect relations are considered too then *History of Australia* is additionally included into several upper level categories, specifically *Australia (continent)*, *Continents* and *Oceania*. Similarly, we can see the *History of wars* article as being included into the *History of Australia* or *Australia* category.

It is easy to see that such categorization of Wikipedia’s articles creates a hierarchy or graph that can be described by the SPKB relations, as shows 3.4. Instead of the *specification* SPKB relation, we can use also the *generalization* SPKB relation. Which of them is being used depends on whether the from general to specified point of view is preferred, or vice versa.

In addition to the afore virtues, part of Wikipedia’s content is available also in a machine readable format. This is made possible by DBpedia¹⁸. The DBpedia project community effort to extract facts

¹⁸<http://dbpedia.org/About>

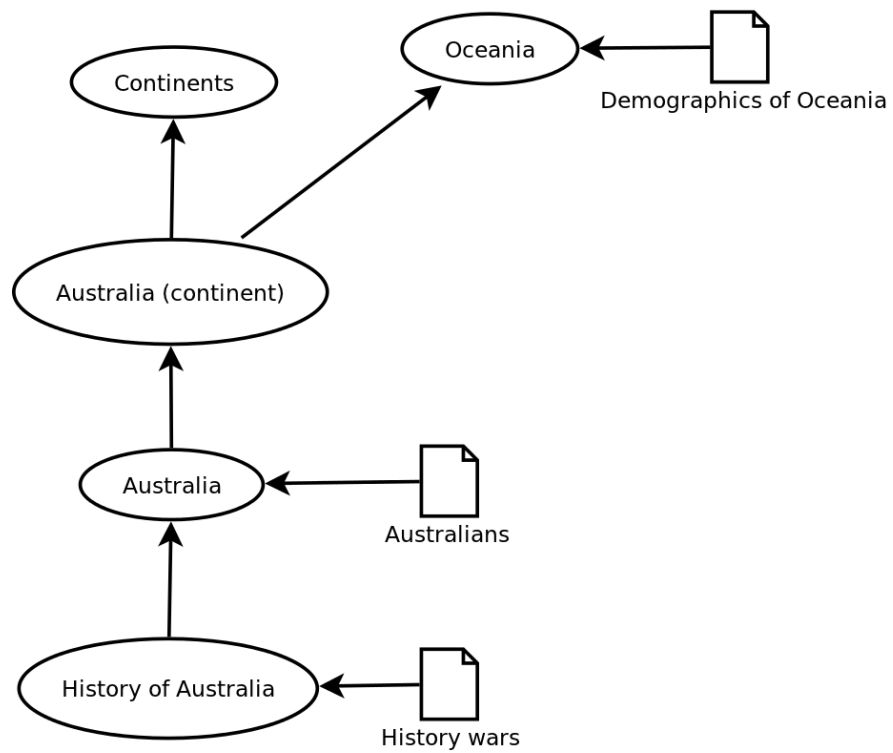


Figure 3.3 – Wikipedia’s categories

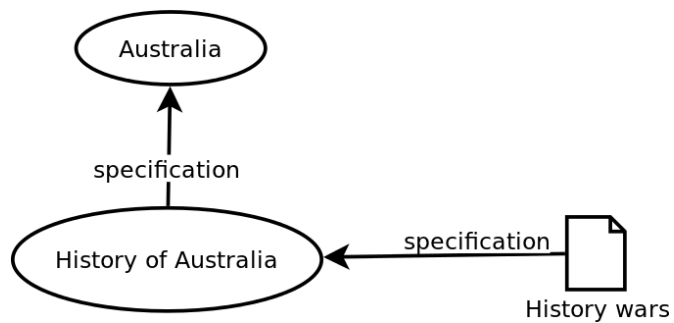


Figure 3.4 – Wikipedia’s categories transformed to the SPKB relations

from Wikipedia. The extracted data are interconnected and a huge publicly available information network (more accurate a graph) has been creating. Besides DBpedia, there exists a very similar project called Freebase¹⁹. Despite the fact that there are some differences between DBpedia and Freebase, such as different methods for accessing data or different technologies and tool sets, both of the projects are basically built upon the same idea. The main difference, though often hidden from a common user perspective, is that whereas DBpedia is an academic project, Freebase is maintained by a private company. Academic nature of the DBpedia project results in occasional failures and unavailability of some of DBpedia's components.

DBpedia uses SKOS for implementing Wikipedia's categories. Because of the direct mapping from SKOS to the SPKB relations (see Table 3.2) using DBpedia/Wikipedia's categories as a SPKB tag vocabulary may be quite straightforward. On the other hand, only the categories themselves cannot represent the full-featured SPKB vocabulary. At least synonyms would be missing in that vocabulary.

3.1.3.3 Similarity relations mining

SPKB should not have been responsible for maintaining a database of tags and relations between the tags. It should rather use an existing database. At this point, it is not important in what form the data are provided by the database. What is important, however, is a way how the data are getting from an external service. There are basically two approaches:

- getting the required data on the fly
- getting whole content of the database in advance

The first option, getting the required data on the fly, requires use of such database which provides content that need no or minimal further processing. Otherwise, the processing could take too long, leading to a delay between a user's request and the response he gets. An example of appropriate content for the needs of this method of getting the data is when the database provides the data and also allows querying for the data in the SKOS format. In such a case, when SPKB needs to find out whether two tags are synonyms, it can ask the database whether the tags are *skos:exactMatch* or *skos:closeMatch* related to

¹⁹<http://www.freebase.com/>

each other. Even though the request sent by SPKB would not be the same request as SPKB had received, a transformation of the original request to the request supported by the database and a subsequent response processing is not too difficult and too time consuming. In this particular example, the request sent to the database could be realized as a SPARQL query (assuming that the database provides a public SPARQL endpoint). SPARQL²⁰ is a query language for databases that support manipulating data stored in the RDF format. Of course, SKOS and SPARQL are not the only options. There are many other formats, protocols or methods how the data can be organized and accessed for easy cooperation with SPKB. On the other hand, SKOS and SPARQL are good candidates, on account of being well known and well supported, alongside with other advantages described in previous sections, for technologies that the SPKB prototype will be build upon.

The second option, getting whole content of the database in advance, allows advanced processing of the data. For example, if the database organizes data in such a way that it is not possible to use them directly for purpose of SPKB, but it is still possible to transform or reorganize them so that they will satisfy SPKB needs, then whole content of the database may be downloaded and processed in advance, before SPKB received the first request from a user. An example of such a processing is reasoning. The reasoning is a process of getting more information out of provided information. In Figure 3.5 there is explicitly said that *Automobile* is specification of *Wheeled vehicle* that is specification of *Vehicle*. There is not said anything about a relation between *Automobile* and *Vehicle*. However, it is possible to establish an internal transitive rule, such as the following one: $\forall a, b, c \in DB : (aRb \wedge bRc) \Rightarrow aRc$; $R = \{specification\}$, which when is applied adds new relations in addition to the original relation set.

In order to apply the reasoning, the database needs to be downloaded before the reasoning is applied. It may be a crucial drawback because it requires SPKB to administrate the database. Because of the fact that real multi-domain databases are huge, it adds other requirements on the SPKB system (such as having powerful hardware, synchronization with the original database, ensuring the availability of the database).

²⁰SPARQL Protocol and RDF Query Language; <http://www.w3.org/TR/rdf-sparql-query/>

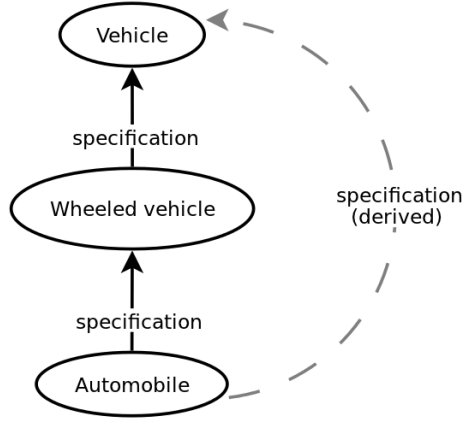


Figure 3.5 – Deriving a relation between tags by the reasoning

In some cases, the reasoning may be replaced by using clever queries. For instance, if the goal is to find out whether *Automobile* is specification of *Vehicle*, then the same result as with the reasoning can be achieved by asking the database whether there exists the following relation chain: $Automobile \xrightarrow{specification} X \xrightarrow{specification} Vehicle$. Algorithm 3.1 presents a pseudo code snippet that suggests a query which can replace the reasoning in the example shown in Figure 3.5. If at least one such record is found which matches the query, then *Automobile* is specification of *Vehicle*.

Algorithm 3.1 A query alternative to the reasoning

```
select ?x where {
    "Automobile" spkb:specification ?x.
    ?x spkb:specification "Vehicle".
}
```

On the other hand, preferring the querying over the reasoning may lead to large number of queries that are being send to the database each time when a tag comparison is required. This is the case, especially if there are many relation chains which match a given SP-KB relation. For example, the following hypothetical list of relation chains:

- $TAG1 \xrightarrow{db:narrower} TAG2$
- $TAG1 \xrightarrow{db:narrower} TAGX \xrightarrow{db:narrower} TAG2$
- $TAG1 \xrightarrow{db:narrower} TAGX \xrightarrow{db:narrower} TAGY \xrightarrow{db:narrower} TAG2$
- $TAG2 \xrightarrow{db:broader} TAG1$

- $TAG2 \xrightarrow{db:broader} TAGX \xrightarrow{db:broader} TAG1$
- $TAG2 \xrightarrow{db:broader} TAGX \xrightarrow{db:broader} TAGY \xrightarrow{db:broader} TAG1$
- $TAG1 \xrightarrow{db:productOf} TAG2$
- $TAG2 \xrightarrow{db:producer} TAG1$

The list above defines only a few of relation chains between *TAG1* and *TAG2* which were (hypothetically) requested to be compared. If the comparing tags were found to match at least one of the relation chains then these tags may be considered similar. In order to get to know this, it is necessary to send these relation chains queries to the database. In the worst and unoptimized variant eight different queries need to be sent only for getting to know whether *TAG1* is specification of *TAG2*. In reality, however, much more queries are expected to be send, and sending and evaluating each of them takes some time.

3.2 SPKB components

At this point, the tasks that need to be addressed are already known. It is possible to see the tasks as the system components. Then, existing technologies may be analyzed to find out what portion of the components' responsibilities the technologies cover.

3.2.1 Article Storage

There are many popular services which can be used for storing SPKB data. In order to compare them and select the one which is the most appropriate for the purpose of SPKB prototype, the following decision making criteria were defined:

- community size (popularity)
- providing an API
- the most popular tags extraction support
- prototype implementation difficulty
- license
- suitability for personal knowledge base usage

These criteria are ordered by an importance (the most important at the top) within context of the SPKB prototype. In a real world implementation of SPKB these criteria would be probably ordered differently.

Table 3.3 presents a comparison of a few the most popular such services. Here is the list of those which were examined: Delicious, Diigo²¹, Evernote²², Faviki²³, Pinboard²⁴, Knowledge plaza²⁵, Historical²⁶, Connotea²⁷. The list of the services is based on user recommendations. The table should not be understood as a very precise and complete comparison of the services. The purpose is not to provide a general guide to help in selecting between the services. The purpose of the investigation is to find an existing service which is good enough to be used as the article storage in the SPKB prototype. There is also an option to not rely on any existing service and use an own data storage. This option, however, is more theoretical than a really applicable, because the Article Storage component is required to provide the most popular tags and for getting such data it is crucial to have an extensive user base. Having an extensive user base is hard achievable with a prototype implementation.

By the preferences of users, it seems that there are not too many such services that are open source and at the same time popular among users and extensively used by them. This brings some license issues and restrictions related to modifying or extending the services. Despite the restrictions, it is mostly possible to use these services freely, even though their nature is commercial. In such cases, the comparison table contains “Limited” value in the license column. If an examined service does not provide any way how to use it without a payment, then the value is “Not free”.

²¹<https://www.diigo.com/>

²²<https://evernote.com/>

²³<http://www.faviki.com>

²⁴<https://pinboard.in/>

²⁵<http://www.knowledgeplaza.net/>

²⁶<http://historio.us/>

²⁷<http://www.connotea.org/>

Table 3.3 – Comparison of potential Article Storage services

	Community size (popularity)	API	Popular tags extraction support	Prototype implementation difficulty	Free to use (license)	Suitability for personal knowledge base
Delicious	Very large	Provided	Supported	Easy	Limited	Low
Diigo	Very large	Provided	Limited	Easy	Limited	High
Evernote	Very large	Provided	Limited	Difficult	Limited	High
Faviki*	N/K	N/K	N/K	Difficult	N/K	N/K
Pinboard	Medium	Provided (Delicious compatible)	Supported	Difficult (license restrictions)	Not free	Low
Knowledge plaza	N/K	Not provided	N/K	Difficult	Not free	High
Historious	N/K	Limited	No	Difficult (too restrictive API)	Limited	Low
Connotea**	Community size N/K, targeted to researchers, scientists	Provided	N/K	N/K	N/K	Low

* Faviki seems to operate (new users registration is allowed, tagging resources works) but project's homepage does not provides basic information, such as API description, WIKI or Help section; ** Connotea discontinued service on March 12, 2013; Source: the project homepage

3.2.2 Tag Vocabulary

Tag Vocabulary is the core component of SPKB. This component is responsible for tasks related to utilizing collective knowledge, specifically the following issues:

- ambiguous tags
- synonym tags
- common tag vocabulary (for multi-domain content)
- similarity between tags (defined by the SPKB relations)

All these issues were already analyzed in previous parts of the document.

From the use of existing technologies point of view, there is a difference between the Article Storage and Tag Vocabulary component. The Article Storage requirements are quite generic. All what was requested was basically having an online database with a big enough user base and the possibility to obtain the most popular tags from the database. Therefore, it was relatively easy to find an existing technology that covers the requirements. In case of Tag Vocabulary, there are more specific requirements. Even though there are various technologies that partially covers the Tag Vocabulary requirements and needs, there was not found a single such technology that covers the requirements fully. In the next part of the document, there is described in details the selected approach that solves all the issues related to SPKB, including Tag Vocabulary, and the way how the used technologies are adapted to SPKB needs. The suggested solution is built upon DBpedia.

3.3 Suggested solution

The former analysis introduces several issues related to sharing personal knowledge and presents some methods how to solve the issues. In this section, the option which is considered to be the most appropriate for the purpose of the SPKB prototype is chosen. Also, the reasons which led to the selection and how to issues are addressed by the selected solution is explained.

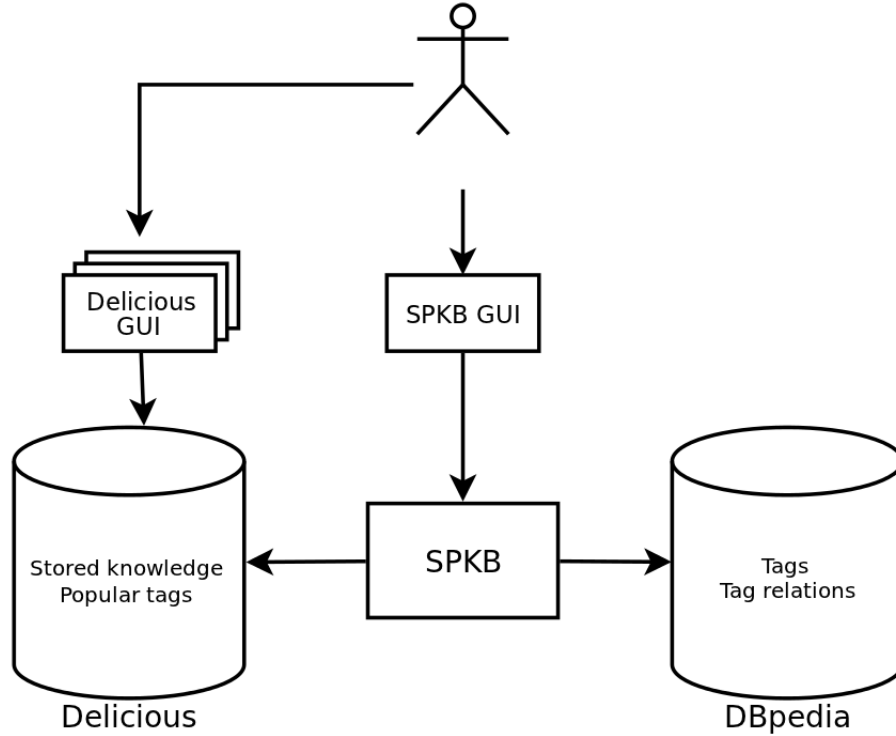


Figure 3.6 – SPKB components

3.3.1 Selected approach

The SPKB prototype is built from three main components. Two of them, the articles/users storage and a tag database reuse existing technologies. Delicious is used as the former component. The latter is built upon data and services provided by DBpedia. The third component is SPKB itself which implements the required functionality and interconnects Delicious with DBpedia. The system schema is outlined in Figure 3.6.

A user uses the Delicious web page, the bookmarklet or any other third-party Delicious compatible application for storing new articles. Later, when the user needs to find stored information he may use SPKB and utilize the benefits of a collective knowledge support combined with recognizing the similarities between tags, as the use cases in the section 2.2 describe. The user is not forced to use SPKB in any specific way, or to use a limited set of tags for describing stored articles. This concept of SPKB rather prefers to let the users to use the system freely, without any restrictions. The concept relies on the assumption that most of the users behave similarly even if they are not forced to act like that.

This concept allows the users to utilize benefits of well-known De-

licious applications and at the same time they may use the benefits of SPKB. Moreover, it is still possible to extend SPKB. For example, it may be extended so that it supports a direct addition of new articles to the personal knowledge database (regardless of a service used as a data storage). After such an extension the users do not need to be aware that other services are used under the hood.

Collective knowledge is encapsulated in both, Delicious and DBpedia. Delicious's users create a list of the most popular tags for stored articles. DBpedia provides tags (usually in form of articles' titles) and data describing relations between tags. SPKB transforms DBpedia's data to the SPKB relations. The SPKB users do not work directly with DBpedia.

3.3.2 Addressing the SPKB issues

3.3.2.1 Maintenance and administration

The suggested solution is of a type client-server. Complexity of system maintenance depends on the client application that is being used by the users. In case of a web interface client, no installation and administration is required from the users point of view. Another option is a browser plugin that represents the client part of the system. In this case, the users need to install the plugin. After the plugin is installed, no more administration tasks are required. Even though there may be another client realizations, basically none of them requires the users to be more skilled than average computer users.

From the server side perspective, it is required to deploy the application to a server. The deployment requires certain administration skills. On the other hand, it is not a responsibility of the users. Moreover, using the external services makes maintaining and administering much easier.

3.3.2.2 Respect to habits and needs of the users

Respecting users' habits follows the fact that the existing technologies are used as the part of SPKB. SPKB supports such methods of storing knowledge for which the users are accustomed. It does not force the users to change their habits or put any restrictions to their workflow. SPKB aims to bring new benefits alongside those that the incorporated technologies provides by themselves.

3.3.2.3 Use of existing technologies

The suggested concept allows SPKB to not be responsible for storing or maintaining data. Instead, repositories of Delicious and DBpedia are used. Delicious is used for more than storing data. It is also used for extracting the most popular tags. It is obvious that benefits offered by the existing services are being used as much as possible. Only the SPKB specific functionality, which is not provided elsewhere, is implemented within the SPKB component.

Moreover, both the Delicious and DBpedia services may be replaced by another one. The concept allows such a replacement. Therefore, preferring other than the selected services may customize the system for another group of users. For example, using Connotes instead of Delicious may customize the system for needs of scientists. Replacing DBpedia by STW Thesaurus for Economics could move the system closer to people in finance.

The main reason why Delicious has been selected as the article storage over another services is its huge generally targeted user base, and as the only one of the examined services supports extracting the most popular tags for stored articles. Even though that it is definitely not the most suitable option for the personal knowledge base, it should be good enough for the purpose of the SPKB prototype.

3.3.2.4 Ambiguous tags

As was pointed out in the former analysis, having homonymous tags is mostly not a problem in context of SPKB. For that reason SPKB does not aim to provide a solution to this issue. It relies on the users and their existing practices regarding the homonymous tags. If a user had in the past face the problem with homonymous tags, then it is probable that he already uses a simple and effective work-around, such as choosing a different tag for one of problematic contexts (for example `Jaguar_car` and `Jaguar_animal` instead of `Jaguar`), or a combination of tags for specifying the context (for example a pair of `Jaguar` and `Animal`, instead of the single `Jaguar` tag). SPKB does not target this issue directly.

3.3.2.5 Synonymous tags

SPKB uses DBpedia as a thesaurus. Since DBpedia does not contain synonyms as a common thesaurus does, SPKB transforms certain

properties (encoded as RDF predicates) between concepts to the SPKB synonym relation. The mechanism of transformation is based on the following principle: if Wikipedia (DBpedia) contains an article titled as “Automobile” and the titles “Car” and “Cars” are redirected to it, then SPKB considers the latter titles to be synonyms of the former. And vice versa. It is probable that the thesaurus created by this method is not as reach as other existing thesauri. On the other hand, DBpedia users are those who decide which words/titles make sense to redirect and therefore it is reasonable to assume that the most commonly used words or phrases are redirected between themselves.

The very same principle is applied for getting the remaining SPKB relations from DBpedia, with the difference that instead of the redirection between concepts, SPKB searches for other properties.

3.3.2.6 Interoperability

SPKB does not generate any new data. It transforms data provided by other services. Thus, the transformed data are not intended to be stored anywhere, and they are not intended to be publicly available either. If someone had been interested in having these data, technically it should not be a problem to provide them in the SKOS format. This is possible because of direct mapping between the SPKB relations and SKOS.

DBpedia uses the RDF standard for storing and providing its data. The fact that DBpedia’s data are obtained in the standardized format makes it possible to replace DBpedia with any other RDF compatible dataset. On the other hand, such a replacement can negatively affect the rest of the system, since searching algorithms may be adapted to DBpedia’s data.

Delicious does not provided data in a standardized format. Therefore, replacing the article storage component is not as straightforward as replacing DBpedia. It would require an additional implementation of an adapter to a new data source. Since providing Delicious’s data through SPKB is not supported, the interoperability is not the issues here. Delicious’s data are read only from the SPKB point of view. If there is a requirement to get Delicious’s data, for example all articles of a user, Delicious API may be invoked directly, instead of using SPKB as a mediator.

3.3.2.7 License

DBpedia and related services. DBpedia’s data are licensed dually, specifically it is Creative Commons Attribution-ShareAlike 3.0²⁸ and GNU Free Documentation License²⁹. It means that it is legally safe to use DBpedia’s data for the purpose of SPKB.

In addition to the datasets, SPKB uses Public SPARQL Endpoint³⁰ (the reason is explained in following parts of the paper). The endpoint is provided by DBpedia using OpenLink Virtuoso³¹, which is licensed under GNU General Public License (GPL) Version 2. OpenLink Virtuoso is not integrated into SPKB. DBpedia allows to use the endpoint but defines Fair User Policy³². In other words, SPKB is allowed to use DBpedia and its related services.

Delicious. There has been found the only paragraph in Delicious Terms of Service³³ regarding the use of Delicious and its content:

In section What Not To Do: “DO NOT: Attempt to access or search the Service or any content on the service or download any content from the Service through the use of any engine, software, tool, agent, device or mechanism (including spiders, robots, crawlers, data mining tools or the like) other than the software and/or search agents provided by AVOS or other generally available third party web browsers;”[9]

On the Developing for Delicious page³⁴, in the APIs - Application Programming Interface section, there is stated the following:

“If you are releasing software or a service for other people to use, your software or service MUST NOT add any links without a user’s explicit direction. Likewise, you MUST NOT modify any URLs except under the user’s explicit direction.”

The way how SPKB uses Delicious does not violate any of the above restrictions.

²⁸<http://creativecommons.org/licenses/by-sa/3.0/>

²⁹<http://www.gnu.org/copyleft/fdl.html>

³⁰<http://dbpedia.org/sparql>

³¹<http://virtuoso.openlinksw.com/>

³²<http://lists.w3.org/Archives/Public/public-lod/2011Aug/0028.html>

³³<https://delicious.com/terms>

³⁴<https://delicious.com/developers>

3.3.2.8 Multi-domain content

Users of Delicious, as well as of DBpedia come from a wide spectrum of area of interest. The former service does not seem to target to any specific audience. In Wikipedia, on the other hand, it is possible to find many qualified knowledge alongside to common knowledge, which makes also DBpedia to be a very wide range knowledge source. It is assumed that DBpedia contains even more potential tags than is really required by the users in order to describe content they store. The integration of Delicious and DBpedia makes SPKB a domain independent service.

Despite that Freebase contains similar data as DBpedia, Freebase uses Metaweb query language³⁵ (MQL), a proprietary query language. Using a proprietary query language would cause complications if a data source relying on the language were used and its replacement were required.

3.3.2.9 Tags vocabulary

The suggested SPKB concept does not restrict the users to use only tags defined within a common tag vocabulary. The users are free to choose whatever tags they want. The used tags are analyzed afterward.

The afore description already indicates that instead of joining several domain-restricted SKOS vocabularies into the single similar tag vocabulary, DBpedia is going to be searched for getting such data that may be transformed into the required SPKB relations. This option has been chosen on account of its simplicity and assumed comparable quality of data with the alternatives.

3.3.2.10 Tags/relations mining

Experimenting with applying the reasoning to RDF datasets led to the conclusion that getting DBpedia's on the fly is a better option for the purpose of the SPKB prototype than processing the data in advance. The experiments described in Attachment 1: Experiments with reasoning with the NTK PSH data set has shown that the reasoning process is too time consuming and demands powerful hardware. Neither of that is available within this project. In a real personal

³⁵<http://mql.freebaseapps.com/index.html>

knowledge base system that is built upon results of this paper, the situation may be different and a use of the reasoning may be a good choice. Under these circumstances, however, from the data point of view, it is possible to get the very same results with querying on the fly. Though, querying DBpedia on the fly and transforming the query results into the SPKB relations is definitely slower than querying a database that was adapted to the specific needs of SPKB.

For querying DBpedia SPKB uses SPARQL and Public SPARQL Endpoint. It may happen that for evaluating whether there exist a required SPKB relation between two tags, multiple SPARQL queries are sent to the endpoint. This is the cost of not having the database transformed so that it contains the SPKB relations in advance.

Performance issues related to creating an inference model may be solved by saving the model once when it is created. The saved model may be used then, instead of creating a new one each time when the application starts. This approach brings new issues related to outdated data, but in context of this project it does not seem to be an issue. The fact is, however, that only the source file of DBpedia’s categories encoded in the N-Triples format contains over 9 millions lines. Therefore, a powerful hardware, even for only a one-time inference model generation, is still required.

3.4 Similar projects

There are other projects which although do not fully cover the SPKB requirements, they need to solve similar issues. These projects use a different approach or philosophy than SPKB for solving the issues.

3.4.1 Faviki

Faviki³⁶ is a service combining social bookmarking and Wikipedia. The users bookmark web pages using Wikipedia’s terms. Faviki tries to solve the problem of tags ambiguity by using a defined set of Common Tags coming from Wikipedia, instead of using flat tags without a semantic information. Therefore, it is possible to distinguish homonym tags from each other.

In spite of the fact that Faviki and SPKB are similar in several aspects, there is a vital difference. The difference is that while Faviki

³⁶<http://www.faviki.com/pages/welcome/>

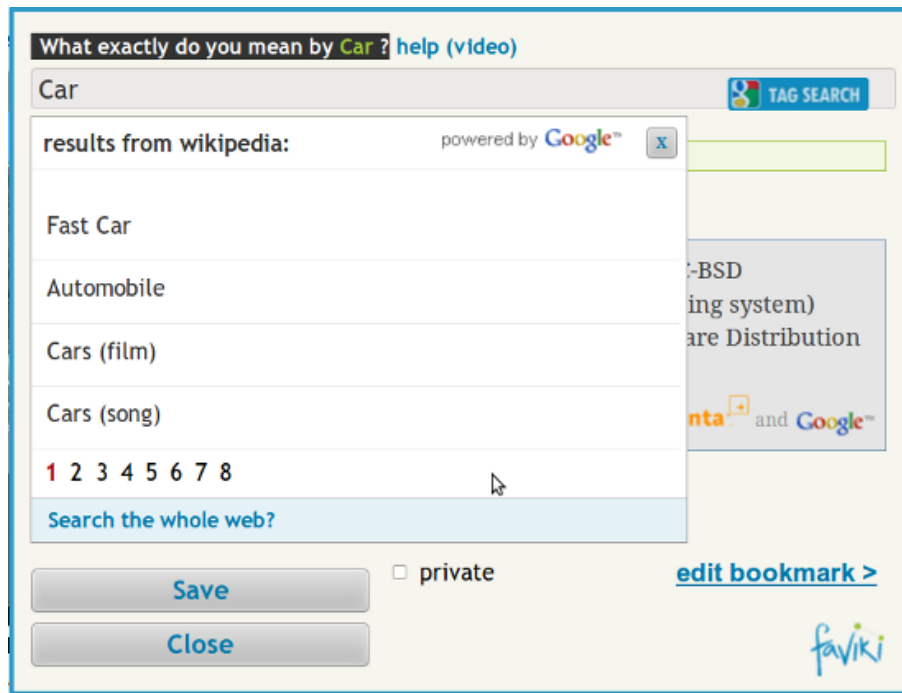


Figure 3.7 – Faviki - specifying context of tags

requires its users during storing articles to specify a context of the used tags (see Figure 3.7), SPKB gives the users a free hand in choosing whatever tags they want and specifying the context is not required either. In other words, the method that Faviki uses is based on preventing the problems arising from the ambiguity or similarity of tags. SPKB, on the other hand, tries to combine available data to resolve the problems after they occur.

Figure 3.7 outlines a restriction coming out of the Faviki’s approach. The users are supposed either to use the “Automobile” tag instead of “Car”, or they are expected to use one of the suggested tags with context information included, such as “Car (film)” or “Car (song)”. It could be in contrary to the users’ habits. Eventually, the Faviki’s approach may lead to misusing tags, as Figure 3.8 shows, where a group of users use the “CaR” tag for describing content related to automobiles. The “CaR” tag, however, relates to “C.a.R.” or “Compass and Ruler”, a geometry software.

3.4.2 Zemanta

Zemanta³⁷ is a content suggestion software intended primarily for content creators, for example bloggers. Besides other features, Zeman-

³⁷<http://www.zemanta.com/>

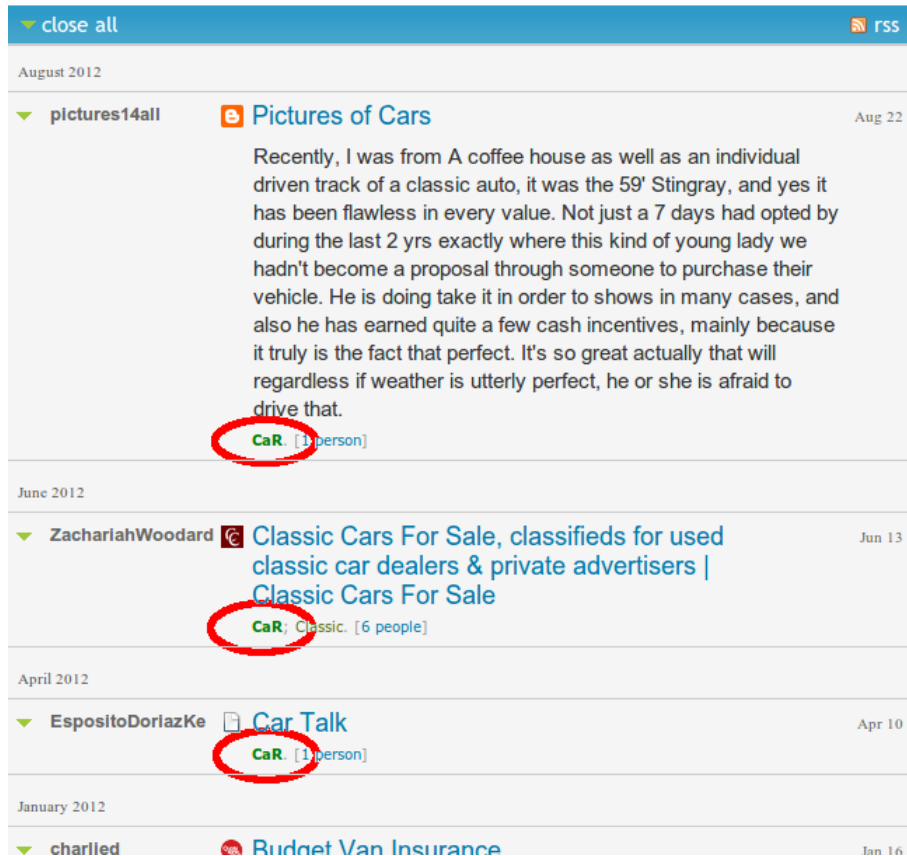


Figure 3.8 – Example of tags misusing

ta analyzes given content and suggests tags suitable for the content. Then the users may decide which of the suggested tags they want to assign to the content. As well as Faviki, also Zemanta creates tags with semantic information. The suggested tags are generated in the Common Tag format. Zemanta uses various publicly available content sources for generating context relevant information. Some of these sources are: Wikipedia, YouTube, IMDB, blogs of others Zemanta users.

The key difference related to tagging between Zemanta and SPKB is that Zemanta generates tags automatically, while SPKB lets the users to create tags. As long as Zemanta's users use the suggested tags, Zemanta interconnects them or their content with content of other users. SPKB, unlike Zemanta, tries to do its job regardless the tags chosen by the users. The common property of Zemanta and SPKB is that they both use public sources and content created by other users.

There are other projects similar to Zemanta. For example, Calais³⁸

³⁸<http://www.opencalais.com/>

analyzes content and creates rich semantic meta-data for the content. The rich semantic meta-data includes entities, topics, events and facts, and also social tags. The social tags created by Calais are, as well as in the cases of the previous projects, based on Wikipedia's articles.

3.4.3 Full-text search engines

Even though that typical use cases of full-text search engines differ from those of SPKB, full-text search engines often use similar methods as SPKB does, to improve the search results. For example, Google supports the following improvements³⁹:

- suggests alternative spelling
- personalizes the search by using information such as sites the user has visited before
- includes synonyms of the search terms to find related results
- finds results that match similar terms to those in the query
- searches for words with the same stem, like "running" when is searched for "run"

The major difference between the full-text search engines and SPKB, from the applying the afore methods point of view, is in a size of knowledge database the applications need to search through. Even personalizing the full-text search by taking previously visited pages into account does not fully solve that kind of problem which is described in Scenario 2 - Tax declaration in the section 2.1.2.

³⁹Source: https://support.google.com/websearch/answer/1734130?hl=en&ref_topic=3081620

4. DBpedia as Tag Vocabulary

Using DBpedia as a thesaurus is the core part of this paper. This chapter describes how the SPKB relations are mapped into DBpedia’s data model and how the required data are obtained from DBpedia. Basically, this chapter presents the transformation of DBpedia into a thesaurus.

4.1 The SPKB relation schemes

DBpedia is a knowledge graph with properties between the concepts. The transformation of DBpedia into a thesaurus is based on the principle of searching for specific property chains between two concepts. Figure 4.1 illustrates the principle. The figure presents a vocabulary in the form of a knowledge graph. Concept labels (vertices) represent words and properties (edges) represent relations between words. The question whether “A” and “B” are similar words may be transformed to the question whether there exists an appropriate path in the graph between the concepts. What does the appropriate path mean depends on a specific SPKB relation that we are interested in.

Having a path between the investigating vertices does not necessarily mean that the words are similar. Not all of the existing paths belong into a defined set of appropriate property chains for the given SPKB relation. In the example there are selected only a few of all possible paths between the examining vertices. The selected paths are represented by colored edges. These paths match tags similarity criteria. The criteria may be understood as sets of property chains. One set per one SPKB relation. If such a path is found in the graph that matches a property chain from the SPKB relation’s criteria set then there exists the SPKB relation between the tags.

The definition of the SPKB relations’ criteria sets starts with creating the property chains schemes. The schemes contain placeholders instead of specific properties. Afterward, in the next section, the placeholders are replaced by specific properties.

4.1.1 Synonyms

Figure 4.2 shows a hierarchical visualization of the schemes that cover the SPKB synonym relation. The fact that synonyms are con-

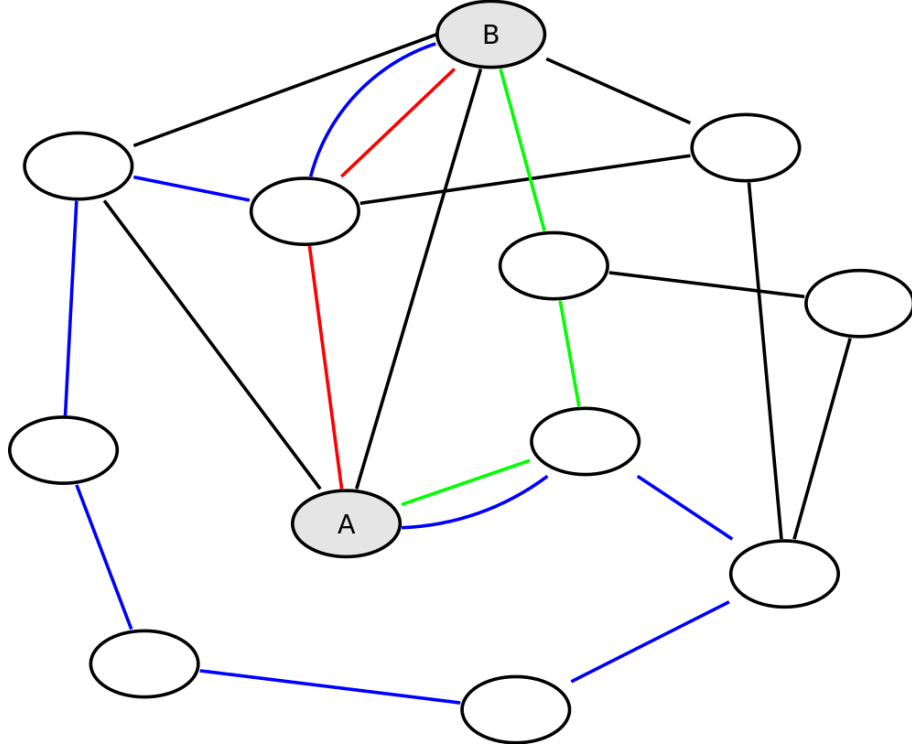


Figure 4.1 – The principle of transformation DBpedia into a thesaurus

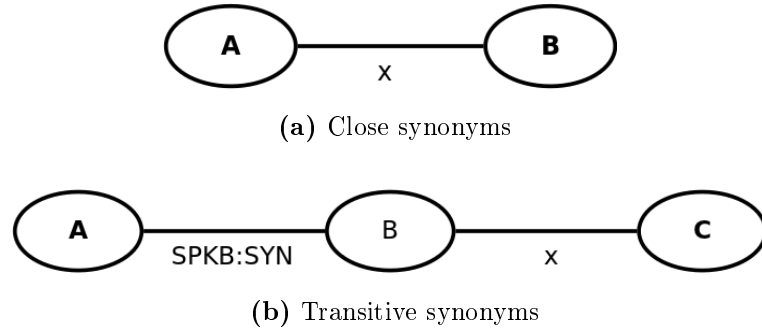


Figure 4.2 – The SPKB synonym relation schemes

sidered to be equal⁴⁰ makes the schemes flat. The first scheme, Figure 4.2a, covers such cases when the equality between tags is explicitly declared in the model. For example, having the model $M = \{(Car, sameAs, Automobile)\}$ and the SPKB synonym relation property set $P = \{sameAs\}$ and $x \in P$ then *Car* is SPKB synonym of *Automobile*, as a result of applying the scheme on the model.

The second scheme, 4.2b, describes such cases when the equality between the examining tags *A* and *C* is not explicitly declared, but

⁴⁰Equal in a sense that if a hierarchy of words is created then those words which are synonyms to each other would be placed at the same level (depth) in the hierarchy. In other words, those words are equal, for which it is not possible to decide which one of them is narrower/broader than others (e.g. *Car* and *Automobile*).

there exists another tag which both of the examining tags are synonyms with. Note that the length of the path between the examining tags is unlimited. In other words, since the synonyms are equal to each other, it is not important how many mediators it is required to find a path between A and C . For example, having the previous example extended as follows: $M1 = M \cup \{(Automobile, sameAs, Automobiles), (Automobiles, sameAs, Vehicles), (Vehicles, sameAs, Vehicle), (Vehicle, sameAs, Motorcar), (Motorcar, altLabel, Wreck)\}$ and $P1 = P \cup \{altLabel\}$. Then, on account of the latter scheme, tags *Car* and *Wreck* are SPKB synonyms.

4.1.2 Specification/generalization

In Figure 4.3 there is a compound scheme that contains all possible paths representing the SPKB specification or generalization relation. Since these relations are inverse to each other, whether it is SPKB specification or SPKB generalization, it depends on the direction of the path. The compounded schema is assembled from 7 sub-schemes, each of which represents one particular SPKB generalization/specification path, specifically (defined by a vertices sequence):

- (A, B)
- (A, B, C)
- (A', A, B)
- (A', A, B, B')
- (A', A, B, C)
- (A, B, C, C')
- (A', A, B, C, C')

From the above list or the figure it is possible to see that in order to consider a path to be a definition of the SPKB specification/generalization relation, the path needs to contain either the (A, B) or (B, C) edge, or both of them.

The x and y placeholders in the scheme will be replaced by a SPKB specification/generalization relation property. For example, having an extended model $M2 = M1 \cup \{(Tesla Motors, producer, Automobile)\}$, and SPKB specification/generalization property set $P2 = \{producer\}$;

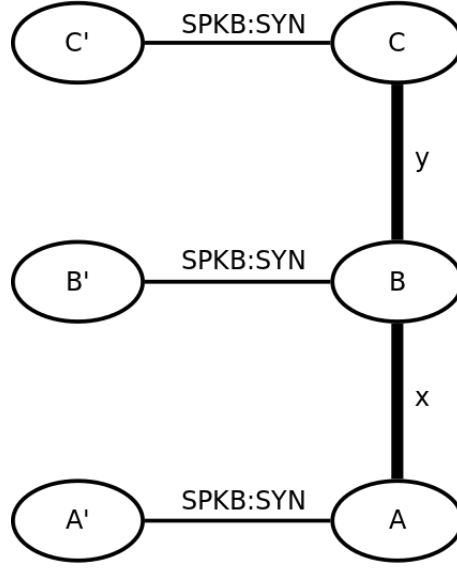


Figure 4.3 – SPKB specification/generalization relation schemes

$x, y \in P2$ and $P1$ represents the SPKB synonym property set, then on account of the existence of a (A', A, B, B') matching path in the model, specifically $((Tesla\ Motors, producer, Automobile), (Automobile, sameAs, Vehicles), (Vehicles, sameAs, Vehicle), (Vehicle, sameAs, Motorcar), (Motorcar, altLabel, Wreck))$, *Tesla Motors* is SPKB generalization/specification of *Wreck*.

4.1.3 Related

There is only a slight difference between the scheme of the SPKB related relation and the SPKB specification/generalization relation. The former is shown in Figure 4.4. Unlike the latter, the SPKB related relation between tags should be explicitly defined and transitivity of the relation is not supported in this case. In other words, if *Automobile* were SPKB related to *Auto racing*, and *Auto racing* were SPKB related to *Racing video game*, then *Automobile* should not SPKB related to *Racing video game*.

The similar restriction as in the case of SPKB specification/generalization applies here too, namely that in order to consider a path to be a definition of the SPKB related relation then the path needs to contain the (A, B) edge.

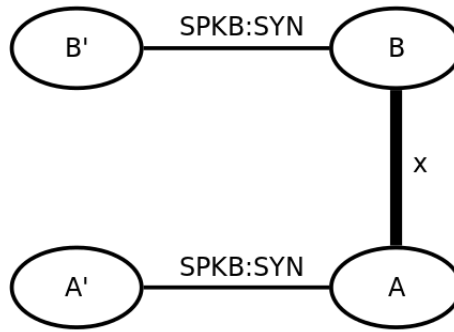


Figure 4.4 – SPKB related relation scheme

4.2 SPKB relations mapping

In this section, the placeholders in the SPKB relation schemes are replaced by specific properties belonging into a few well known ontologies.

4.2.1 Ontologies

There are many ontologies that could be used for defining the SPKB relations. Only those are considered though, whose properties occur in DBpedia extensively. Such ontologies are:

- SKOS
- DBpedia Ontology
- Dublin Core

Since RDF Schema⁴¹ is not an ontology it is not included in the list. In spite of that, the *rdfs:label* property that is defined within RDF Schema is used across all SPKB relations. The reason why the use of this property is necessary is the fact that the SPKB relation schemes presented in the previous section are simplified and they do not show that labels of the concepts are actually properties of the concepts. In Figure 4.5 there is the SPKB close synonym relation (see Figure 4.2a) extended by the label properties.

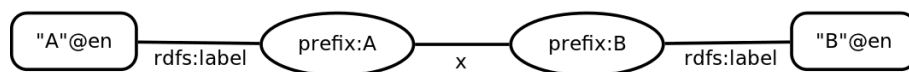


Figure 4.5 – The close synonyms scheme extended by label properties

⁴¹<http://www.w3.org/TR/rdf-schema/>

A practical consequence of having the label properties is that every WHERE clause of the SPARQL queries used by SPKB to check the existence of a required property chain in the model starts and ends with a label specification, as it is possible to see in Algorithm 4.1.

4.2.1.1 SKOS

DBpedia uses SKOS for implementing the hierarchy of the categories. Since DBpedia defines more than 800 000 categories⁴² they are significant source of tags. Moreover, since there is direct mapping between the SPKB relations and SKOS properties, then the SKOS ontology is a reasonable choice.

SKOS properties compatible with the SPKB relations. After exploring all properties defined in the SKOS data model it is possible to pick some of them that may be used for creating a hierarchy of tags. Here is the list of such properties: *skos:altLabel*, *skos:broadMatch*, *skos:broaderTransitive*, *skos:closeMatch*, *skos:exactMatch*, *skos:hiddenLabel*, *skos:narrowMatch*, *skos:narrower*, *skos:narrowerTransitive*, *skos:prefLabel*, *skos:relatedMatch*, *skos:topConceptOf*. For the purpose of SPKB it is possible to use only some of the listed properties, though. Specifically the following:

- *skos:prefLabel*
- *skos:broader*
- *skos:related*

Other properties from the list DBpedia does not use. Therefore, using them in SPKB does not make any sense. Even though the *skos:prefLabel* property is used in DBpedia, SPKB does not use this property since it is redundant to *rdfs:label* that is being used instead.

SPKB generalization/specification mapping. The *skos:broader* property is used by SPKB to define the SPKB generalization, respectively SPKB specification relation. Since DBpedia does not use *skos:narrower* at all, SPKB has to use only *skos:broader* for both of the SPKB relations too. All SPKB has to do in order to use the one SKOS property for both of the SPKB relations is changing the direction when it

⁴²The number relates to the most extensive English version of DBpedia. The labels of the categories may be downloaded from <http://wiki.dbpedia.org/Downloads38#h224-1>.

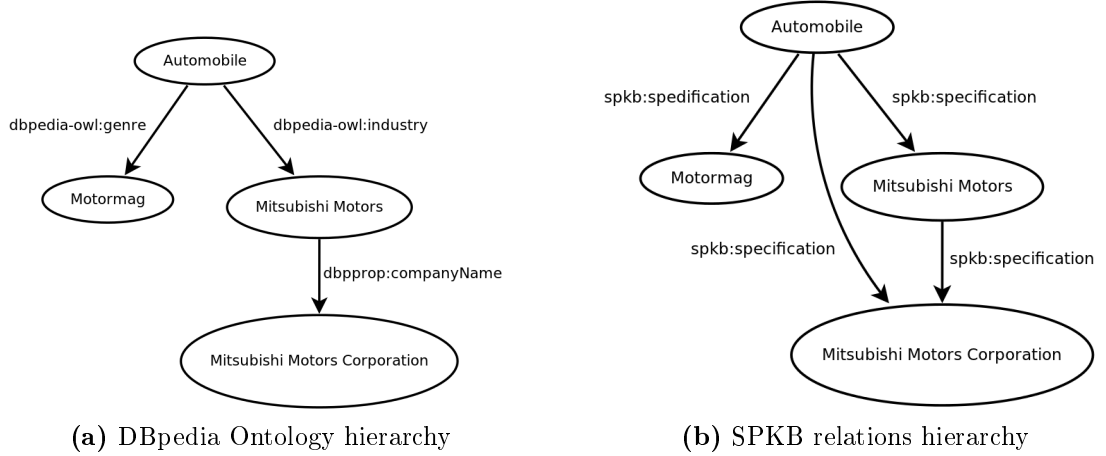


Figure 4.6 – Use of DBpedia Ontology to create a hierarchy

checks the existence of the SPKB specification relation between the tags. Formally, if the predicate $P(a, b)$ tells that a is SPKB specification of b , then the predicate $Q(a, b)$ which tells that a is SPKB generalization of b is defined as follows: $Q(a, b) \Leftrightarrow P(b, a)$.

SPKB related mapping. The SKOS specification provides the symmetric and not transitive *skos:related* property. The property is semantically equal to the SPKB related relation. Therefore, having tags a and b , and model M , then $(a, skos : related, b) \in M \Rightarrow (a, spkb : related, b) \in M$.

4.2.1.2 DBpedia Ontology

DBpedia Ontology⁴³ is a cross-domain ontology which was extracted from the most commonly used Wikipedia’s info boxes. The idea of utilizing DBpedia Ontology in SPKB is based on the fact that some properties that the ontology defines may be used for creating a hierarchy of tags. For example, the following model: $\{(Motormag, dbpedia-owl:genre, Automobile), (Mitsubishi Motors, dbpprop:industry, Automobile), (Mitsubishi Motors, dbpprop:companyName, Mitsubishi Motors Corporation)\}$ may be understood as the hierarchy outlined in Figure 4.6a, 4.6b respectively.

The DBpedia Ontology properties are too specific, however. Thus, they are context dependent. Mapping the SPKB relations to the DBpedia Ontology properties leads either to having covered only a small part of DBpedia, if a small number of the properties is selected,

⁴³<http://dbpedia.org/Ontology>

or to having a huge number of the SPKB mapping rules, if the whole ontology should be covered. Since DBpedia Ontology defines 1775 properties, the latter option is not realistic, especially on account of performance issues. The former option, on the other hand, does not bring a context independent solution, which is required by SPKB so its implementation does not make a sense.

Even though that DBpedia Ontology seems to be not suitable for the purpose of the SPKB prototype, it could change in possible prototype extensions, or after some optimizations are implemented into the prototype.

SPKB synonym mapping. One property is, despite its DBpedia specificity, heavily used by SPKB, though. It is *dbpedia-owl:wikiPageRedirects*. The mechanism of the Wikipedia articles redirection and its conversion to the SPKB synonym relation was already described in the previous sections. At this point, it is necessary to know that such Wikipedia redirections are recorded by the *dbpedia-owl:wikiPageRedirects* property in DBpedia. SPKB maps the SPKB synonym relation to this property.

Note that despite the former declaration that length of the SPKB synonym property chain is unlimited, SPKB will practically never look for paths longer than 4 (*rdfs:label* properties inclusive), since *dbpedia-owl:wikiPageRedirects* is the only property to which the SPKB synonym relation is mapped and DBpedia does not chain properties of this type. In other words, instead of a two steps redirection (for example Motorcar \rightarrow Car \rightarrow Automobile) there are two single step redirections (for example Motorcar \rightarrow Automobile and Car \rightarrow Automobile).

4.2.1.3 Dublin Core

The Dublin Core⁴⁴ (DC) ontology is a generic, well known ontology that is suitable for describing a wide range of context data. The ontology defines a set of properties, some of which may be used for mapping to the SPKB generalization/specification relations. Such properties⁴⁵ are: *alternative*, *contributor*, *hasPart*, *isFormatOf*, *isPartOf*, *isReplacedBy*, *replaces*, *publisher*, *relation*, *source*, *subject*. However, since

⁴⁴<http://dublincore.org/>

⁴⁵All the listed properties belong into the *http://purl.org/dc/terms/* namespace; usually prefixed as “dcterms”.

DBpedia does not use almost any of them, similarly as in the case of the SKOS properties, only the last listed DC property may be really used by SPKB.

DBpedia uses the *dcterms:subject* property for assigning concepts into the categories. Therefore, SPKB maps the SPKB specification/generalization relation alongside the SKOS properties also to *dcterms:subject*.

4.2.2 SPKB relations mapping overview

Table 4.1 provides an overview of mapping from SPKB relations to the examined ontologies.

Table 4.1 – Mapping the SPKB relations to ontologies properties

Ontology/ SPKB relation	Synonym	Generalization/ Specification	Related
RDF Schema	label	label	label
SKOS		broader	related
DBpedia Ontology	wikiPageRedirects*		
Dublin Core		subject**	

* <http://dbpedia.org/ontology/wikiPageRedirects>; ** <http://purl.org/dc/terms/subject>

4.3 Querying DBpedia data sets

SPKB uses query templates, one template per a SPKB property chain, for converting the SPKB relations mapping to the SPARQL queries. Every SPKB relation, except the SPKB specification relation, uses a set of templates that covers all combinations of the ontology properties which the SPKB relation is mapped to. For example, combining the SPKB synonym relation schemes (Figure 4.2) with the properties corresponding to the relation (Table 4.1) leads to the following two property chains:

- TAG1 $\xrightarrow{\text{wikiPageRedirects}}$ TAG2
- TAG1 $\xrightarrow{\text{wikiPageRedirects}}$ SYNONYM $\xleftarrow{\text{wikiPageRedirects}}$ TAG2

Algorithm 4.1 shows the template created from the second listed property chain. When SPKB needs to check whether two tags are synonyms, SPKB replaces the label placeholders in the templates by the

Algorithm 4.1 SPKB synonym relation template

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?x ?common ?y WHERE {
    ?x rdfs:label "__LABEL1__"@en .
    ?x dbo:wikiPageRedirects ?common .
    ?y dbo:wikiPageRedirects ?common .
    ?y rdfs:label "__LABEL2__"@en .
} LIMIT 1
```

tags entered by the user and sends the queries to DBpedia. If DBpedia contains at least one matching path then the tags are synonyms. The same principle is applied to all other SPKB relations. The number of such templates related to a particular SPKB relation depends on the scheme complexity and on the number of the selected properties.

SPKB may use common templates for the SPKB generalization and SPKB specification relations. The only difference between these relations is in swapping the label placeholders.

5. Prototype implementation

5.1 Platform

The SPKB prototype is a client/server application. Most parts of the prototype are server side. The server is implemented on the JAVA⁴⁶ platform. The client side part (SPKB client) is a thin client that is represented by a simple web user interface implemented in HTML/-JavaScript. The second part of the client represents Delicious and its tools (Delicious client), especially Delicious bookmarklet. However, none of the Delicious tools is an integral part of SPKB prototype. They stay as separate, free to use applications. SPKB client does not replace any functionality provided by Delicious client, for example adding articles. In order to use the SPKB prototype it is necessary to use Delicious client alongside SPKB client.

In order to run the SPKB prototype it is necessary to deploy it to a servlet container compatible with Servlet 2.5 specification⁴⁷, for example Tomcat⁴⁸, version 6 or higher. Since the prototype is built upon the JAVA platform it may run across a variety of operation systems.

5.2 Architecture

Figure 5.1 presents the architecture of the SPKB prototype. As is shown the whole system is assembled from components which are distributed across several nodes. Some of the components are not an integral part of SPKB. Instead, they run as independent services. Such services are specifically Delicious and DBpedia. Both of these are maintained independently by third party organizations and from their point of view, SPKB is only a client of them. There is a remote communication between SPKB and these services. On contrary, the server side SPKB components run at the same JVM instance and there is an in-memory communication among them. The entry point to the SPKB system is the SPKB search web page (SPKB client) shown in a web browser. SPKB GUI involves also Delicious web pages and other client tools that the service provides.

⁴⁶<http://www.oracle.com/technetwork/java/index.html>

⁴⁷Servlet 2.5 specification: <http://jcp.org/aboutJava/communityprocess/mrel/jsr154/>

⁴⁸<http://tomcat.apache.org/>

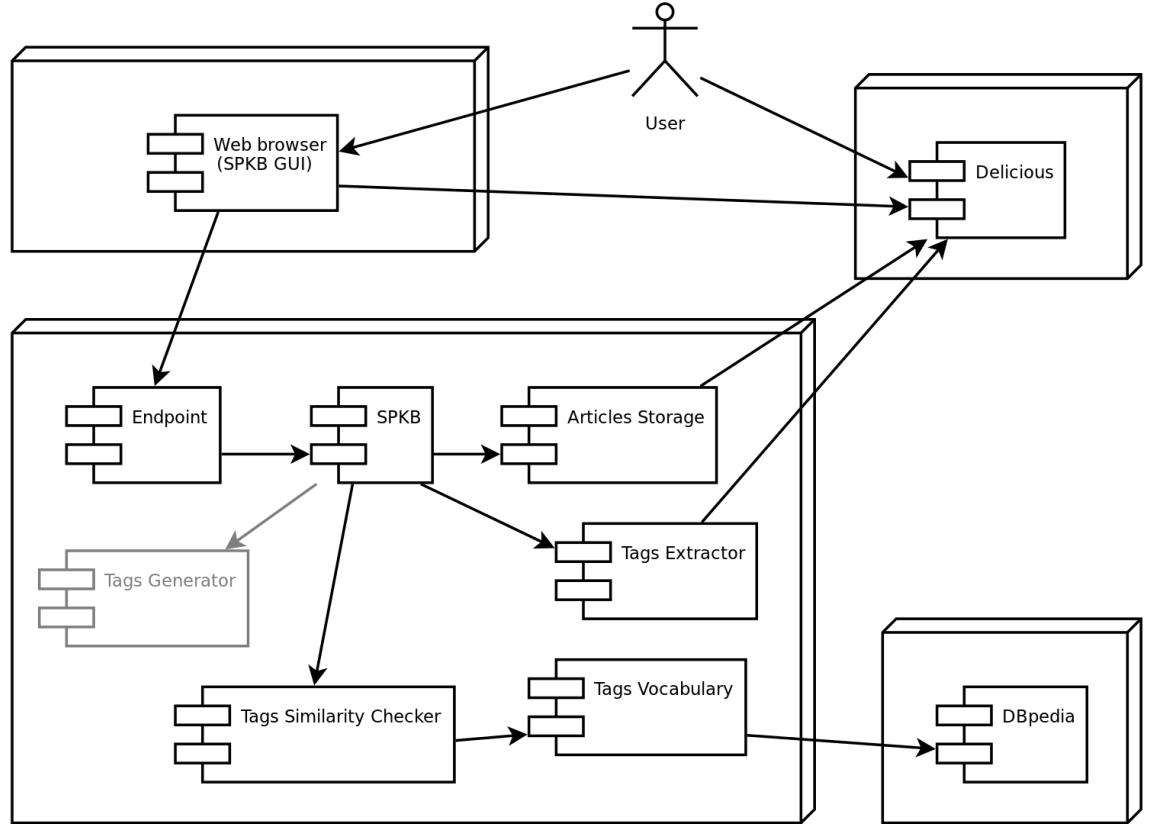


Figure 5.1 – The SPKB prototype architecture diagram

GUI. As was already outlined, SPKB user interface consists of the two parts, SPKB web page and Delicious tools. The former is intended for searches in SPKB (see Figure 5.2, selection number 1). The latter allows the users to add articles into the knowledge base (see Figure 5.2, selection number 2). This separation exists because of two reasons. The first one is the fact that client applications for adding and managing the knowledge base already exist and Delicious provides them for free. Therefore implementing the same functionality again does not make any sense, especially if the second reason is the fact that SPKB is just a prototype, sort of experimental, to verify the idea and at this stage it is not guaranteed that Delicious is the right choice for the Article Storage component.

Endpoint. Since SPKB user interface is only a thin client, everything except receiving user inputs and displaying the search results happens on the server. The endpoint component represents a HTTP API that the client uses for communicating with the server components. The endpoint is not intended for a public usage.

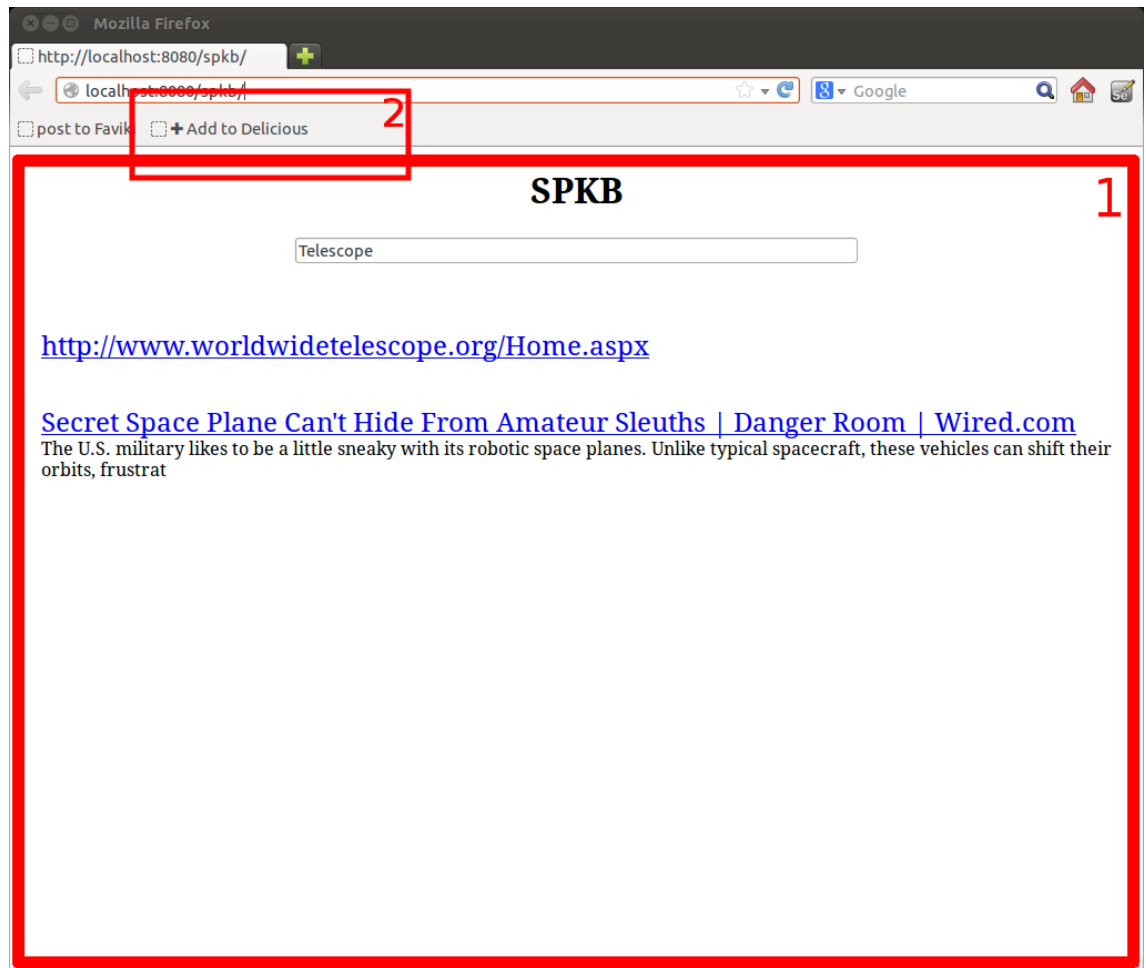


Figure 5.2 – SPKB user interface

SPKB. The SPKB component is a central component, a controller that executes and directs all the SPKB processes. Note that the SPKB component is ready for such an extension of the system that besides the popular tags related to the articles, also user assigned tags and content generated tags are supported. Tag Generator, a component responsible for generating tags by content is out of scope of this paper, however.

Articles Storage. SPKB does not use own database for storing articles and tags. These data are stored in Delicious, an external well known service with a huge user base. SPKB uses Article Storage for getting articles from Delicious. SPKB does not allow the users a direct access to Article Storage from SPKB client. Instead, the users needs to use Delicious client.

Tags Extractor. The key feature of SPKB is based on popular tags. Tag Extractor is a component responsible for obtaining the popular tags for articles. Tag Extractor requires Articles Storage, Delicious respectively, to provide a support for getting the popular tags.

Tags Similarity Checker. Tags Similarity Checker is a component that makes a decision whether two groups of tags are similar to each other or not. The first tag group are the entered tags, acquired through SPKB client. The second tag group are the popular tags of an investigating article. The similarity decision is made by the existence of SPKB relations between the tags. Tag Similarity Checker does not manage the relations between tags. It internally delegates discovering the tag relations to Tag Vocabulary.

Tags Vocabulary. When finding a SPKB relation between two tags is required, Tag Vocabulary is requested. Tag Vocabulary searches DBpedia concepts, looking for appropriate properties between the concepts and transforms them into SPKB relations. Tags Vocabulary does not store any data.

5.3 Design

This section explains some design decisions and possibly clarifies the reasons that led to the decisions which have been made. The description stays at a higher level. There is not an attempt to presents a division of the system to the classes, because such information may be understood either from the implementation documentation or source code.

5.3.1 Endpoint

A communication between SPKB client and SPKB server is based on HTTP protocol. The endpoint provides the only */api/findArticle.json* method whose invocation starts a search. As the name of the method implies the search results data are encoded into JSON⁴⁹ format.

⁴⁹<http://www.json.org/>

Table 5.1 – An example of the search results reduction

Entered tags	Search results
Car	A Future of Cargo Transport Is a Slow UFO
	Mercedes Benz introduces off-road of the future
Car, Terrain	Mercedes Benz introduces off-road of the future
Car, Cargo	A Future of Cargo Transport Is a Slow UFO

5.3.2 Articles Storage and Tags Extractor

A communication between Article Storage and Delicious, as well as between Tags Extractor and Delicious is realized through a public API provided by Delicious. In fact, both of the components are implemented as a single one. The reason is that they both need to communicate with the same external service and they both use the same communication method. The Delicious API is based on HTTP protocol and data may be provided either in the RSS⁵⁰ or JSON format. The JSON format has been chosen by SPKB.

5.3.3 Tags Similarity Checker

Similarity algorithm. If $E = (e_1, e_2, \dots, e_n)$ represents entered tags and $P = (p_1, p_2, \dots, p_m)$ popular tags then the algorithm for checking the similarity of the entered and popular tags works as follows:

$$E \text{ is similar } P \Leftrightarrow \forall e_i \exists p_j : e_i \text{ is similar } p_j; e_i \in E, p_j \in P$$

where two tags e_i and p_i are similar if and only if there exists an arbitrary SPKB relation between them.

This algorithm allows reducing the search result set when the number of hits is too large. The reduction is made by appending new tags to the set of entered tags (to the search form). Table 5.1 shows an example from the user point of view.

Case sensitivity. Since the popular tags are obtained from Delicious as lower cased, but labels of DBpedia concepts are case sensitive, at some point it is necessary to adjust the popular tags so that they equal to the case sensitive labels. Such adjustment is not trivial when the tag is assembled from multiple words. In such cases it is not clear which beginning letters should be transformed to upper case

⁵⁰<http://www.rssboard.org/rss-specification>

and which should not. The current implementation of Tags Similarity Checker uses a simple rule that only the very first letter of the tag is transformed to upper case. For example, *thomas jefferson* is transformed to *Thomas jefferson*. Even though that applying this simple rule does not lead to correct results 100 percent of times, it works pretty well in most cases. An incorrect transformation happened also in the example, since surname Jefferson should have started with the uppercase. It turned out, however, that such mistakes do not mind too much because “typos” of this kind are often redirected (dbpedia-owl:wikiPageRedirects) to the concept with the correct label. The redirection is understood as the synonym relation from SPKB point of view and therefore such incorrect transformation is covered by the SPKB relations.

Other approaches to the transformation have been tested but none of them was acceptable from performance point of view. They either generated too much upper case - lower case combinations, leading to an intolerable increase of the number of SPARQL queries, or make the evaluation of the queries too time consuming.

5.3.4 Tags Vocabulary

Instead of using own stored data, Tags Vocabulary queries DBpedia and gets required data on the fly. As Figure 5.3 shows, Tag Vocabulary retrieves data from DBpedia using its SPARQL endpoint. Tags Vocabulary uses a database of SPKB relation rules internally. The rule is basically a SPARQL query template with a specified SPKB relation for which the rule is valid. An example of one rule that is valid for the SPKB specification relation presents 5.1.

Tag Vocabulary sends the SPARQL queries, that are generated from the rules by Query Builder, over HTTP and the responses are received in the RDF/XML format.

Various modifications of the templates have been tested to improve the response times, or to resolve the problem with the case sensitivity that was described earlier, for example using filters and regular expressions, or using Virtuoso SPARQL Query Service⁵¹ built-in functions, such as *bif:contains*[13]. It turned out that none of the modifications brought acceptable results. Especially the performance was the issue. However, there is still space for optimizing the queries but at this

⁵¹DBpedia SPARQL Endpoint is realized by Virtuoso SPARQL Query Service.

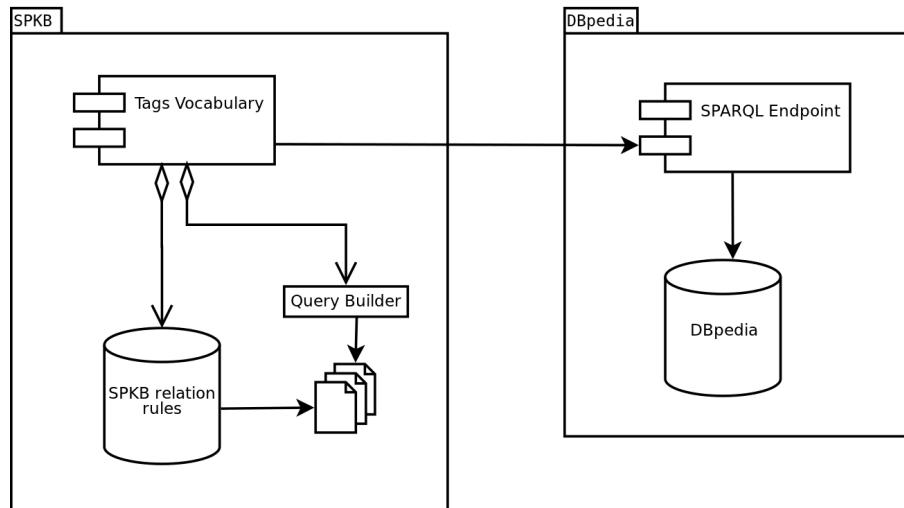


Figure 5.3 – Tags Vocabulary design

Algorithm 5.1 An example of a SPKB relation rule

Relation type:	SPECIFICATION
Rule name:	specification_skosBroader_skosBroader.sparql
Template content:	<See the program listing below>

```

PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
SELECT ?x ?y ?z WHERE {
    ?x rdfs:label "__LABEL1__"@en .
    ?x skos:broader ?y .
    ?y skos:broader ?z .
    ?z rdfs:label "__LABEL2__"@en .
} LIMIT 1

```

stage doing heavy optimizations is premature.

5.4 Build and installation

The build process is designed to get all all required dependencies automatically during the build. The list of the dependencies may be found in the build configuration file. Detailed instructions for building and running the prototype may be found on the attached CD, see Attachment 2: CD

Part of the building process is also testing. Various type of tests, such as unit tests, integration tests and also end to end tests, are run. On account of performance issues observed during the prototype testing and described in section 6.1.1, the project building may take long time. It depends on the current load of DBpedia.

6. Prototype evaluation

This chapter presents test methods that were used for testing the SPKB prototype. The first experience with using the prototype and obtained test results are evaluated, and further directions of the development are introduced.

6.1 Testing

Two tests were created to measure a search success rate, and precision and recall parameters of the SPKB similarity search feature. The testing has enforced some modification of SPKB, however, since performance issues with a significant impact to SPKB search had been found.

6.1.1 Performance issues

Originally, more extensive testing was planned to evaluate the prototype. At the very beginning of the testing process it was found, however, that there are serious performance issues which caused that testing would take much longer than is acceptable. Searching for the causes revealed that DBpedia is rather unreliable to be used in a way as was designed.

6.1.1.1 DBpedia

Maintenance. First of all, DBpedia is under maintenance more often than was expected to be. When DBpedia seemed to be down, its availability was being checked either directly by accessing a sample resource over the Web⁵², by querying DBpedia through online SPARQL endpoint⁵³, or by checking the availability at SPARQL endpoint status page⁵⁴.

Response times. Even if DBpedia was not completely down, the response times were quite often miserable. It was found that the response times were in range of tens of milliseconds to tens of seconds, in extreme cases even minutes, per a SPKB request. Such response times

⁵²<http://dbpedia.org/OnlineAccess#h28-13>

⁵³<http://dbpedia.org/sparql>

⁵⁴<http://labs.mondeca.com/sparqlEndpointsStatus/>

cause that a single SPKB search may take tens of minutes when there are only 10 articles stored in the database which are being searched through. Therefore, the testing in a range of hundreds of articles, as was originally planned, is practically impossible. For this reason, besides some performance measures in SPKB, testing scenarios were modified and number of articles significantly reduced, despite the fact that such a drastic reduction does not bring accurate and reliable results.

6.1.1.2 SPKB performance measures

In order to test the SPKB prototype, despite the slow response times of DBpedia, some SPKB specification/generalization and SPKB related relation property chains defined in the section 4.1 were disabled. After an examination of DBpedia’s data related to the defined property chains it was found out that some of them do not bring too much benefits and they may be disabled without too much costs. Having a smaller set of the SPKB relations property chains leads to a fewer DBpedia queries and to obtaining the SPKB search results faster.

It was observed that a redirection to a skos:concept typed entity does not bring too much new and useful tags. If entities like internal Wikipedia categories (for example CAT:HIST), “List” prefixed entities (for example List of mystics) or entities that may be obtained by other property chains are filtered out then there remains only a small portion of potential tags. The filtered cases may be ignored since users usually do not use such words as tags. For example, instead of List of mystics there is a better alternative Mysticism. The alternative may be obtained otherwise, without the redirection.

The redirection is used in several property chains of the SPKB specification/generalization and SPKB related relation. Specifically, the following list of the property chains derived from schemes illustrated in Figure 4.3, Figure 4.4 and the SPKB relations mapping (4.1) are disabled:

- $\text{TAG1_SYN} \xrightarrow{\text{wikiPageRedirects}} \text{TAG1} \xleftarrow{\text{broader}} \text{TAG2}$
- $\text{TAG1_SYN} \xrightarrow{\text{wikiPageRedirects}} \text{TAG1} \xrightarrow{\text{broader}} \text{TAG2}$
- $\text{TAG1_SYN} \xrightarrow{\text{wikiPageRedirects}} \text{TAG1} \xleftarrow{\text{subject}} \text{TAG2}$
- $\text{TAG1_SYN} \xrightarrow{\text{wikiPageRedirects}} \text{TAG1} \xleftarrow{\text{subject}} \text{TAG2} \xleftarrow{\text{wikiPageRedirect}} \text{TAG2_SYN}$
- $\text{TAG1_SYN} \xrightarrow{\text{wikiPageRedirects}} \text{TAG1} \xleftarrow{\text{broader}} X \xleftarrow{\text{subject}} \text{TAG2}$

- $\text{TAG1_SYN} \xrightarrow{\text{wikiPageRedirects}} \text{TAG1} \xleftarrow{\text{broader}} X \xleftarrow{\text{broader}} \text{TAG2}$
- $\text{TAG1_SYN} \xrightarrow{\text{wikiPageRedirects}} \text{TAG1} \xrightarrow{\text{related}} \text{TAG2}$
- $\text{TAG1} \xrightarrow{\text{related}} \text{TAG2} \xleftarrow{\text{wikiPageRedirects}} \text{TAG2_SYN}$
- $\text{TAG1_SYN} \xrightarrow{\text{wikiPageRedirect}} \text{TAG1} \xrightarrow{\text{related}} \text{TAG2} \xleftarrow{\text{wikiPageRedirects}} \text{TAG2_SYN}$

6.1.2 Single article search test

This test verifies the realization of all three use cases defined in the section 2.2. The purpose of the test is to identify any problems related to the integration of the SPKB components and causes of the problems. The test operates always with one article at a time since having the only article makes it easier to observe and analyze problems.

6.1.2.1 Testing method

In order to test the SPKB search feature it is necessary to have an article which is tagged by many users and to have a SPKB user who is looking for the article. The user uses his very limited memories of content of the article during the search. An example of a scenario like this was introduced in the section 2.1.1. The first requirement, having an article with popular tags is easily achievable since all that needs to be done is to find an popular Delicious bookmark which has been bookmarked by a lot of users. Thus, there exists a set of popular tags associated with the bookmarked article. The second requirement, having a SPKB user is tricky since the SPKB prototype is not intended for a real user testing. It is necessary to simulate the user, especially his limited memories of the article's content.

The selected method of the simulation is based on the fact that after 30 days people retain only a small portion of information they originally received. As [10], [11] point out it is between 2 and 20 percent. Therefore, it is reasonable to assume that the SPKB users will remember only a few keywords and key concepts of the article. For simulating such limited memories an automated tool for extracting keywords from content was used and obtained keywords were modified afterward. They were simplified and supplemented by synonyms or other semantically similar words. For selecting such words publicly available thesauri and common sense were used. Some of the supplemented words were added in addition to the obtained keywords

Table 6.1 – An illustration of the process of simulating user’s memories

Article	Brief description of content	Generated keywords	Remembered keywords
Astronomy Photographer of the Year 2011	The article presents winning photos of the Astronomy Photographer of the Year competition. The competition is run by the Royal Observatory Greenwich and Sky at Night Magazine. The photos capture various things, such as giant oval storms, remnants of a supernova explosion, details of Jupiter’s moons or beauty of Northern Lights.	astronomy, royal, observatory greenwich, surface of jupiter, images, supernova explosion, kukula, northern lights, astronomer, remnants, olivia, curtain, storms, photographer, photos	Astronomy Photography, Photos, Pictures Planets, Sun Competition

because they were assumed to be such words which would retain in the user’s memory for a long time after he read the article. Of course, this is a subjective matter that differs from individual to individual. Table 6.1 shows the process for one of the selected articles.

Yahoo! Term Extraction Service⁵⁵ has been used for generating keywords from content of the article. The service has been chosen because some users consider it to be low recall but high precision. Put another way, the service is expected to return a small number of keywords but returned keywords are precise. Some of relevant keywords are missing in the result, though. This is in the line with the method of simulating the SPKB user and his memories. Even though no research or benchmark solidly supporting this expectation has been found, retrieved keywords do not disprove it either.

The very same process has been repeated for 10 articles which were being tested one by one. Note that because of the performance issues the remembered keywords were adjusted with respect to a form of labels of DBpedia concepts. In other words, the first letter was upper

⁵⁵<http://developer.yahoo.com/search/content/V1/termExtraction.html>

cased and plural form of words were preferred.

The remembered keywords are those words which the SPKB user enters when he is searching for the article. The test records, alongside other data, which of the remembered keywords led to finding the article. The data are used for computation of a search success rate.

6.1.2.2 Search results

Comparing the search success rate among individual articles, see Table 6.2, shows that the success rate is rather unstable. In the best case (article #1) 71% of entered tags led to finding the article. On contrary to this good result there are cases (article #2 and #10) when SPKB did not find the given articles by none of the entered tags.

On average, a user needs to perform 4 searches, as Figure 6.1 shows, until the search results contains the requested article.

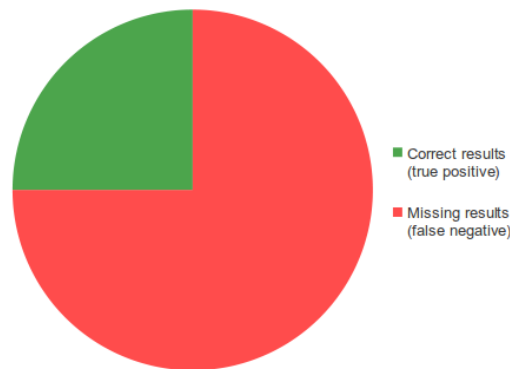


Figure 6.1 – Average search success rate

6.1.2.3 Examining results

Table 6.3 shows two tag groups, popular tags and entered tags, that were being compared during the testing. The table records those of the comparing tags which were found to be similar. The articles are ordered by the search success rate.

Table 6.2 – The search success rate values

#	Article	Expected positive search tags*	Search success rate (SSR)**
1	Astronomy Photographer of the Year 2011	<i>Astronomy, Photography, Photos, Pictures, Planets, Sun, Competition</i>	0.71
2	Is Sugar Toxic?	Corn, Syrup, Consumption, Obesity, Psychology, Hormones, Research, Sugar	0.00
3	Secret Space Plane Can't Hide From Amateur Sleuths	Spaceship, Africa, <i>Space</i> , Pilotless plane, Drone, Remote control, Sleuth, Tracking, Space, Shuttle, Secret, NASA	0.09
4	One Gene Lost = One Limb Regained? Scientists Demonstrate Mammalian Regeneration Through a Single Gene Deletion	Cells, Mouse, Regeneration, <i>Regenerating, Science</i> , Mammal, Genes, Salamander	0.25
5	A Physicist Solves the City	Taxi, <i>Physicists</i> , Traffic jams, Traffic, Urban, <i>Cities, Towns</i> , Rules, Universal rules, <i>Theories</i> , Laws, Natural laws, Chaos	0.31
6	Astronomy Picture of the Day	USA, Montana, <i>Thunderstorm</i> , Clouds, <i>Storm</i> Pictures, Images, Aliens, Wind, Spaceships, <i>Tornadoes, Photos</i>	0.33
7	British Amputee Cat First to Get Bone-Grafted Exoprosthetic Paws	<i>Prosthesis, Implants</i> , Accident, Cat, <i>Surgery</i> , Amputation, Paws	0.43
8	A woman's history of vaginal orgasm is discernible from her walk.	Orgasm, Vagina, Vaginal orgasm, <i>Health</i> , Gait, Movement, Walking, <i>Research</i>	0.25
9	Must Watch TED Videos	Ideas, Inspiration, Motivation, TED, Must see, towatch, <i>videos</i> , the best	0.13
10	Vine seeds become "giant gliders"	Vine, Vine seeds, Seeds, Wings, BBC, Java, Indonesia, Aircraft design, Aerodynamics, Glider	0.00

* Emphasized tags represents observed positive searches; ** Search success rate = $\text{count}(\text{observed positive searches}) / \text{count}(\text{expected positive searches})$

Table 6.3 – Found similarities among popular and entered tags

Article no.	Popular tags	Entered tags	Found similarities	Search success rate
1	astronomy, photography, space	Astronomy, Photography, Photos, Pictures, Planets, Sun, Competition	(photography, Photography) (photography, Photos) (photography, Pictures) (astronomy, Planets)	0.75
7	science, medicine, medical	Prosthesis, Implants, Accident, Cat, Surgery, Amputation, Paws	(medicine, Prosthesis) (medicine, Implants) (medicine, Surgery)	0.43
6	storm, photo, science	USA, Montana, Thunderstorm, Clouds, Storm Pictures, Images, Aliens, Wind, Spaceships, Tornadoes, Photos	(storm, Thunderstorm) (storm, Storm) (storm, Tornadoes) (photo, Photos)	0.33
5	science, physics, cities	Taxi, Physicists, Traffic jams, Traffic, Urban, Cities, Towns, Rules, Universal rules, Theories, Laws, Natural laws, Chaos	(science, Physicists) (cities, Cities) (cities, Towns) (science, Theories)	0.31
4	science, regeneration, health	Cells, Mouse, Regeneration, Regenerating, Science, Mammal, Genes, Salamander	(regeneration, Regenerating) (science, Science)	0.25
8	science, sex, research	Orgasm, Vagina, Vaginal orgasm, Health, Gait, Movement, Walking, Research	(science, Health) (science, Research)	0.25
9	videos, video, technology	Ideas, Inspiration, Motivation, TED, Must see, towatch, Videos, the best	(video, Videos)	0.13
3	space, science, astronomy	Spaceship, Africa, Space, Pilotless plane, Drone, Remote control, Sleuth, Tracking, Space, Shuttle, Secret, NASA	(space, Space)	0.09
2	cancer, diet, cholesterol	Corn, Syrup, Consumption, Obesity, Psychology, Hormones, Research, Sugar		0.00
10	science, nature, biology	Vine, Vine seeds, Seeds, Wings, BBC, Java, Indonesia, Aircraft design, Aerodynamics, Glider		0.00

Too few popular tags. Regardless of the search success rate value, it is possible to see an imbalance of cardinality of the compared tag sets. The imbalance is caused because Delicious returns at most three popular tags for an article. The lower cardinality of the popular tag set, the lower chance to find a similarity between the popular tags and an entered tag. The situation is even worse in cases like the article #9 when two of the given popular tags are basically equal. Specifically those tags are *video* and *videos*. If none of the entered tags had been similar to the former one, there is only a small chance that it would have been similar to the latter. Increasing the number of popular tags could lead to the higher search success rate.

If one of the comparing tag sets had contain fewer tags, it should be the entered tag set. The reason is that the entered tag set contains tags which were entered by a user when the user was searching for an article. In other words, the more tags the entered tag set contains, the more (unsuccessful) searches of the same article have been performed.

Not found similarities. A closer examination of the tag groups reveals that some tag pairs which may be intuitively considered to be similar or related SPKB does not evaluate in the line with the expectations. For example, no relation was found between *NASA* and *space*, *science* or *astronomy*. Another example is missing relations between *diet* and *Obesity*, or *diet* and *Sugar*. The article #3 or #2 are examples showing that when the *too few popular tags* and *not found similarities* characteristics meet each other, the search success rate is very low.

Tags selection. It is believed that there are other matters related to the tags selection, either those provided as the popular tags or those entered by the SPKB users, that significantly affects the result search success rate. One of such matters, too excessive similarity of tags, was already outlined (*video* and *videos*). This may happen especially in a relation to singularity and plurality of tags, when one group of users uses the singular version of a tag while the other group of users prefer the plural version of the tag. SPKB considers both of the tag versions to be synonyms, since DBpedia usually redirects the singular form to the plural, therefore chances that SPKB finds new relations between comparing tags are lower than if the popular tag set contains less similar tags.

Another assumed complication from the perspective of searching for the similarities between tags is when the user enters such tags which although can be considered to be similar, but they are semantically too far away from each other. An example of such a case is the article #10. The popular tags are too high level in comparison to the entered tags. Specifically, *Vine* or *Seeds* are definitely specification of *biology*, but there are too many levels between these concepts for SPKB to be able to detect the relation. The SPKB specification/generalization property chain between *Vine* and *biology* is the following: $\text{Vine} \xrightarrow{\text{subject}} \text{Plants} \xrightarrow{\text{broader}} \text{Botany} \xrightarrow{\text{broader}} \text{Biology}$. Such a long chain is not supported by SPKB.

In order to prove or disprove the afore assumptions it is necessary to do much extensive testing and get statistically relevant data. The scope of this test is insufficient for making any conclusions related to the tag selection and its influence to the search success rate.

6.1.2.4 Comparison with Delicious search

The Delicious' internal search engine supports the full-text search and tags based search. Comparing the results obtained by the particular search types shows that SPKB may help to reduce irrelevant articles and on contrary add missing relevant articles to the search results.

Full-text search results. The full-text search searches through articles description. The description is either entered by the users or is automatically added by Delicious. A comparison of SPKB search results with Delicious full-text search results shows that the latter may contain also such articles which do not fully suit the original intention. For example, if a user wants to find all astronomy articles and hence he enters "Astronomy" into the search form, Delicious returns also the article no. #6 (see Table 6.2) since the description of the article contains the requested word. In fact, the content of the article is actually not too much related to astronomy. Instead, the article shows a photograph of a supercell thunderstorm cloud over Montana. When SPKB is requested to find all astronomy articles using the same search query, this article is not included into the search results since SPKB search is based on a human content verification.

Tags based search results. SPKB allows a user to find an article even if there is not an exact match between a popular tag and an entered tag. Delicious requires an exact match. For example, if a Delicious user looks for an article by the *Planets* tag, even though the content of the article no. #1 (see Table 6.2) is related, the article is not included in the Delicious search results. The reason is that *Planets* does not exactly match to *astronomy*, *photography* or *space*. SPKB includes this article into the search results since SPKB recognizes the similarity between *astronomy* and *planets*, as Table 6.3 shows.

On the other hand, Delicious does not need to support the similarities recognition. Delicious is based on the idea that users tag their articles and therefore they know what tag they should use when they want to find the article. SPKB is based on a different idea. The users do not need to tag their articles. Such users then rely fully on those who tag the articles. The SPKB users do not have a knowledge about tags which were assigned to the articles by the other users. Therefore, the exact match is not an effective method for SPKB and its users, even though it may work for Delicious users.

6.1.3 Precision and recall test

Both, precision and recall are values or properties of an search algorithm that express the ability of the algorithm in retrieve relevant documents from a given collection of documents. The purpose of this test is to measure precision and recall of SPKB for a defined testing scenario that is supposed to simulate a SPKB user.

Precision. Precision is the fraction of retrieved articles that are relevant. A perfect score 1.0 means that all retrieved articles are relevant. Precision does not say anything about how many of all relevant document were retrieved. Formally, precision is defined as follows:

$$precision = \frac{|\{\text{relevant articles}\} \cap \{\text{retrieved articles}\}|}{|\{\text{retrieved articles}\}|}$$

Recall. Recall is the fraction of relevant articles that are retrieved. A perfect score 1.0 means that all relevant articles were retrieved.

Recall does not say anything about how many irrelevant articles were retrieved. Formally, recall is defined as follows:

$$recall = \frac{|\{relevant\ articles\} \cup \{retrieved\ articles\}|}{|\{relevant\ articles\}|}$$

Relevance. Since the term *relevant articles* has been used in context of precision and recall, it is good to know how to decide whether an article is relevant or not. In context of this paper, relevancy is understood in accordance with the following definition:

“The extent to which information retrieved in a search of a library collection or other resource, such as an online catalog or bibliographic database, is judged by the user to be applicable to ("about") the subject of the query. Relevance depends on the searcher’s subjective perception of the degree to which the document fulfills the information need, which may or may not have been expressed fully or with precision in the search statement.”[12]

A crucial part of the definition is when it says that the relevance depends on the subjective perception. All decisions related to relevancy of test articles were made on the subjective perception of a person who read a given article. The perception does not necessarily need to correspond with the popular tags assigned to a given article.

6.1.3.1 Testing method

This test uses the same articles as the previous one, but this time the number of the stored articles was doubled. To the articles that were used in the previous test were added new 11 articles. The added articles are shown in Table 6.4. All the articles used in this test are bookmarked also by other users so Delicious provides the most popular tags for each of them. On contrary to the previous test, this time the searches are performed against all articles at the same time.

The articles are thematically selected so that each of them belongs into a specific category. The selection is made hierarchically, such that if an article belongs into a lower level (more specific) category then it belongs also to the upper level category. Table 6.5 show the distribution of the articles into categories. Since a vital factor for a quality of the search results is quality and scale of DBpedia’s data,

Table 6.4 – A list of articles specific for the Precision and recall test

#	Article	Popular tags
11	Google Moon	moon, google, space
12	Qu8k	rockets, rocket, qu8k
13	Why Can't We Drink Sea Water?	saltwater, salt, life
14	JOURNEY OF MANKIND - The Peopling of the World	history, anthropology, evolution
15	WorldWide Telescope	astronomy, science, space
16	Optical Illusions and Visual Phenomena	illusions, optical, illusion
17	Popular Science New Technology, Science News, The Future Now	science, technology, magazine
18	Scale of universe	universe, scale, physics
19	What's Special About This Number?	math, numbers, mathematics
20	Hubble Telescope - Blank Note Cards - HRH1	blank note cards, note cards blank, hubble
21	The Last Question by Isaac Asimov	fiction, story, science

Table 6.5 – A distribution of the articles into the categories

	Category			
	Unspecified (total number)	Science	Astronomy	Telescope
Number of articles	21	19	6	2

which are not of the same quality for general and more specific topics, the purpose of the categories is to compare precision and recall under these different circumstances.

For each of the categories there were defined such keywords which a SPKB user could use when he is looking for the articles of that kind. The keywords has been selected with a respect to content of the stored articles. The keywords selection is based on a common expectation of such keywords which could lead the user to the given article. The selected keywords (listed in Table 6.6) were used one by one as search queries.

6.1.3.2 Results

Table 6.6 presents the precision and recall values obtained by the testing. Figure 6.2 shows average values of the precision and recall in a relation to the number of relevant articles stored in the database during the testing.

Table 6.6 – The precision and recall values

Search query	Precision	Recall
Science	0.83	0.52
Astronomy	1.00	0.83
Space	0.55	0.83
Universe	0.55	0.83
Telescope	0.33	0.50
Telescopes	0.33	0.50
Hubble telescope	0.00	0.00
Scope	0.00	0.00

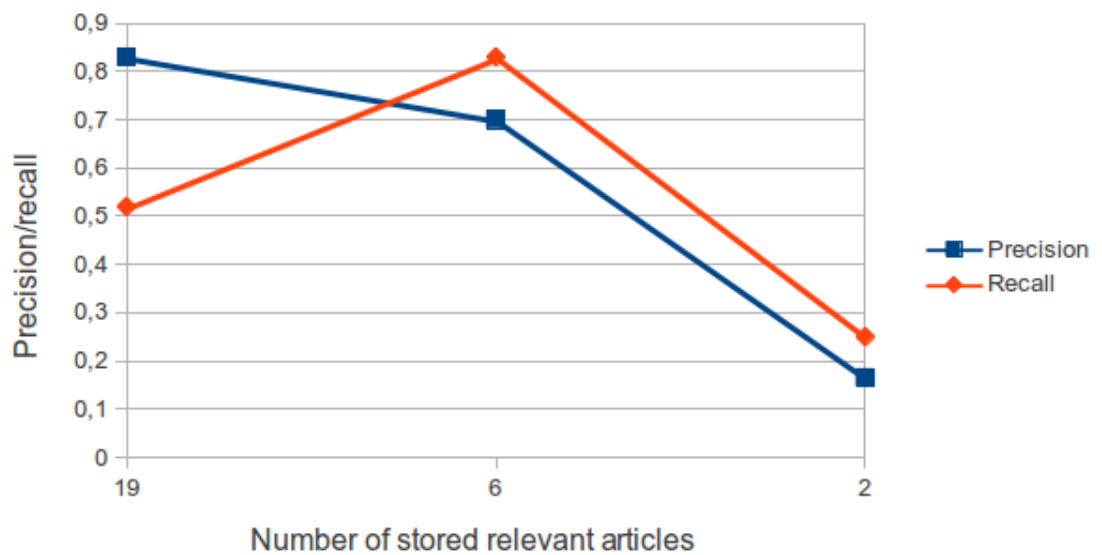


Figure 6.2 – Average precision and recall in relation to number of relevant articles

Precision. High precision value suggests that once the article is part of the search results, there is a good chance that the article is really relevant. It happens only rarely that an article is declared to be relevant when it is not. High precision can come out of the fact that popular tags assigned to the articles are created by the users who understand the content of the article. Thus, the assigned tags are precise. A strong deterioration of the precision along with the number of all relevant articles may be explained by a higher sensitivity of this value to irrelevant articles, when there are only a few relevant articles in the whole database. In other words, an incorrect evaluation of the single article causes more damage than when there are a lot of other relevant articles in the database which may appear in the search results. However, in order to prove whether it was only a coincidence related to the selected tags, or whether it is typical for the SPKB prototype, more testing is needed.

Recall. In order to understand what the obtained recall value suggests it is important to realize that a high recall in context of SPKB means that majority of the users tagged the article so that the tags are similar to the tag(s) entered by the SPKB user when he is searching for the article. This is not always the case. An article may be tagged well, assigned tags may represent a proper description of content of the article, but the SPKB user enters such tags which are much more specific or general in comparison to the popular tags. An example was already provided when the results of the previous test were being examined. A closer look at the popular tags of this test's articles (see Table 6.4) shows that the same situation may happen when science articles were being searched. Some of the popular tags are too specific in order to be evaluated as similar with *Science*. It led to a lower recall value afterward.

Examining DBpedia's data, specifically concepts with relations to *Science*, *Astronomy* and *Telescope* reveals that there is much higher potential for using related concepts of *Astronomy* as tags, than in the case of other two concepts. For example, it is more probable for *Space*, *Astronomers* or *Universe* to be used as tags than *Scientific knowledge*, *Science in society*, *Astronomical instruments* or *Optical devices*. Since SPKB searches DBpedia and looks for concepts related to entered tags and then compares found concepts to the popular tags, it is important for the found concepts to have a potential for being used as tags. If

the found concepts do not have this potential, for example *Science in society*, then users do not use the concepts as tags and SPKB won't find any match. The point here is to show the importance of DBpedia's data for SPKB search results. The chart on Figure 6.2 shows that the highest recall value was reached when was searched for articles from the field of astronomy. The lack of tag potential concepts related to Science and Telescope decreased the recall value when articles from the other two fields were being searched.

6.2 Drawbacks and possible improvements

Since SPKB prototype is a proof of concept implementation it reveals some issues which need to be solved in order to use the system publicly. This section summarizes the most significant drawbacks and presents some possibilities of further development of SPKB.

6.2.1 Long taking search

The biggest of the drawbacks is that each SPKB search takes much longer that may be accepted for a real world usage. For a real user testing, or even to measure more accurate search success rate, precision and recall, it is necessary to optimize the search process.

Currently, each time a SPKB user performs a search (and no results is found), SPKB sends $2 \cdot |ARTICLE \times POPULAR \times ENTERED \times RULES|$ queries to DBpedia, where ARTICLE represents the set of the user's articles; POPULAR represents the set of popular tags of a currently examining article; ENTERED represents the set of tags entered by the user into the search form and RULES represents the set of SPKB rules that are used for finding a similarity between the popular and entered tags. Having the only stored article and for the current configuration of the Delicious and SPKB prototype it is 52 requests.

There are several methods whose incorporating into SPKB may decrease the search time:

- bulk querying
- private DBpedia instance
- SPKB relations compliant data model derived from DBpedia

Note that not all of these methods are fully compliant with the defined SPKB requirements.

Bulk querying. It was calculated that there are a lot of SPARQL requests sent to DBpedia during the search. Sending requests in a bulk can reduce an overhead related to creating and processing the requests. On the other hand, such a change requires major modification to Article Storage.

Private DBpedia instance. Having an own DBpedia instance would precede such problems as often maintenance or slow response because of a heavy load. On the other hand, it brings a new issues related to administrating the service and to the necessity to keep DBpedia data updated with Wikipedia. In this matter DBpedia Live⁵⁶ may help.

SPKB relations compliant data model. Having a data model adapted specially to needs of SPKB can significantly reduce the number of sent requests. One method how to create such a model is to use DBpedia's data, find all paths matching the SPKB relations property chains and save found paths as a corresponding SPKB relations as the new model. For example, if there is the Vine $\xrightarrow{dcterm:subject}$ Plants $\xrightarrow{skos:broader}$ Botany property chain in DBpedia then it may be saved as Vine $\xrightarrow{spkb:specificationOf}$ Botany, or even Vine $\xrightarrow{spkb:similar}$ Botany. The new data model would contain only 4 properties between concepts, or even the only one if the latter alternative is applied.

6.2.2 Too few popular tags

Having too few popular tags decreases a chance for finding the similarities. The tests revealed that Delicious provides only three popular tags per an article. Since Delicious provides besides the popular tags also another set of tags, called recommended tags, adding the support of the recommended tags into SPKB may resolve the problem of too few popular tags. However, the recommended tags are not a result of public knowledge.

⁵⁶<http://live.dbpedia.org/>

6.2.3 No popular tags

Current implementation of the SPKB prototype is powerless when there is an article without any popular tags. Such situation happens when there is too few users who bookmarked the article. In order to use SPKB in such situations, a new component, Tags Generator, may be added to SPKB. Nevertheless the SPKB prototype in the current form focuses specifically to use of human knowledge, it is prepared for adding the Tags Generator component. The automatic generating of tags may be ignored as long as there are other options, for example a presence of popular tags. If it is not the case, then some tags may be generated from the content, instead of ignoring the article. A suitable existing service for the Tag Generator component is OpenCalais Web Service⁵⁷ which generates tags based on Wikipedia articles.

⁵⁷<http://www.opencalais.com/>

Conclusion

This paper introduces an inefficiency of common solutions when a repetitive search for already found information is requested. An alternative solution based on utilizing collective knowledge and semantic similarities between tags is proposed. The alternative is implemented in the form of a prototype. The prototype presents an option when a user does not need to pay too much attention to maintaining his knowledge base. When the user needs to find an article, which he previously found as useful, the proposed solution allows him to look for the article using his limited memories of the content in the form of keywords or tags. Once the keywords are entered, similarities between them and the most popular tags extracted from the other users are being searched. The similarities between the tags allow the article to be considered as suitable to the entered keywords, on contrary to full-text based search methods, even when none of the keywords is presented within content of the article. The prototype uses existing technologies and services. Delicious, a bookmarking system is used as an article storage. DBpedia is used for searching similarities between the entered keywords and popular tags. Since content of both of these services is a result of collective knowledge, their incorporating into the prototype is in line with the defined goals.

The most important contribution of the paper is an illustration that by combining human content recognition, masses of users and incorporating keyword similarities into the search process may be achieved easier repetitive searching for information, without the user's involvement to the process of organizing the information. Moreover, the proposed solution may lead to more accurate search results than are obtained by an internal Delicious search engine based exclusively on a full-text or exact match between tags search.

The goals of the thesis were achieved partially. Even though, the proposed solution may really make the repetitive search easier and all three defined requirements, using a personal knowledge base, utilizing public knowledge in organizing information and supporting similarity between tags, were satisfied, the prototype also revealed some serious performance issues. In order to prove that the proposed solution is usable in practice and that the search results are better than existing solutions may provide, more extensive user testing is required. Such

extensive testing is not possible, though until search times in range of seconds are not achieved. Such search times are beyond the capabilities of the prototype. The paper also suggests further development options and methods that may lead to resolving the performance issues.

Bibliography

- [1] DAVIES Stephen, Scotty ALLEN, Clayton LEWIS. Popcorn: the personal knowledge base. Computer Science Education Labs, 2006. Retrieved: https://csel.cs.colorado.edu/~daviessc/writings/Popcorn_the_personal_knowledge_base.pdf. University of Colorado, Boulder.
- [2] Luis von Ahn, Ben Maurer, Colin McMillen, David Abraham and Manuel Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. 2008. Science 321 (5895): 1465–1468.
- [3] Luis von Ahn. NOVA ScienceNow s04e01 (Television production), 2009. Event occurs at 46:58. Retrieved: <http://www.youtube.com/watch?v=R7Lp00oGU7k>.
- [4] OWANO, Nancy. Duolingo launches as crowd-sourced translation service. PhysOrg.com, June 19, 2012. Retrieved: <http://phys.org/news/2012-06-duolingo-crowdsourced.html>
- [5] The Bing Team. Foursquare Arrives on Bing [Weblog entry]. June 18, 2012. Retrieved August 9, 2012, from http://www.bing.com/community/site_blogs/b/search/archive/2012/07/18/foursquare-.aspx.
- [6] Robu V., Halpin H., Shepherd H. Emergence of consensus and shared vocabularies in collaborative tagging systems, ACM Transactions on the Web (TWEB), Vol. 3(4), art. 14, 2009.
- [7] Golder, Scott; Huberman, Bernardo. Usage Patterns of Collaborative Tagging Systems, 2006. Journal of Information Science 32 (2): 198–208. Retrieved: http://pkudlib.org/qmeiCourse/files/Golder_usage_patterns_collaborative_tagging.pdf.
- [8] Dictionary.com - DevCenter API. *Thesaurus.com* [online]. 2013 [cit. 2013-04-10]. Retrieved from: <http://content.dictionary.com/api>

- [9] Delicious Terms of Service. In: <https://delicious.com/terms>. January 10, 2013.
- [10] UNIVERSITY OF WATERLOO. Curve of Forgetting. Retrieved from: <https://uwaterloo.ca/counselling-services/curve-forgetting>
- [11] EBBINGHAUS, Hermann. Memory: A Contribution to Experimental Psychology. New York city: Teachers college, Columbia university, 1913. ISBN 978-1614271666.
- [12] REITZ, Joan M. Dictionary for library and information science. Westport, Conn: Libraries Unlimited, 2004, p. 568. ISBN 1591580757.
- [13] OPENLINK SOFTWARE DOCUMENTATION TEAM. OpenLink Virtuoso Universal Server: Documentation [online]. OpenLink Software, 1999 - 2009, p. 1845 [cit. 2013-07-29]. Retrieved from: <http://docs.openlinksw.com/pdf/virtdocs.pdf>

Attachments

Attachment 1: Experiments with reasoning

This experiment used the Apache Jena⁵⁸ reasoner and rules engine for creating an inference model containing a selected SPKB relations. The SPKB relations are created from the SKOS relations between concepts defined in the original model. NTK PSH was used as the original model. The inference rules for generating the SPKB specification relations, and the inference model visualization is shown in Figure 6.3.

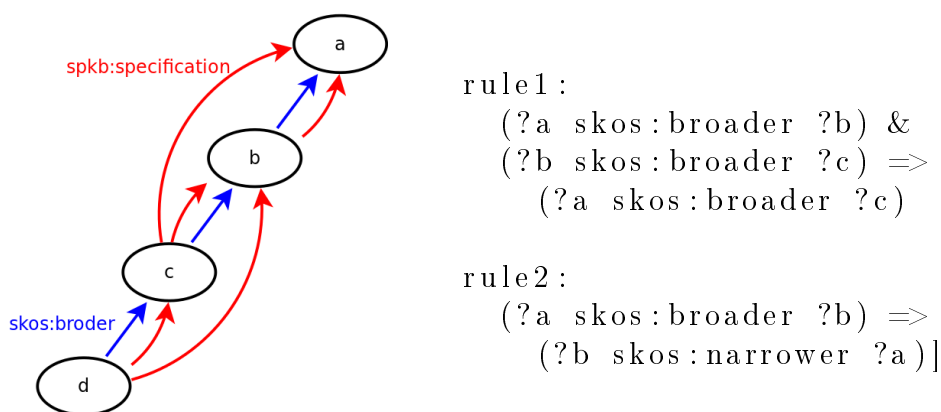


Figure 6.3 – Inference model rules

There was a restriction to maximum length of the path created by *skos:broader* properties between concepts. The maximum length was set to 2. Creating the inference model for longer allowed paths takes significantly longer time. Moreover, similarity of words hidden behind the concepts when the path between the concepts is longer is questionable. Algorithm 6.1 shows the rules written in Apache Jena rule syntax.

⁵⁸<http://jena.apache.org/>

Algorithm 6.1 Inference model rules written in Apache Jena rule syntax

```
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.
@prefix test: <http://test/res/>.
```

```
[isSpecification-1:
    (?a skos:broader ?b) ->
    (?a test:isSpecification-1 ?b)]
[isSpecification-2:
    (?a skos:broader ?b) (?b skos:broader ?c) ->
    (?a test:isSpecification-2 ?c)]
[isSpecification-from1:
    (?a test:isSpecification-1 ?b) ->
    (?a test:isSpecification ?b)]
[isSpecification-from2:
    (?a test:isSpecification-2 ?b) ->
    (?a test:isSpecification ?b)]
```

NTK PSH contains approximately 13500 headwords. The process of creating the inference model with the only SPKB specification relation in addition to the properties in the original model, took more than 200 seconds.

Attachment 2: CD

Part of the thesis is an attached CD. The content of the CD is the following:

- electronic version of this document in Portable Document Format (PDF)
- source code of the prototype
- distribution package the prototype
- the prototype programming documentation (JavaDoc)
- installation manual

The root directory on the CD contains the *Readme.txt* file. The file describes the directory structure of the CD and references to individual parts of the content listed above.