



University of Bergen

Master thesis

Using RDFa to reduce privacy concerns for personal web recommending

Author:

Christoffer M. Valland

Supervisor:

Andreas L. Opdahl

Department of Information Science and Media Studies

June 2015

University of Bergen

Abstract

Faculty of Social Sciences

Department of Information Science and Media Studies

Master's degree

Using RDFa to reduce privacy concerns for personal web recommending

by - Christoffer Valland

The amount of available information on the web is increasing, and companies are expanding the way to both collect and use the information available. This is the situation for both personal information, and technological information such as HTML-documents. Throughout this paper, I will describe the development of a semantic web recommender system that aims to reduce the amount of personal information needed to provide personal web recommendations. Semantically marked up documents on the web contain information, which is not necessarily provided in a user interface. This means there are possibilities to expand the area of use for this technology. The use of Semantic Web-technologies can therefore contribute to reduce the need of giving away personal information on the web.

This thesis is divided in two parts: The first part focuses on the development of a semantic application, and the new area of use of this technology. The other part focuses on how standard recommenders handle privacy concerns on the web. The thesis will provide a description of the development of the recommender system, as well as an explanation of online privacy and how different web service providers' deals with it. The system uses an RDFa-API to collect semantic information available on web-documents, and further uses this information to provide recommendations for the users. This thesis concludes that it is possible to recommend new web content for a user with this method, but the collected information varies wildly. This is related to both the complexity of the developed system and the way "things" are marked on the web. It is further shown that this method can reduce personal information, however it is shown that users who are comfortable with social medias are not worried about privacy on the web.

Acknowledgement

First of all I would like to thank my supervisor Andreas L. Opdahl. He has been a great help and motivator. Andreas' knowledge about the field is inspiring, and his feedback and guidance on this thesis have been crucial for me during these semesters.

I would like to thank my family, Live and Eskil, for being patient with me during the long days at the study room.

In the end I would like to thank my fellow students and friends at reading room 637. You have been entertaining, motivating and helping me a lot throughout these years.

Thank you all!

Table of content

1. INTRODUCTION	1
1.1. MOTIVATION	2
1.2. RESEARCH QUESTION	4
2. THEORY	5
2.1. PRIVACY	5
2.1.1. Informational privacy	5
2.1.2. Google's privacy policy	6
2.1.3. Users privacy concerns	7
2.2. THE SEMANTIC WEB	8
2.2.1. RDF – Resource Description Framework	8
2.2.2. Microformats	10
2.2.3. RDFa – Resource Description Framework in attributes	10
2.2.4. Usage of markup on the web	11
2.3. COLLECTING INFORMATION, COMMON TECHNIQUES	12
2.3.1. User-provided information	12
2.3.2. Cookies	14
2.3.3. Click tracking	15
2.4. OTHER RECOMMENDERS AND CONTENT PROVIDERS	16
2.4.1. RSS	16
2.4.2. Flipboard	17
2.4.3. Facebook Instant Articles	17
3. TECHNOLOGIES	18
3.1. JAVASCRIPT	18
3.2. JQUERY	18
3.3. MONGODB AND MONGODBLAB	18
3.4. CHROME EXTENSION	18
3.4.1. Chrome Extension Manifest	19
3.4.2. Browser- or page-action	19
3.4.3. Background or content script	20
3.5. GREEN TURTLE	20
3.6. GIT AND GITHUB	21
3.7. SPIDER	21
4. METHODS	22
4.1. DESIGN SCIENCE	22
4.1.1. Design as an Artifact	23
4.1.2. Problem Relevance	23
4.1.3. Design Evaluation	24
4.1.4. Research Contributions	24
4.1.5. Research Rigor	24
4.1.6. Design as a Search Process	25
4.1.7. Communication of Research	25
4.2. DEVELOPMENT METHOD – RUP (RATIONAL UNIFIED PROCESS)	25

4.2.1.	<i>Develop software iteratively</i>	26
4.2.2.	<i>Manage requirements</i>	26
4.2.3.	<i>Use component-based architectures</i>	26
4.2.4.	<i>Visually model software</i>	27
4.2.5.	<i>Verify software quality</i>	27
4.2.6.	<i>Control changes to software</i>	27
5.	IMPLEMENTING THE EXTENSION	28
5.1.	RESEARCH	28
5.2.	DEVELOPMENT	29
5.2.1.	<i>Iteration 1 – Pre-programming work</i>	29
5.2.2.	<i>Iteration 2 – Modifying Green Turtle</i>	31
5.2.3.	<i>Iteration 3 - Database</i>	32
5.2.4.	<i>Iteration 4 – Database (continues)</i>	33
5.2.5.	<i>Iteration 5 – User Interface</i>	34
5.2.6.	<i>Overview of the complete system</i>	36
5.3.	DATA FLOW / INFORMATION FLOW	36
6.	ANALYSIS AND DISCUSSION	38
6.1.	WHY RDFA?	38
6.2.	THE EXTENSION IN USE	39
6.2.1.	<i>Limitations</i>	39
6.2.2.	<i>Clearing the database</i>	40
6.2.3.	<i>Installing the extension</i>	40
6.2.4.	<i>In use</i>	41
6.2.5.	<i>Result of use</i>	42
6.3.	SOLVED PRIVACY ISSUES	44
6.3.1.	<i>Storing information</i>	45
6.3.2.	<i>Reducing personal information</i>	45
6.4.	UNSOLVED PRIVACY ISSUES	46
6.5.	WHY IT STANDS OUT FROM THE CROWD	46
6.5.1.	<i>Compared to RSS</i>	47
6.5.2.	<i>Compared to Facebook Instant Articles</i>	47
6.5.3.	<i>Compared to Flipboard</i>	47
6.6.	FUTURE WORK AND IMPROVEMENTS OF THE EXTENSION	48
6.6.1.	<i>Collecting information</i>	50
6.6.2.	<i>Recommending</i>	54
6.6.3.	<i>Searching</i>	55
6.6.4.	<i>Posting objects to the database</i>	56
6.6.5.	<i>Improving User Interface and User experience</i>	57
6.7.	EXPANDING THE VISION OF THE SYSTEM	58
6.8.	EVALUATION OF RESEARCH METHODOLOGY	61
6.8.1.	<i>Solving a problem</i>	61
6.8.2.	<i>Changing the way to solve a problem</i>	62
6.8.3.	<i>Evaluating the artifact</i>	62
6.8.4.	<i>Contributing to research</i>	63
6.8.5.	<i>Research Rigor</i>	63
6.8.6.	<i>Search process</i>	63

6.8.7. <i>Communicating the research</i>	63
6.9. EVALUATING THE DEVELOPMENT METHODOLOGY	64
7. SUMMARY AND CONCLUSIONS	65
8. SOURCES	67
9. APPENDIX	71
9.1. APPENDIX 1: DESIGN-SCIENCE RESEARCH GUIDELINES	71
9.2. APPENDIX 2: DESIGN EVALUATION METHODS	72
9.3. APPENDIX 3: INFORMATION SYSTEMS RESEARCH FRAMEWORK.....	73
9.4. APPENDIX 4: LIST OF LITERALS FROM IRENE CELINO'S TEST SITE.....	74

List of Figures

Figure 2.1: Example of RDF triples shown in a graph.....	9
Figure 2.2: Difference between browsers and humans, collected from: http://www.w3.org/TR/xhtml-rdfa-primer/	11
Figure 2.3: Graph showing the spread of markup-methods, from Webdatacommons.org (Bizer et al., 2014).....	12
Figure 2.4: Komplett.no's online registering sheet.....	13
Figure 3.1: Screenshot of icon used in browser-action marked with a ring.....	19
Figure 3.2: Screenshot of icon used in page-action marked with a ring.....	20
Figure 5.1: Image of my Kanban board late in the development phase.....	29
Figure 5.2: Screenshot of the User Interface.....	35
Figure 5.3: Data- and Information-flow.....	37
Figure 6.1: Number of documents in each collection.....	40
Figure 6.2: The button to load unpacked extension.....	41
Figure 6.3: Screenshot of the list of most common objects. Example of RDFa-objects not understandable for users.....	43
Figure 6.4: Screenshot of the result of searching for the object marked in Figure 6.3.....	44
Figure 6.5: Extending the architecture of the system.....	50
Figure 6.6: Sketch of a suggested new design.....	58
Figure 6.7: ITavisen.no's approach to marking articles with topics.....	59
Figure 6.8: Chrome's overview of my "most visited" sites.....	60

Chapter 1

1. Introduction

Given the fact that most people spend many hours of web surfing each week, we can say that the web has become a quite central part of our everyday-life. The amount of information and applications increases all the time, giving us even more reasons to spend time on digital platforms. What we also see online is the amount of information being collected about the users. This collected information is both from information you voluntary give away to different web providers, and information being automatically collected from use. The vision for my system is to create an application that both stores information from visited web sites, and are capable of finding similar web content on other web sites. The information should be semantically marked up information from the user's web surfing, and the system should further be able to produce recommendation for other web content.

Semantic web technologies make us capable of implement such a system. Using semantic technologies will not only extract the information needed, but also reduce the need of spreading any personal information. A system that is installed directly on a user's web browser, and which does not require any information in order to work, will not interfere with privacy concerns. The only information it needs is information provided in the web sites' HTML documents. When collecting information that is technologically provided by the web service providers, there is no leak of personal information to the web providers or the system provider. The recommendation could be done through parsing the user's browser history, but in order to recommend "new" web content it's necessary to go through more than the browser history. If the system only parses the user's history, it will never be recommended any new content. The system will then only provide content and web sites the user already has visited. Therefore I aim to, in addition to the history log, also crawl the web sites a user visits, so that I build up an understanding of what web sites and services the user are interested in. Almost like an online user profile.

This thesis is further developed to show possibilities with semantic web technologies, especially the semantic web markup technology RDFa. Through this thesis I will describe the development of an extension in Google Chrome that collects information through the user's

web surfing, and recommend new web content based on the collected information. There are two reasons why this is an interesting topic of research: The amount of structured and connected data on the web is increasing, and the privacy concerns regarding web surfing. This means the use of semantic technologies is a possible technology to be used in a recommender. When using semantic technology for a recommender tool, it can become highly accurate, and avoid the need of making users describe what they are interested in or creating an account on the web service.

The extension is developed in JavaScript, HTML and CSS. The actual development of the extension is described in detail in chapter “5 Implementing the extension”. It is developed as a proof-of-concept, meaning it’s not a complete and fully working extension. It only provides a proof for how such technologies could be used, and in addition how it can contribute to reduce privacy concerns on the web. The source code of the extension is published on my GitHub profile: <https://github.com/christoffervalland/Semantic>.

The development of the extension has been based on the methodology “Design Science in Information Systems Research” by Hevner, March, Park, & Ram (2004). This methodology is followed to ensure that the extension and its quality reach the highest quality possible. The methodology is described in detail in section “4.1 Design Science”. The actual programming part of the extension follows the development methodology called RUP, Rational Unified Process, see section “4.2 Development method – RUP (Rational Unified Process)” for details. This is because following a development methodology makes it easier to keep smaller tasks at hand, having the focus on the most important part, and work more purposeful.

1.1. Motivation

Starting off with this thesis, I thought the interest of making the web more personal and private was a hot topic. Thinking of all the web content created and collected on the web today. People are posting pictures to Facebook and Instagram, writing personal status messages, checking in to places with positional data, and much more. A lot of today’s web sites and web applications are storing unnecessary much information about their users. We can divide this information into two groups: Information users voluntarily inform to the web service, and information the service provider collects in the background without any user involvement. This, and privacy in general, will be discussed in more detail in section “2.1 Privacy”.

For me in personal, I won't say that I'm afraid of web providers collecting or using information about me. There's still something strange about the necessity of collecting that much information about me to provide a service. With service I mean for instance advertises, mail, recommendations and more. Some of this will be discussed further in section "2.1.2 Google's privacy policy" and "2.1.3 Users privacy concerns". To provide accurate advertises online, the provider needs to know what their users are interested in, but not all other data such as what devices are used to surf the web, what web browser was used, time of the day, or other personal information. Then this question comes up: How can web service providers know what you're interested in without knowing all this personal information? This is discussed in the upcoming section ("1.2 Research question").

"So, like our universe, the digital universe is something to behold – 1.8 trillion gigabytes in 500 quadrillion 'files' – and more than doubling every two years. That's nearly as many bits of information in the digital universe as stars in our physical universe." (Gantz & Reinsel, 2011).

With the citation above in mind, there's no doubt that there's a need for good organization- and search- methods in the digital universe. Organizing the entire digital universe is very hard or completely impossible. That's why the urge for other methods has become as central as it has. In the early phase of the web, the tools and methods were for instance Web Crawlers, which crawled around the web to collect information. "But it has long been apparent that an approach based only on the full-text indexing of the contents of Internet sites is not a complete or fully adequate solution for providing access to these resources. We need means to augment and enrich the 'self-description' of materials and encourage creators and third party agencies to engage in this task." (Efthimiadis & Carlyle, 1997)

The use of semantic technologies is becoming more and more central when discussing the web today. "This is the vision of the Semantic Web – an organized worldwide system where information flows from one place to another in a smooth but orderly way." (Allemang & Hendler, 2011, p. 11). Since more and more web creators and web service providers have started using semantic technologies, and especially semantic markup on web sites, the need for applications using this technology are becoming bigger.

RDFa seems to be one of the leading semantic web markup technologies together with Microformats, which means we need more applications and different approaches to benefit from this technology. RDFa are explained further in section “2.2.3 RDFa – Resource Description Framework in attributes”, and Microformats in section “2.2.2 Microformats”. As explained in section 2.2.3, RDFa is a new approach provided by W3C to add structured data directly in HTML attributes. Since the web creators add this information, it’s information they want to become useful. Further this is technological information that does not directly describe a person surfing the web in any way, which brings me back to another important topic of this thesis: Privacy.

1.2. Research question

The idea is to understand how new web technologies can be used to recommend web content without the need of collecting any personal information. To withhold the privacy concerns, I aimed to collect as little as possible information that is directly related to the user, but at the same time collect enough to be able to produce accurate recommendations. My focus will be on using semantic web technology, specifically RDFa, to collect information provided by the web sites a user visits. This information is detailed, technologically provided information in the HTML documents, which the web providers add to their web services for different reasons. The information will later be used in a recommender system, suggesting “new” web content for the user based on what gets collected from the user’s web surfing. The research question I ended up with was therefore:

Can semantic web technologies be used to reduce the privacy concerns when recommending web content?

Chapter 2

2. Theory

This section will describe theory covering my field of research, including topics as the semantic web in general, RDF and RDFa, some information about privacy and more. This will help to understand the research question, and also create an understanding of what I write about in the upcoming chapters.

2.1. Privacy

One of the main drivers and research areas of this thesis is the privacy issue. Privacy is becoming more and more central regarding the web and web surfing. When creating new accounts around on different web systems, you always need to accept some terms. I'll describe examples of such terms in greater detail later on. What most users don't know is that when they're logged in to a web system, it often stores information about you. Either personal information or just technical information, such as time and date, what URLs you visit and similar. Google is one of the leading companies when coming to information gathered from the web.

There's no easy definition of what privacy is. A very general definition of privacy is the "right to be let alone", a form of freedom from intrusion. This definition comes from a law review from Brandeis and Warren (1890), with the title "The Right to Privacy". When talking about privacy today, we often divide between intrusion or physical privacy, and informational privacy. This informational privacy is what will be discussed in this thesis.

2.1.1. Informational privacy

Informational privacy is the concern about privacy around computers and the web. In the early 90s, Moor (1990) raised a concern around personal information in computer systems. "Furthermore, because personal information about us is stored in computer databases, most of us have no control over how that stored information is used" (Moor, 1990, p. 75). This is also the situation today, even though web services are becoming better and better to inform users about the information stored. When including computers in the discussion around privacy, it raises several new issues and concerns as well.

First is the information part. The issue here is that instead of anything physical, the focus is on the information. As Moor highlighted: Most of us don't have any control over how this information is used. Second when adding the word personal in front of information, it often seems even more frightening. Many services are storing personal information such as name, e-mail address, phone number, birthdate and more. When not having any control of this information one could be suspicious around what the information is used for.

2.1.2. Google's privacy policy

Since Google is one of biggest companies on the web, I decided to use them as an example. Once you register a Google account, you also approve that Google can start collecting information about you. Google has two different categories of information to store: Information you as a user provide to the service, and information they get from your use while using their services.

The former one is often seen as the least frightening. This can be information such as name, email, and other information you need to provide to create an account. Since this is information users voluntarily give away, they know what and how much information they give up. Most users don't think this is frightening the same way as information they don't even know is collected.

The latter one is often unknown to most people, and might be frightening to some people. This is information gathered from your use. "We collect information about the services that you use and how you use them, like when you watch a video on YouTube, visit a website that uses our advertising services, or you view and interact with our ads and content." (Google, 2015c). Details around this information gathering is found on "Google - Privacy & Terms" (2015a) page, and I'll describe examples of some of them now:

- Device information; including hardware model, operating systems and even phone number.
- Location information; where you are when using Google's services. This is collected in several ways such as IP-address, GPS and cell towers and more.
- Log information; that is details of how users use their service, such as search queries. Phone log information like phone number, time and date of calls, duration of calls and more. Device event information such as crashes, activity, hardware, browser and more.

2.1.3. Users privacy concerns

There are several statistics and analysis around users privacy concerns. Already in 1999, we find analysis around Internet users privacy concerns. A technical report by AT&T Labs-Research from April 14, 1999, reports several interesting findings. They found for instance that Internet users dislike automatic data transfer. “When asked about several possible browser features that would make it easier to provide information to Web sites, 86% of respondents reported no interest in features that would automatically transfer their data to Web sites without any user intervention.” (Cranor, Reagle, & Ackerman, 1999).

The report also describes how Internet users dislike unsolicited communications. They reported that 61% of the respondents who said they would be willing to provide their name and postal information to receive free pamphlets and coupons said they would be less likely to provide the information if it would be shared with other companies and used to send them additional marketing materials.

This was reported already in 1999, and the automatically collected information hasn't been reduced in modern times. As mentioned in section “2.1.2 Google's privacy policy”, different web services are able to collect information about both software and hardware. This information is then used to “personify” the web for each user. An example of this is advertisement on the web.

On Google's “Privacy & Terms” page, you'll find information about their advertising-tools where they explain what information they collect about users. “For example, if you frequently visit websites and blogs about gardening, you may see ads related to gardening as you browse the web. And if you watch videos about baking on YouTube, you may see more ads which relate to baking.” (Google, 2015d). In addition they write on the same webpage that they are able to automatically scan content of their own services, such as Gmail. This means they are able to know if you bought a shirt and received the receipt on your email address. They can further use this information to show you ads about similar shirts.

In more recent times, research shows that people aren't necessarily that concerned about informational privacy. In 2012-13 Tessem and Nyre (2013) studied people's willingness to share personal information. In their study (called “The Influence of Social Media Use on

Willingness to Share Location Information”) they looked into how people are sharing personal information through mobile phones, the high connectivity and social media. In their report they discuss and highlight the issue around informational privacy. Their study was specifically about location-based applications and the willingness to share location, but this is still interesting regarding informational privacy.

According to Tessem and Nyre (2013) informational privacy is about protecting personal information or controlling its propagation and use. Users need to balance the value gained from sharing (with recipients), thinking that recipients can be everything from family to distant organizations. They found for instance that users who are frequently using social medias are more willingly to share location and other personal information than others. “The analysis shows that frequent social media users are more inclined to share location and other personal information than others.” (Tessem & Nyre, 2013). One of their main findings were that the more experienced a person was with social media in general, was an important cause for the increased confidence.

2.2. The Semantic Web

As you already might have guessed: It has something to do with the World Wide Web (WWW) and semantics. Semantic means “meaning”, and as (Hebeler, Fisher, Blace, & Perez-Lopez, 2009) describes: “Meaning enables a more effective use of the underlying data. Meaning is often absent from most information sources, requiring users or complex programming instructions to supply it.” Further they describe the semantic web as a “...web of data described and linked in ways to establish context or semantics that adhere to defined grammar and language constructs” (Hebeler et al., 2009).

When data is connected it is easier to understand it’s actual meaning. For instance ambiguous words like spot, live, skin and so on. The word “spot” has no actual meaning without its context. A person reading the word doesn’t know if it’s about LED spots or a spot on someone’s shirt. This is what the semantic web adds to the World Wide Web.

2.2.1. RDF – Resource Description Framework

Resource Description Framework (RDF) is a framework based on the foundation of the web: the connection between two “things”. “RDF relies heavily on the infrastructure of the Web, using many of its familiar and proven features, while extending them to provide a foundation

for a distributed network of data.” (Allemang & Hendler, 2011, p. 27). RDF extends the standard linking between two “items”, adding a third feature to describe the relationship between them. RDF uses URIs (Uniform Resource Identifier) to describe the two ends of the link. These two URIs and a third one to describe their relationship creates a “triple”. “RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a “triple”).” (Herman, 2009). The three properties of a triple are called subject, predicate and object. “The subject of a statement is the thing that statement describes, and the predicate describes a relationship between the subject and the object.” (Hebeler et al., 2009, p. 68).

Without going too much into depth the object of a triple could be either a literal or a resource. A literal is a constant value that, according to Hebeler et al. (2009, p. 69), represents concrete data values like numbers or strings and cannot be the subjects of statements, only the objects. In contrast, resources could be subjects in a triple, meaning they could represent “... anything that that can be named.” (Hebeler et al., 2009, p. 69).

It is easier to understand what a triple is with the use of examples. To describe what music that exists in my music playlist with RDF, we could say it like this: “Christoffer listens to Led Zeppelin”. Other examples could be the relationship between me and my friends at the University: “Christoffer knows Lars Petter”, “Christoffer knows Stian” and so on.

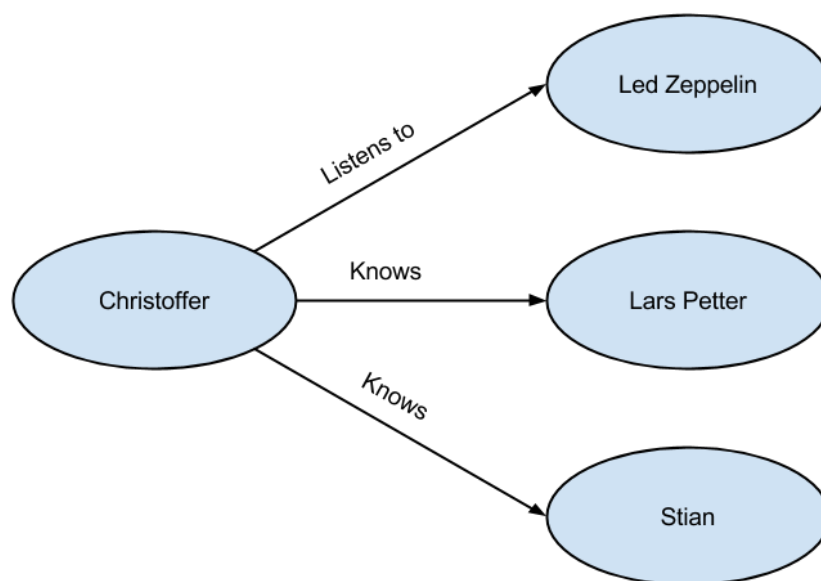


Figure 2.1: Example of RDF triples shown in a graph

2.2.2. Microformats

Microformats is a method that is based on the idea of adding structured information in web pages. Microformats rely on a standard vocabulary, making web providers add very specific data to the web sites. A vocabulary is according to W3.org (W3.org, 2015) a set of terms that can be used in a particular application, to describe possible relationships, and to define possible constraints on using those terms. According to Allemang & Hendler (2011, p. 53) the first Microformats were used for business cards and events. Further Allemang & Hendler (2011, p. 53) also explains that the vocabulary for business cards included names, position, company, and phone number. While for events it included location, start time, and end time. This is both a limitation and an advantage of the Microformats. The advantage is that there is a controlled environment, meaning everyone who uses this technique is using it the same way with the same vocabularies. The limitation is this exact controlled environment: The need for a specific vocabulary for different topics, and the need of a specific parser to process the vocabularies.

2.2.3. RDFa – Resource Description Framework in attributes

In response to the above-mentioned limitations to Microformats, W3C proposed Resource Description Framework in attributes (RDFa) as the way to add structured data (in form of RDF) to web sites. RDFa is a single syntax for marking up HTML pages with RDF data (Allemang & Hendler, 2011, p. 53). "...the ability to add structured data to HTML pages directly. RDFa (Resource Description Framework in Attributes) is a technique that allows just that: it provides a set of markup attributes to augment the visual information on the Web with machine-readable hints." (Herman, Adida, Sporny, & Birbeck, 2015). This means RDFa opens for adding attributes in HTML or XHTML documents that adds structured data and data connections in HTML and XHTML documents. RDFa works as an extension of the existing HTML and XHTML. RDFa uses the attribute tags in HTML to provide such structured data that can be parsed into RDF according to Allemang & Hendler (2011, p. 53).

RDFa also tends to reduce the gap between machine and human. When you open a website in your browser, the only thing the machine sees is the HTML attributes telling the machine whether it's a headline, bold or italics font, if it's a link to another page and so on. What humans see is much more detailed. We understand that the headline is the title of a webpage,

what the context of the text on the site is and much more. W3.org have created a nice figure, showing this exact problem (see Figure 2.2). The left side of the figure shows what the browser see, and the right side what humans see.

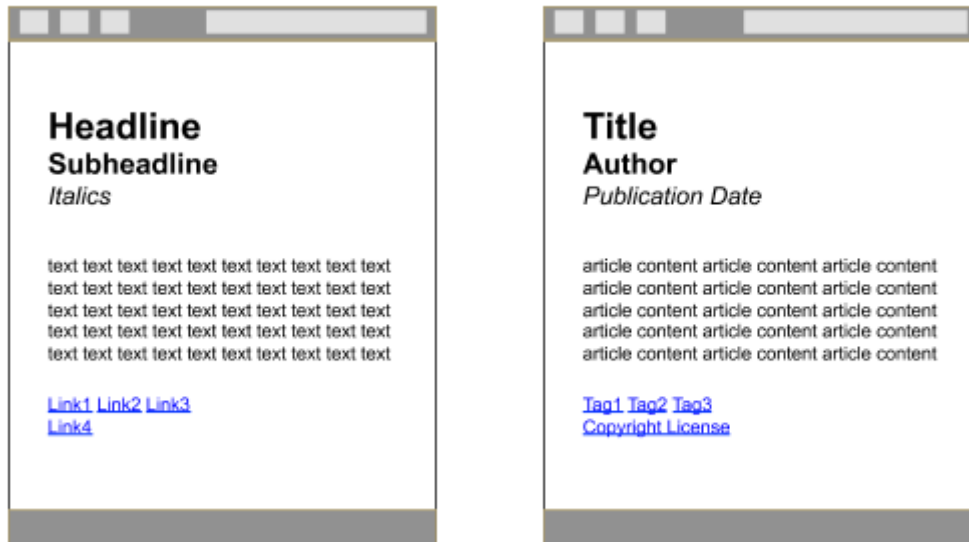


Figure 2.2: Difference between browsers and humans, collected from: <http://www.w3.org/TR/xhtml-rdfa-primer/>.

2.2.4. Usage of markup on the web

In December 2014, Web Data Commons did a crawl of over two billion URLs. They looked for how many of these who included structured data in some form. Their results showed that approximately 620 million HTML pages contained structured data (Bizer, Meusel, & Primpeli, 2014). From all these URLs the most used methods to create structured data, were Microdata and RDFa (see Figure 2.3). From the statistics provided by webdatacommons.org we find that RDFa existed on 257,251,367 URLs crawled in December 2014 (Bizer et al., 2014).

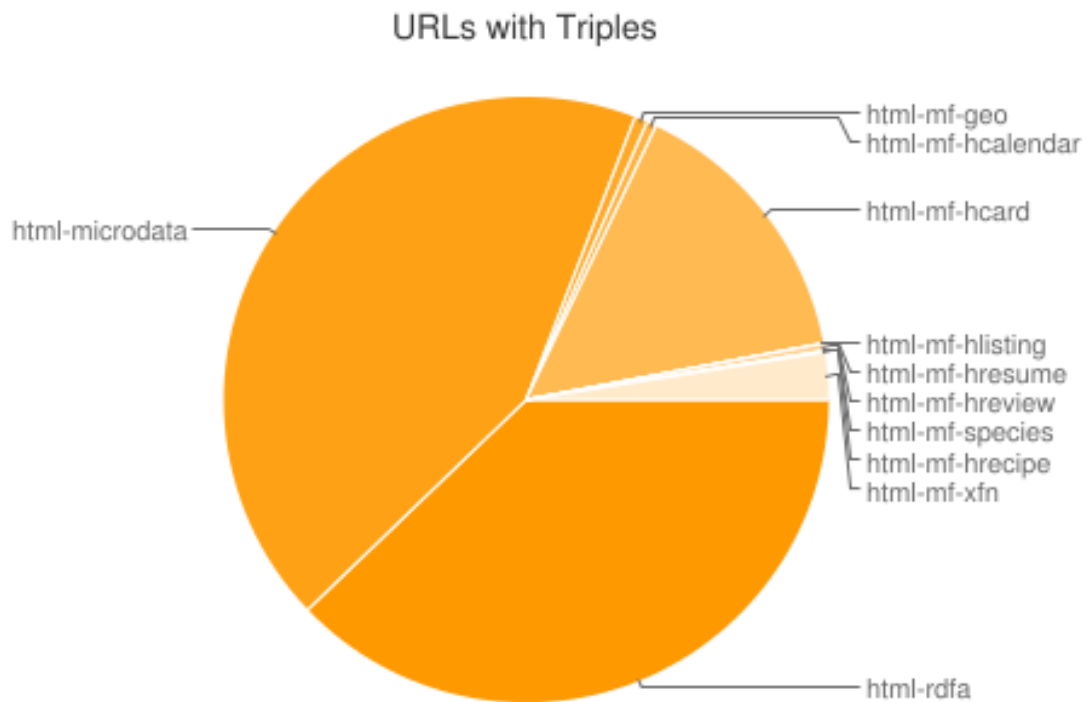


Figure 2.3: Graph showing the spread of markup-methods, from Webdatacommons.org (Bizer et al., 2014)

2.3. Collecting information, common techniques

To go in depth on many of the techniques used on the web will be too time consuming and also be a bit out of my research topic. It is still interesting and necessary to explain some of the techniques used to collect information about web users. Therefore the upcoming sections will explain cookies (web cookies) and click tracking, two different techniques for collecting information about web users.

2.3.1. User-provided information

One of the easiest ways to collect information from users of different web services are to let the users add the details themselves. This is a well-known situation for online shopping, mail systems, social media and much more. Often on such services, you'll not only have to provide a name, but also your address, e-mail address and phone number. This means all your personal contact information. Regarding online shopping users does not only need to provide contact information, but it's also very common to provide payment information. According to Statistics Norway throughout the second half of 2013 up to September 2014, as much as 77% of the Norwegian population has performed some form of online shopping (SSB, 2014).

These web services or online shops need this information the first time you order something, and some also need it for the upcoming visits. Very often do web services provide the functionality of creating an account for their service, this way users don't need to provide all this information the next time they are using the service. Below you'll find an example of such registration. The example is collected from www.komplett.no. As you see in Figure 2.4 you have to provide name, last name, address, postal code, postal place, phone number and email address. Some may wonder why you need to provide phone number or email address instead of a username.

The image shows a registration form titled "Registrering av privatkunde" with a close button (x) in the top right corner. The form contains several input fields, each with a small icon on the left: "Fornavn" (first name) with a person icon, "Etternavn" (last name) with a group of people icon, "Adresse" (address) with a location pin icon, "Postnummer" (postal code) with an envelope icon, "Poststed" (postal place) with a globe icon, "Mobilnummer" (mobile number) with a mobile phone icon, "E-postadresse" (email address) with an @ symbol icon, and "Ønsket passord" (desired password) with a key icon. Below the fields are two checkboxes: "Jeg ønsker å motta ukens tilbud, trender og anbefalinger på e-post" and "Jeg aksepterer [salgsbetingelser](#)". At the bottom left is a blue link "Tilbake" and at the bottom right is a dark button labeled "REGISTRER".

Figure 2.4: Komplett.no's online registrering sheet

2.3.2. Cookies

Maybe the most well known technology for everyday users of the web is the cookies. Cookies are a way to let applications on the web transfer data between browser and server. They are small pieces of information that are sent between browser and server. A cookie is often used to identify users; meaning cookies are capable of storing usernames and other HTML filled forms. “Cookies are data, stored in small text files, on your computer. When a web server has sent a web page to a browser, the connection is shut down, ... Cookies were invented to solve the problem “how to remember information about the user: ...” (W3Schools, 2015a).

Advertisers use cookies to build a “profile” of web users. “A typical profile might say how much a person is interested in sports or in consumer electronics, or how much he follows current events and the news.” (Garfinkel & Spafford, 2002, p. 219). This is of course information intended to be anonymous. Garfinkel & Spafford (2002, p. 219) further explains that when such anonymous information is combined with either IP-address or information provided at the web service, it becomes possible to unmask the anonymous data.

Using cookies can both improve and weaken privacy. It improves privacy because it helps reduce the amount of personal information needed by different web services. Web services no longer need to store information in a central location (server), since they’re now able to store information in the cookie itself. As highlighted by Garfinkel & Spafford (2002, p. 220) one of the most important benefits of storing information in the cookie instead of a server, is that there is no database of personal information that needs to be protected.

On the other hand cookies can weaken privacy. An example of this is provided by Garfinkel & Spafford (2002, p. 220): “When cookies are used to tie together a whole set of seemingly unconnected facts and pieces of information from different web sites to create an electronic fingerprint of a person’s online activities.” Further they explain that such cookies usually contain an identifier that works as a key into a database. Cookies are able to help search engines create a “user profile” of anonymous users. In a Utah Law Review, they explain that cookies enable search engines to recognize a user as a recurring visitor and amass his or her search history (Tene, 2008, p. 1447).

2.3.3. Click tracking

As the name of this technique indicates, this is about tracking a user's clicks around on the web. This technique is often used in advertisement on the web, such as Facebook Marketing or search results from search engines. Alberdeston, Dondyk, & Zou (2014, p. 570) write that when users are using search engines, the user's click action are first tracked by returning back to the search engine before redirecting to the corresponding target website. This is a common method, and is used by all big search engines and other web services. Alberdeston et al. (2014, p. 570) highlights two reasons for search engines using such techniques: Improving advertisement relevancy and maximizing revenue.

Even though users are not logged in to a user account related to a search engine (as for instance Google Mail), the search engine still collects a wide range of information. According to Alberdeston et al. (2014, p. 571) this information includes IP address, query term, and cookie based ID. What this actually means is that whether you are a registered user of a web service or not, they are capable of collecting information about you and your surf habits.

Keeping this in mind, we see that there are several privacy issues with this method. Even though these are anonymous data, search engines and other web services can create a user profile without any actual personal information about the user. The way you search and the search habits in general, may help identify you as a person. Back in 2006, AOL released the search history of more than 650,000 users' search history. "The 21 million search queries also have exposed an innumerable number of life stories ranging from the mundane to the illicit and bizarre." (McCullagh, 2006). Not very complex methods were needed to create an understanding of the persons behind the different searches. The search histories were connected to an anonymous identifier, such as six digit numbers. Even though this is anonymous, the search terms are connected to you no matter how anonymous it is done.

In the AOL released search histories, they found that a user had searched for terms making it easy to create an understanding of whom this person is. "... AOL user 710794 is an overweight golfer, owner of a 1986 Porsche 944 and 1998 Cadillac SLS, and a fan of the University of Tennessee Volunteers Men's Basketball team." (McCullagh, 2006). What they also could tell about user 710794, was that he was interested in the Cherokee County School District in Canton, GA., and had looked up the Suwanee Sports Academy. This was pretty

disturbing since the same user also had searched for “lolitas”, a term commonly used to describe photographs and videos of minors who are nude or engaged in sexual acts according to McCullagh (2006).

2.4. Other recommenders and content providers

There are a bunch of recommenders out there, and I will only introduce some of them. I have chosen the technology RSS, the application Flipboard, and the newest arrival from Facebook: Instant Articles. If we look on the web, we will find several others also, but I decided to limit my focus to the three that I found comparable to my system.

The big Internet firms such as Google, Facebook and others seem to be continuously competing to be the number one site to visit when surfing the web. In 2006 Google bought Youtube for \$1.65 billion according to La Monica (2006). Meaning Google is not only a search engine, but also a media center on the web. Facebook has several methods to keep the users on their application. For instance are they providing an own browser inside the Facebook app so users do not need to leave the application to get the information they want.

A problem with this could be their methods for handling privacy. All the above-mentioned methods are user-based, meaning you have to be a registered user to benefit from these technologies. In addition is the information you give away, and probably more, stored in a central database.

2.4.1. RSS

RSS stands for Rich Site Summary, and is a technology where users can add certain web sites, blogs, newspapers and more to their RSS feed. The RSS feed is continuously updating every time one of the subscribed RSS providers is posting something on the web. “RSS is technology used to monitor rapidly changing information on the web in an organized and user friendly way.” (RSS.com, 2015). RSS uses XML to tell the RSS feed when a page has been updated or changed. This means that if for instance a newspaper or a blog provide RSS on their web site, a visitor of the blog doesn’t need to visit the actual blog to know if something is new. The user will know if there’s something new, and what has been updated through the RSS feed.

2.4.2. Flipboard

Flipboard is an application both on the web and on mobile devices. Its goal is to provide personal content to their users. A user can choose topics they are interested in, and the application will provide content regarding these topics. As Flipboard writes on their own web site: "With the world's best sources organized into thousands of topics, it's a single place to follow the stories and people that matter to you." (Flipboard, 2015).

Flipboard only works for registered users meaning they store your registered information. Flipboard let you sign up with either Facebook or manually by email address. In addition to your email address, they need you to provide a full name and a password. It seems like they have tried to keep the amount of personal information needed as low as possible.

2.4.3. Facebook Instant Articles

A fairly new technology is Facebook's "Instant Articles". The aim for this system is to provide articles directly into Facebook users' feed. This way news-publishers can easily distribute articles to their readers directly in the Facebook application. As of today, Instant Articles are under testing with just a small set of publishers. What Facebook aim to do is to let the web providers post articles on Facebook, and that these become readable and interactive directly in the users' Facebook feed.

On the FAQ-page of Facebook's Instant Articles, one of the questions asked is how it will influence referral traffic, where Facebook answers: "Instant Articles display within the Facebook app, so readers no longer redirect to the publisher's website." But they further assure that they work with both publishers and comScore to "... enable Instant Articles views in Facebook's app to count as traffic to the original publisher, just as they do on the mobile web." (Facebook, 2015). ComScore is a company that collects information about peoples navigation on the web, what they click on, where they spend the most time and more. They describe themselves this way: "comScore is a leading internet technology company that measures what people do as they navigate the digital world - and turns that information into insights and actions for our clients to maximize the value of their digital investments." (ComScore, 2015).

Chapter 3

3. Technologies

3.1. Javascript

JavaScript is a programming language to be used on the web. Many, or all, modern websites are using JavaScript to make their websites more interactive. All modern browsers including Internet Explorer, Safari and Google Chrome support it.

3.2. jQuery

jQuery is meant to make it easier to develop web applications with JavaScript. JQuery makes common programming tasks much more easy and possible with fewer lines of code. According to (W3Schools, 2015b) jQuery supports HTML/DOM-manipulation, CSS-manipulation, HTML event methods, effects and animations, AJAX and utilities.

3.3. MongoDB and MongoLab

MongoDB is an open source database system. According to MongoDB, Inc. (2013) introduction site, their focus is flexibility, power, speed and ease of use. MongoDB stores data in JSON-documents. This makes it easy to use with several programming languages, including JavaScript.

MongoLab is a cloud host for MongoDB databases. When registering you get 500MBs free, and hopefully that's enough to support my system. If the storage comes up as a problem, it's easy to buy more storage without doing anything with your information stored in the database. MongoLab also supports the REST API. This means you could easily connect, get, post and edit information stored on the database with JavaScript. In addition MongoLab provides a user interface where you can easily check the documents in a collection, empty a collection, add or remove collections and more.

3.4. Chrome Extension

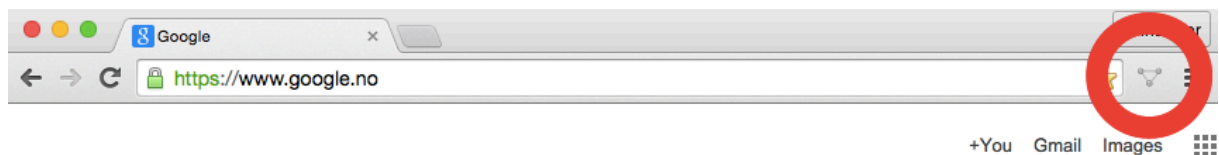
As Google themselves describe an extension: “Extensions are extra features and functionality that you can easily add to Google Chrome.” –(Google, 2015a). Extensions support integration with other websites and services than Google, and may therefore give you a personal experience of web browsing.

3.4.1. Chrome Extension Manifest

To allow a script to run in the background, it needs to be declared in the manifest of the extension. A manifest is a JSON formatted document providing important information about the extension. This information is name, version, description and more specific information about the extension. It also provides information about user interaction, permissions and more.

3.4.2. Browser- or page-action

Browser-action based extensions have the extension icon visible at all times in the browser toolbar, similar to Chrome's menu-button (see Figure 3.1). This means the extension will be visible and possible to access all the time, and is not depending on the site a user visits.



Google Search

I'm Feeling Lucky

Figure 3.1: Screenshot of icon used in browser-action marked with a ring

Page-action based extensions are displayed in the end of the address bar (see Figure 3.2). On Google's developer page about page action (Google Developer, 2015c), they write that extensions developed with page action have certain requirements and aren't applicable to all web sites. This means that the same requirements need to be fulfilled before the extension becomes visible and interactive. If the requirements aren't fulfilled or some other error occurs, the extension won't be accessible at all since the extension icon won't be displayed. Examples for this type of extension are the RSS feed symbol, which becomes visible when users visit sites with an RSS feed.



Figure 3.2: Screenshot of icon used in page-action marked with a ring

3.4.3. Background or content script

The background script allows extensions to perform tasks with longer lifetime than content scripts. This means you can use background script to handle events not directly affected by the web sites visited. As described on Google Developer: “... you can use the background page to handle events such as user clicks” (Google Developer, 2015a).

The content scripts are directly affected by the content of the web sites. These scripts can read, collect, and make changes to the DOM (Document Object Model) of web sites visited. Google describes content scripts as “... JavaScript files that run in the context of web pages” (Google Developer, 2015d).

3.5. Green Turtle

Green Turtle is an implementation of RDFa for browsers. It works as an extension for Google Chrome, and makes it possible to find RDFa triples on websites you visit. The author of the extension writes this about the extension: “By simply adding a bit of JavaScript, the DOM is extended to include the RDFa API and an RDFa 1.1 processor is available to process any ancillary documents to harvest triples” (Milowski, 2015). When these triples are discovered on a web page, you get the opportunity to open a page in the extension to view these triples in a graph.

3.6. Git and GitHub

“Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.” (Git, 2015). Git is a command line-based version control system. It allows users to create repositories that they can commit, pull and push to. The *commit* is to record changes to the repository, *pull* to fetch and integrate with branches or other repositories, and *push* to update the remote repositories and references with the local changes.

GitHub is a Git repository host service. “At the heart of GitHub is an open source version control system (VCS) called Git. Git is responsible for everything GitHub-related that happens locally on your computer.” (GitHub, 2015). In addition to Git’s properties, it adds a user interface to the Git repository. GitHub also provides some other functions, such as watching the code directly in the web browser, a graph showing your activity and more. With GitHub you are enabled to work on several hardware setups without affecting the others.

3.7. Spider

The site spider is a program or script that visits web sites, and reads and collect connected pages and other information. The classic way these spiders work is that they follow all hypertext links on a given web site, and does this for a certain number of iterations to reach a certain depth. One called “Site Spider” by Neil Fraser (2011) is written entirely with JavaScript, and allows setting the depth of crawling very easy.

Chapter 4

4. Methods

This chapter will describe the different methods used throughout the work with this thesis. The research in this thesis follows the methodology and guidelines from design science research, presented (in the article “Design Science in Information Systems Research”) by Hevner et al. (2004). The development, or programming part, of the extension follows the development method called RUP, with some modifications.

My implementation and evaluation of the methods is described in section “6.8 Evaluation of research methodology” and “6.9 Evaluating the development methodology”. In section “6.8 Evaluation of research methodology” I will describe how each of the guidelines (provided by Hevner et al. (2004)) suit my project. In section “6.9 Evaluating the development methodology” I describe how the development method worked for my project, in addition to the modifications needed order to make it work for my project.

4.1. Design Science

In order to understand the research behind an Information System, we need to understand two different paradigms that characterize the research: behavioral science and design science. Since the former is regarding more of human and organizational behavior, this thesis will deal with the latter one; design science.

Design science rooted from engineering and the science of the artificial (Hevner et al., 2004, Simon, 1996). According to (the article by) Hevner et al. (2004), design science creates and evaluates IT artifacts intended to solve identified organizational problems. Design science researchers need to understand the problem addressed by the artifact and the feasibility of their approach to its solution (Hevner et al., 2004). In the article, they provide seven guidelines to follow when working with design-science research. These guidelines are also provided in the appendix “9.1 Appendix 1: Design-Science Research Guidelines”.

1. Design as an Artifact
2. Problem Relevance
3. Design Evaluation
4. Research Contributions
5. Research Rigor
6. Design as a Search Process
7. Communication of Research

4.1.1. Design as an Artifact

This is the first guideline provided in the article, and states that “Design-science research must produce a viable artifact in the form of a construct, a model, a method or an instantiation.” (Hevner et al., 2004, p. 83). They describe artifacts constructed in design science research as rarely full-grown information systems that are used in practice. “Instead, artifacts are innovations that define the ideas, practices, technical capabilities, and products through which the analysis, design, implementation, and use of information systems can be effectively and efficiently accomplished.” (Hevner et al., 2004, Denning, 1997; Tschritzis, 1998). In the article they highlight that the construction of an artifact in a research setting is only a first step towards deployment, but that it is a necessary step. The research results can contribute to highlight new possibilities or address problems of the design of an information system.

4.1.2. Problem Relevance

The main goal with this guideline is according to Hevner et al. (2004) to acquire knowledge and understanding that enable the development and implementation of technology-based solutions of unsolved and important business problems. “Formally, a problem can be defined as the differences between a goal state and the current state of a system.” (Hevner et al., 2004, p. 85). Solving such problems will be discussed further in section “4.1.6 Design as a Search Process”. The research in design-science further needs to be relevant to a community. According to Hevner et al. (2004) the community consists of practitioners who plan, manage, design, implement, operate and evaluate the technologies that enable their development and implementation.

4.1.3. Design Evaluation

The third guideline tells that “The utility, quality, and efficacy of a design artifact must be rigorously demonstrated via well-executed evaluation methods.” (Hevner et al., 2004, p. 85). There exist several methods to evaluate an artifact. A table in the article describes five different evaluation methods: Observational, analytical, experimental, testing and descriptive. I won’t go into depth on each of these, see the appendix “9.2 Appendix 2: Design Evaluation Methods” for the complete table. Such evaluation of artifacts can involve functionality, consistency, accuracy, performance, usability, and more.

4.1.4. Research Contributions

The introductory line for guideline four is: “Effective design-science research must provide clear contributions in the areas of the design artifact, design construction knowledge (i.e., foundations), and/or design evaluation knowledge (i.e., methodologies).” Hevner et al. (2004, p. 87). Further it says that the ultimate assessment for any research is: “What are the new and interesting contributions?” When working with design-science research, you have the potential to contribute in the knowledge base. According to (Hevner et al., 2004) design-science research holds the potential for three types of research contributions. The three types are the novelty, generality and significance of the artifact.

As shown in the figure in “9.3 Appendix 3: Information systems research framework”, the development of an artifact contributes to both the environment and the knowledge base. The arrow pointing left on the bottom of the figure points on the “artifact in the environment produces significant value to the constituent IS community.” (Hevner et al., 2004, p. 87). The arrow pointing to the right side shows the contribution to the knowledge base. In addition “... use of evaluation methods (...) and new evaluation metrics provide design-science research contributions.” (Hevner et al., 2004, p. 87).

4.1.5. Research Rigor

The fifth guideline describes how design-science research requires rigorous methods: “Design-science research requires the application of rigorous methods in both the construction and evaluation of the designed artifact.” (Hevner et al., 2004, p. 87). The aim is to see how well an artifact works. This means the researcher needs to find suitable ways to evaluate the artifact, and in addition the right theory to justify the artifact.

4.1.6. Design as a Search Process

“The search for an effective artifact requires utilizing available means to reach desired ends while satisfying laws in the problem environment.” (Hevner et al., 2004, p. 83). This is the description of the sixth guideline. The search relates to the search for an optimal design, which often means an iterative process of developing. Means, in this context, relates to a set of actions and resources to construct a solution or artifact. The laws are related to the environment thinking of uncontrollable forces and everything else that is unforeseeable.

The way design science research works, is to divide problems into smaller problems or set of problems. As the smaller problems get a solution and the scope of the problems are expanded, the design artifact also becomes more realistic and valuable. “Such simplifications and decompositions may not be realistic enough to have a significant impact on practice but may represent a starting point.” (Hevner et al., 2004, p. 89).

4.1.7. Communication of Research

Guideline seven explains how “Design-science research must be presented effectively both to technology-oriented as well as management-oriented audiences.” (Hevner et al., 2004, p. 90). The technology-oriented audience need detailed enough information to be able to implement the artifact in an appropriate organizational context. It is also important that the technology-oriented audience understands the process of developing and evaluating the artifact. For the management-oriented audience, they need sufficient details to determine if the artifact will give any advantages for their organization.

4.2. Development method – RUP (Rational Unified Process)

Since I’m not working with anyone else, in a team, it’s hard to follow any development method accurately. I found several development methods that could suit my project in some form, but all required several modifications to work well. For developing my system, I ended up with following the development method called RUP – Rational Unified Process. This is actually a development methodology, which enhances team productivity in several ways (IBM, 1998, p. 1). My application and modification to RUP are discussed in section “6.9 Evaluating the development methodology”.

RUP is an iterative and incremental development method, with roots from the spiral method. In RUP we find a description of some “best practices”. These best practices are described in

details in the article “Rational Unified Process - Best Practices for Software,” (1998, p. 1–2), and consist of:

1. Develop software iteratively
2. Manage requirements
3. Use component-based architectures
4. Visually model software
5. Verify software quality
6. Control changes to software

4.2.1. Develop software iteratively

The first practice is “Develop software iteratively”. This means that with today’s complexity in software systems, it’s not possible to first define the entire problem, design the entire solution, build the software and then test the product at the end (IBM, 1998, p. 2). RUP supports developing software in iterations, which helps to attack risk through demonstrable progress, executable releases and feedback. “... iterative approach to development that addresses the highest risk items at every stage in the lifecycle, significantly reducing a project’s risk profile.” (IBM, 1998, p. 2).

4.2.2. Manage requirements

The second practice is “Manage requirements”. This is used to document required functionality and different constraints, but is also used to track and document tradeoffs and decisions made through the development (IBM, 1998, p. 2). Further, they explain that use cases and scenarios have proven to be an excellent way to capture functional requirements which again making it more likely to fulfill the end user needs.

4.2.3. Use component-based architectures

The third practice explains how RUP supports developing through components, where a component is seen as “non-trivial modules, subsystems that fulfill a clear function.” (IBM, 1998, p. 2). Some components are seeking to solve a wide range of common problems, and are built for reuse. This way developers are able to use existing components rather than building it all from scratch. A specific way to show that RUP supports components is through the iterative process, developers are able to “progressively identify components, and decide which ones to develop, which ones to reuse, and which ones to buy.” (“Best Practice: Use Component Architectures,” 2001).

4.2.4. Visually model software

This step wasn't a central part of my development, but shortly explained in (IBM, 1998, p. 2): "... write code using 'graphical building blocks'." This means the use of visual elements shall provide an understanding of how the elements of the system fit together. Further they explain that the use of UML (Unified Modeling Language) is the foundation for successful visual modeling (IBM, 1998, p. 2).

4.2.5. Verify software quality

RUP focuses on keeping quality a part of the whole process of developing. It's highlighted that quality assessment is part of all activities and involves all participants, and are not treated as a separate activity performed by a separate group (IBM, 1998, p. 2).

4.2.6. Control changes to software

The last practice provided is the "ability to manage change". "The process describes how to control, track and monitor changes to enable successful iterative development" (IBM, 1998, p. 2). This practice is mostly suited for teamwork, and therefore not directly affecting my development. The practice aims to control changes in "an environment in which change is inevitable" (IBM, 1998, p. 2).

Chapter 5

5. Implementing the extension

In the next section I'll go through how I implemented this system. This includes some of the choices I've made, how I used the technologies, implementation of the database and more. My development will be based on existing scripts and extensions that use RDFa. The reason for this is described in details in both the upcoming sections, in section "2.2.3 RDFa – Resource Description Framework in attributes" and in section "6.1 Why RDFa?". The extension and its source code is found on my GitHub profile:

<https://github.com/christoffervalland/Semantic>.

5.1. Research

It all started with a lot of research on the topic. What had been done before, what could I reuse for my system, was it possible to do this at all? In the early stages I felt it necessary to collect some information about building extensions, JavaScript applications and web browsers in general. When this was covered, I needed some information about semantic technologies on the web. This includes existing APIs, applications and so on.

As shown through section "2.2.3 RDFa – Resource Description Framework in attributes", RDFa and Microformats are the leading semantic markup technologies. Regarding the fact that Microformats are more closed and not as agile, my focus has been on the RDFa technology. I found out early that there is some semantic APIs developed in JavaScript. For my system, I ended up using one called "Green Turtle" (described in section "3.5 Green Turtle"). Since Green Turtle is an open source extension for Chrome, I could use this as a foundation for my own extension. I used this to learn how Chrome extensions actually work, and I also use this as an RDFa API in my extension.

Green Turtle is working client side meaning nothing gets stored about the users. It is just an extension that lets users get information about what RDFa triples that exist on the web site visited. This means there is no privacy concerns regarding use of Green Turtle.

5.2. Development

This section will explain how I developed and programmed the system. It will include an overview of each of the iterations of development, from research to complete system. Each of these iterations will describe my goal for the iteration and how I worked to get it done.

5.2.1. Iteration 1 – Pre-programming work

My goal during this iteration was to get an overview of what resources I needed, and create a plan for the programming. I decided to create a simplified Kanban with four fields: “To do”, “In progress”, “Review” and “Done”. This way I could continuously set minor goals for development. In addition this made me keep track of what parts that needed a review and which were completed. I wrote down the goals on yellow post-it notes, together with an estimation of how many hours I would need to get it done. These estimates rarely matched the actual hours used, but it was nice to use as a motivation. It helped me become more focused on the task at hand, in addition in times where I could risk becoming distracted, I had these post-it notes telling me that I only had that many hours to get the task done.

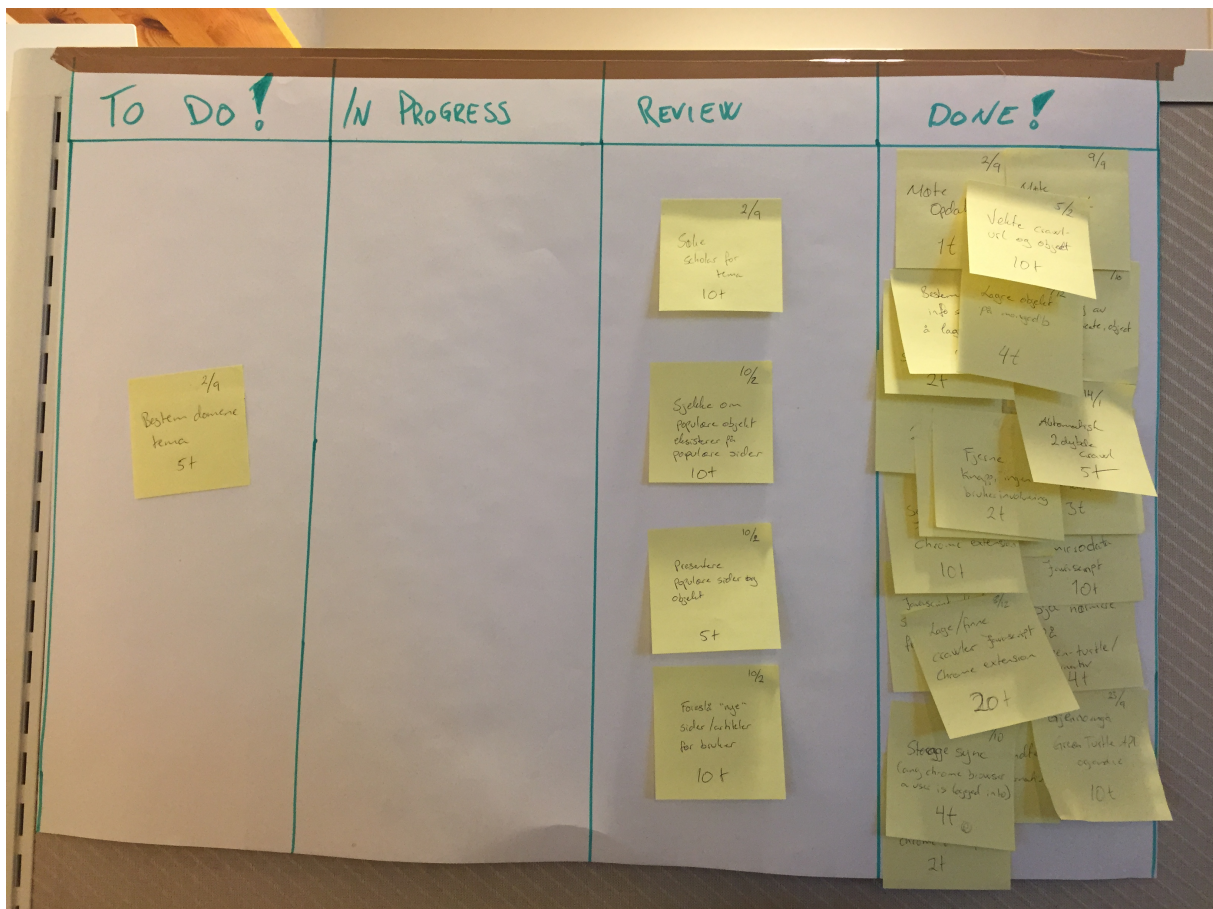


Figure 5.1: Image of my Kanban board late in the development phase

On the start of this iteration I also needed to find tools and frameworks for development in JavaScript. I wanted to find the suitable tools so that I could reduce the frustration and the time spent being unproductive. In case something happened, I would need some backup of the system. Since virtually every developer either uses or at least have heard about GitHub (described in section “3.6 Git and GitHub”), I felt that this was the system to use. I used GitHub to backup my daily work on the system. It was also great to have in case something would happen during development. If I managed to hit the wrong button, write non-functional code or anything, I could easily just go back to a previous working version of the system.

Green Turtle (described in section “3.5 Green Turtle” and section “5.2.2 Iteration 2 – Modifying Green Turtle”) is an extension meant to find and visualize RDFa triples on visited web sites. I therefore needed to modify it, remove several lines of code and make it more suitable for my system. The biggest part was finding out how and which lines of code I would be using. This way I could more easily keep track of how I could use the Green Turtle API. It involved removing several lines of unnecessary code, adding some of my own code and creating a completely new user interface for my users. This part will be described in “5.2.5 Iteration 5 – User Interface”.

```
1 {
2   ...
3   "background": {
4     "scripts": ["background.js"]
5   },
6   "browser_action" :
7     {
8       ...
9     },
10  "content_scripts": [
11    {
12      "matches": [ "<all_urls>" ],
13      "js": [ "harvest.js", "Thirdparty/SiteSpider/spider.js" ],
14      "run_at": "document_end"
15    }
16  ],
17  ...
18 }
```

Snippet 5.1: JSON of the extension's manifest file

I also needed to create the manifest file as explained in section “3.4.1 Chrome Extension Manifest”. The snippet above (“Snippet 5.1”) is rewritten from my own manifest file. I’ve chosen not to include the entire file since it’s not everything that is very interesting. The code snip shows my declaration of what script should be run in the background, that it’s a browser-action based extension and that there are two script that are working with the content of visited web sites.

As you can see in the “Snippet 5.1”, I have one background script. This background script is used to post all the visited URLs to the database (described further in section “5.2.3 Iteration 3 - Database”), and also to handle the browser action when users click the extension icon. Further I have two content scripts. These are used for several purposes including collecting the RDFa triples, crawling URLs and posting collected content to the database. I’ll explain them more detailed in the upcoming sections.

5.2.2. Iteration 2 – Modifying Green Turtle

For my own extension I used Green Turtle to harvest the triples that is used to build an understanding of the user’s interests. The Green Turtle extension extracts all the triples, so my goal for this iteration was to collect them in a way so I could easily reuse them. I also aimed to collect URLs a user visits, and the URLs connected to the visited page.

When a user opens my extension, the Green Turtle scripts are running in the background checking for RDFa and extracting triples. For this to work, I had to do quite some changes to the scripts. The original Green Turtle script didn’t show any information, triples or anything without a user interacting with it. I needed to collect information from the triples without any interaction, that means as a background function. I needed to use several of the scripts from Green Turtle, and each of the scripts I needed were quite long. The main RDFa script is originally more than 3200 lines of code, and the script to actually collect this RDFa information is more than 1700 lines. This means it was quite time consuming when I needed to remove a lot of the existing code in order to make Green Turtle suit my project better.

In section “2.2.1 RDF – Resource Description Framework” I describe the difference of a literal and a resource in an RDF-object. From the collected RDFa-triples I was interested in collecting the literals of these object, since these are actual data values. As a result, I spent

some time reading and understanding how the API processes RDFa. In the end I managed to collect only the literals, meaning the collected information was valuable data.

Next thing was to implement a method to collect the information without any user involvement. The easy part of this method was to extract the URLs someone visits. To do this I only needed to add a listener on the open tab in Google Chrome. To write the current tab's URL, you could do it as easily as this:

```
1. chrome.tabs.onUpdated.addListener(function(id, changeInfo,tab){
2.     console.log(tab.url);
3. });
```

Snippet 5.2: A way to collect the URL from the open tab

In the code above, the id is just a unique integer for each tab, the changeInfo is a list of all changes to the state of a given tab, and the tab is what I'm interested in. The "tab" keeps information and properties about the tab.

The URL a user visits isn't the only information I needed to extract without any interaction. I needed to get the objects from the triples on each site that has RDFa markup. Therefore I added code in the background script of the extension so that these methods are called automatically when a site is loaded. In addition to the visited URLs, I needed to collect information about which pages are connected to these. I needed to implement a crawler or a spider. Since this has already been done many times before, and there's nothing special about doing this, I found one already written in JavaScript by a person named Neil Fraser (see section "3.7 Spider"). Since this script is written in JavaScript it was easy to implement in my own system.

5.2.3. Iteration 3 - Database

The needed information was now available in a proper format, so my goal for this iteration was to create a functional way to store the collected information. The first step was to find the different opportunities to this problem. With some time spent on Google, I found two different approaches:

1. Storing on Google's Chrome Storage (Google Developer, 2015b), or;
2. Make my own database.

I tried to read me up on the topic and add some test information to Chrome Storage, but I soon found out that it lacked some of the functionality I wanted. In the development phase I needed to have some sort of control of, and being able to access, the information at any time. Chrome storage has quite strict limits on both size and number of items. The size limit is 102,400 bytes, and maximum number of items that can be stored are 512 (Google Developer, 2015b). With such limitations I saw the need of a more flexible and functional way of storing my collected information. Therefore before spending too much time on it, I decided to make my own database.

The solution I ended up with was creating a MongoDB hosted by MongoLab. Since MongoLab support REST API, there was no need for any installation or setup, and I could start adding collections and documents right away. Now I needed to decide what and how the information should be stored, so that I could easily extract information from the database again. The database collections I ended up with was one collection for visited URLs (“urls”-collection), one for URLs found during crawling the visited web sites (“spiderurls”-collection), and one for storing the objects (“objects”-collection). The objects are the objects from the RDFa triples found with Green Turtle.

5.2.4. Iteration 4 – Database (continues)

As already mentioned I made three different collections. In the upcoming iteration I needed to fill these collections with documents. Both the visited URLs and the objects were on the right format, and could be posted right away. Regarding spidered URLs I needed to modify the spider script with a call to the database, so that I could post all the URLs found with the spider script. The documents in the collections contained two name-fields, one for what I’m storing and one named “weight”. For the collection of visited URLs, this meant that I created one name-field called “visited” consisting of two new name-fields “url” and “weight”.

```
1. “visited”: {  
2.     “url”: http://christoffervalland.no,  
3.     “weight”: 17  
4. }
```

Snippet 5.3: JSON of a collected URL

I created the field “weight” to work as a counter. The first time I add a document to a collection the weight is set to one. When I find a document (in this case a URL) that already exists in the database, the weight is increased by one. This is also why the weight of the URL in the JSON code snip is 17.

My approach to updating the weight-field was to check if a tab’s URL already exists on the database or not. If it exists I increase the weight with one, and if it doesn’t exist I create a new document containing the tab’s URL with the weight one. To manage to do this with the crawled URLs and objects needed a lot more thinking and coding. That’s because it can be several hundreds or thousands of URLs and objects on larger web sites. In addition I can’t put them all up at once since that will result in too many request to the database.

A solution could be to put all the URLs found on a web site to an array, and put the entire array up on the database. This could bring up some other problems: For instance if a user opened the main site of an online newspaper to just check the headlines, it’s not sure that anything would be posted to the database at all. This is because the user most likely has closed the browser window before the list has been pushed to the database. Therefore I implemented a method that posts or update the collection every time a new URL is found on a web site. This way if a user just opens an online newspaper to check the headlines, at least some URLs will be posted or updated to the database.

Regarding the last collection “objects”, this had the same problem as crawled URLs. With the objects I wouldn’t risk losing too much information, so I had to be careful that objects actually got stored as soon as possible. Therefore this script is added as a background script, which means it can run for as long as possible. In the developer section of Google’s web site, they describe background scripts like this: “Makes Chrome start up early and shut down late, so that apps and extensions can have a longer life.” (Google Developer, 2015e).

5.2.5. Iteration 5 – User Interface

This iteration was all about building an interface for the users to interact with. The goal here was to create the interface in a way that the user was shown the most possible information without creating too much of a mess. To do this I used the Twitter Bootstrap framework to build the HTML page. I ended up with using Bootstrap because of its easy grid system and

its focus on responsiveness. I use the grid system to divide the different content collected from the database.

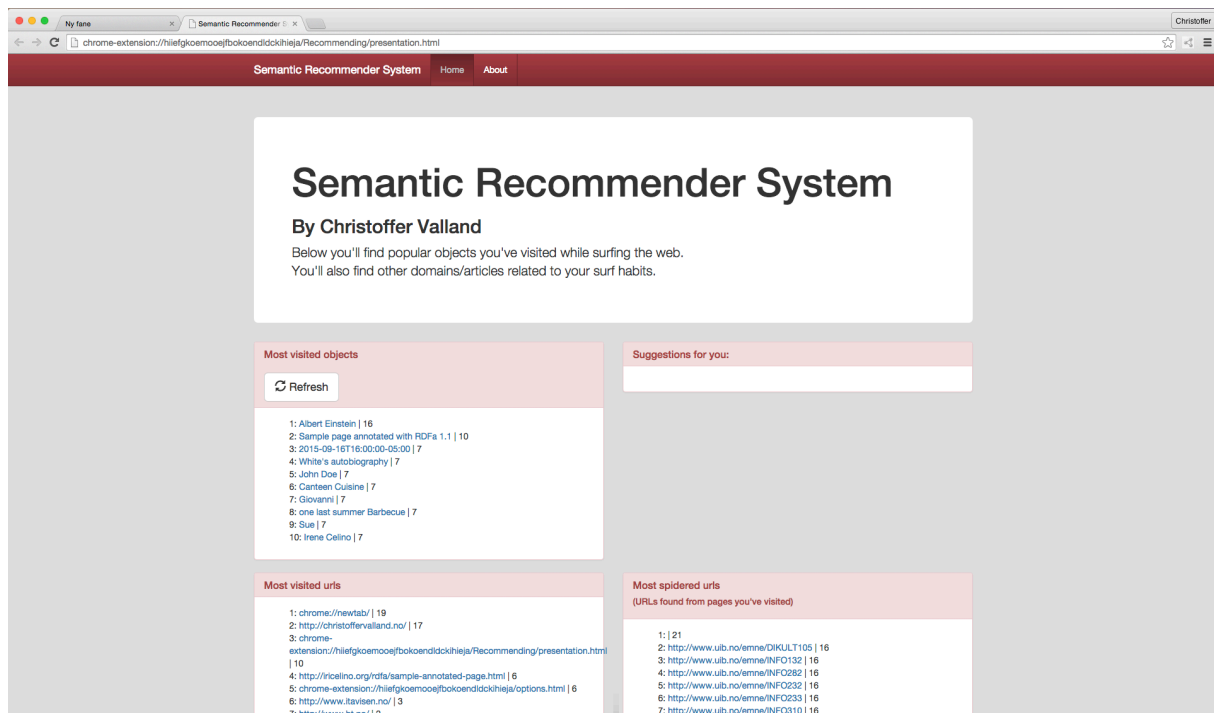


Figure 5.2: Screenshot of the User Interface

There's a JavaScript connected to this page that are doing all the queries to the database. The script is sorting the content on the database so that the ones with highest weight come first. This means it's just to iterate from highest to lowest, and loop through the amount of objects I want to present to the user. On the screenshot above, I've chosen to present ten of each; ten objects, ten visited URLs and ten crawled URLs. If I want to expand to show 15, 20 or more documents from each collection, I can just update the script with a loop going through the number of iterations wanted. I've also implemented a refresh-button, which updates the object-list with ten new objects every time the user clicks the button.

In addition I need the lists to be interactive. Since the list of visited URLs and crawled URLs consists of only URLs, this was easy. The solution is to add a HTML `<a>` href attribute to each list item. On the objects-list I needed to make them clickable, but instead of opening a URL, I needed it to search for the object on visited URLs and crawled URLs. I expanded the script with a method, which loads each page in the background and searches for the object on the web sites stored in the database. If the object exists on a site, the site gets added in an

empty grid, the one on the upper right side of the screenshot in Figure 5.2: Screenshot of the User Interface. Since the database will contain more and more URLs, searching every single site will be quite time consuming in the end. The list is, as already mentioned, sorted with the highest weight first, so setting limits to 50 or 500 isn't any different than setting the limit for how many documents that should be presented in the user interface. It's only to loop through the amount needed.

5.2.6. Overview of the complete system

The extension itself is built up by several different modules. The modules are coupled in a way that they don't affect one another when edited or modified. The modules are:

- Green Turtle API for finding RDFa triples (section "3.5 Green Turtle").
- Green Turtle module, which harvest the objects from the triples (section "5.2.2 Iteration 2 – Modifying Green Turtle").
- Crawler/spider to find URLs (Fraser, 2011).
- Database created with MongoDB, and hosted by Mongolab.com.
- User interface created with Twitter Bootstrap 3 (<http://getbootstrap.com/>).
- A module to recommend potential interesting URLs for a user.

5.3. Data flow / Information flow

Figure 5.3 below shows an overview of how data and information flows in the system. A user who uses the extension injects scripts to sites they visit. These scripts are connected to a database, which stores all the information about URLs visited, and RDFa objects found on these websites. When the user interacts with the extension, they are presented with a web site created to show the user's surf habits. This means that the web site the users are presented with calls the database for information, and presents it in tables. There are four tables; one for the often visited URLs, one for URLs found through the site spider, one for the objects found during web surfing and the last one to show the users new sites they might be interested in.

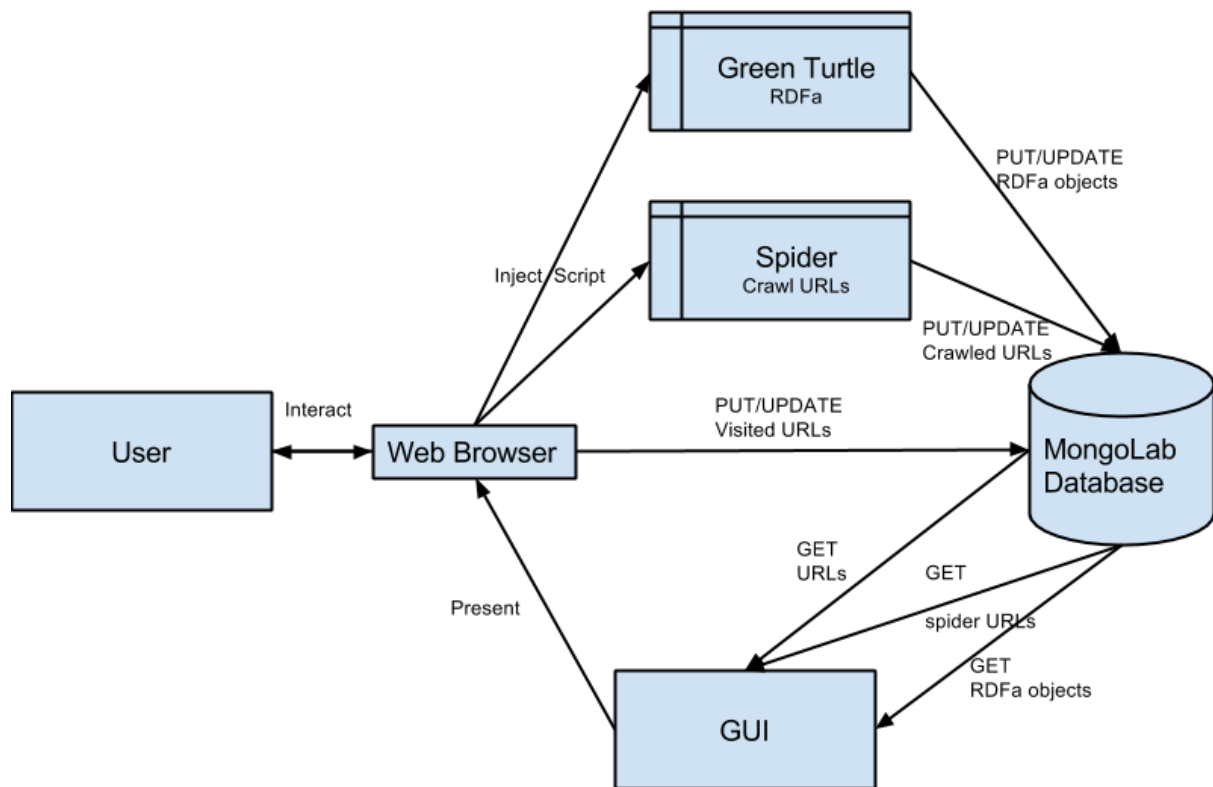


Figure 5.3: Data- and Information-flow

Chapter 6

6. Analysis and Discussion

This section will provide an analysis and a discussion of the application in relation to the research question: *Can semantic web technologies be used to reduce the privacy concerns when recommending web content?* First I will go through a use case scenario where I have the extension installed on my own web browser. Then I will discuss the results from this user test. I will discuss solved and unsolved privacy issues, future improvements and visions, in addition to an evaluation of the methodology used for this research work.

6.1. Why RDFa?

As mentioned in section “2.2.2 Microformats”, Microformats have quite some limitations. These limitations are making it a lot harder to work with the data provided, especially the fact that microformats need a parser to process the specific vocabulary. RDFa are easier to work with this way (see section “2.2.3 RDFa – Resource Description Framework in attributes”). The method is to add information that can be parsed into RDF through the HTML attribute tags. This makes it easier to collect and use the actual data provided in these tags.

In addition, several of the big web service providers adopt RDFa. According to Allemang & Hendler (2011, p. 53), web providers such as Google, BestBuy, Overstock.com and Facebook adopt RDFa. There are also other advantages of using RDFa: Regarding the data consumers, “... it is easier to harvest the RDF data from pages that were marked up with structured data extraction in mind” (Allemang & Hendler, 2011, p. 53), and from the perspective of authors of different web sites: “... it allows them to express the intended meaning of a web page inside the web page itself.” (Allemang & Hendler, 2011, p. 53).

In Figure 2.3 I provide a chart from Webdatacommons.org (Bizer et al., 2014), where one clearly sees that the most used semantic markup technologies are Microformats and RDFa. Even though Microformats is the most used according to this chart, the limitations of using this technology helped me decide to use RDFa for my project. If it was the right choice or not, I will discuss in “6.2.5 Result of use”.

6.2. The extension in use

I decided not to include any other persons to contribute in evaluating the system. This decision was made due to lack of time and the complexity of the system. The results during the test throughout development and my own test case (see section “6.2.5 Result of use” for more details) have shown that the created system is not accurate enough. In addition I thought of the problem around potential users testing my extension regarding a user’s surf habits. Most likely a potential test user would not surf the way they otherwise would. I believe that a user who knows that I am collecting information about his or hers web surfing, would keep this in mind at all times, and therefore not visit all the same sites as they otherwise would.

When starting my own test of the system I will start with an empty database and the newest update of Google Chrome (version 42.0.2311.135 on Mac) with no other extensions active. This way I will not get any results from previously completed tests, such as the tests during development and similar.

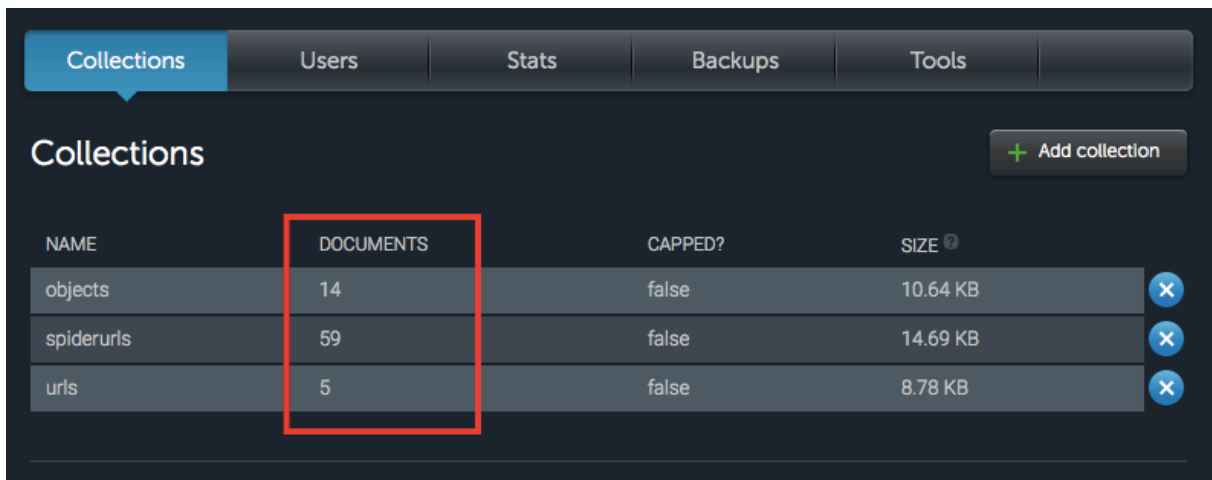
6.2.1. Limitations

The extension has some serious limitations. Probably the biggest one is how it collects and stores the RDFa-information. I tried to store both the predicate and the object of a triple, but this was later reduced to only storing the RDFa-object. This was decided as a result of new problems when recommending web content. If the system should take the predicate in consideration too, the number of recommendations would be heavily reduced. In my own test, it resulted in no recommendations at all.

In addition when storing both the predicate and the object, it was more demanding for the database. A result of this was several errors when using the extension, in addition to the more serious problem of some of the information not getting posted to the database. These problems helped me decide to only store the object of the found triples. When using only the objects, the extension works more like I first thought it should, and the recommendations are more extensive. The recommendations are not as precise as they should be, and this is discussed further in section “6.6.1 Collecting information”.

6.2.2. Clearing the database

As pointed out in section “3.3 MongoDB and MongoLab”, Mongolab provides a user interface where I can see and edit the collections and their documents. To assure that I was testing the application from an empty database, I removed all the existing documents in each collection before installing the extension.



NAME	DOCUMENTS	CAPPED?	SIZE
objects	14	false	10.64 KB
spiderurls	59	false	14.69 KB
urls	5	false	8.78 KB

Figure 6.1: Number of documents in each collection

6.2.3. Installing the extension

Since my extension is not published or released in a way that people can access and use a working version, I need to install it manually from the local space on my hard drive. Google has made it easy for developers to run and test their extensions, and has a button called “Load unpacked extension...”. This button lets us add our own extension to Chrome without any approvals, official releases or similar.

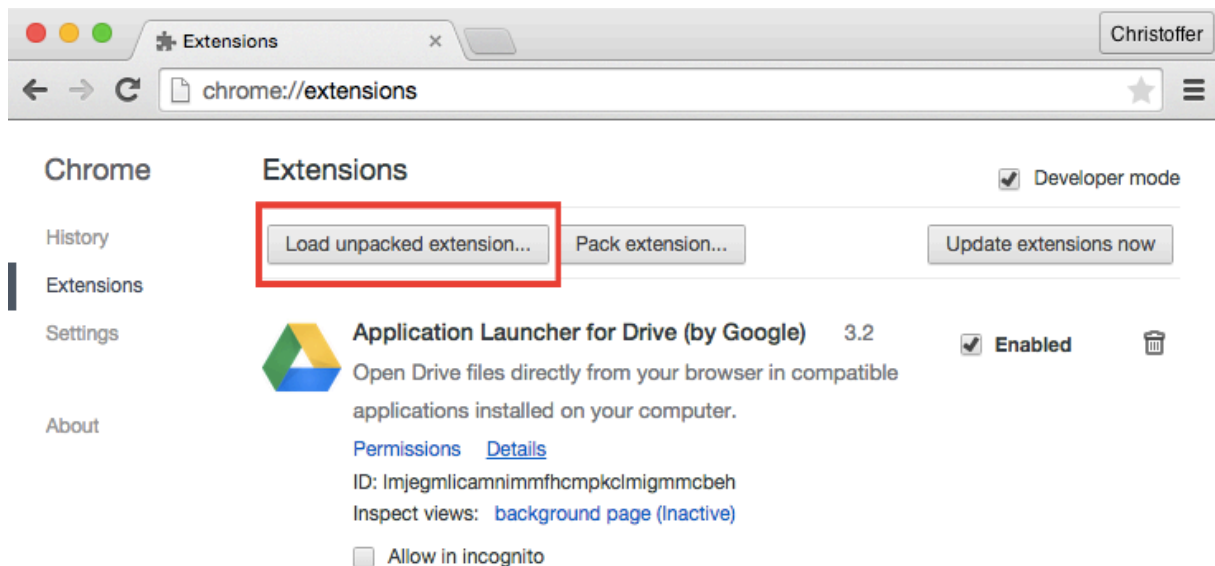


Figure 6.2: The button to load unpacked extension

6.2.4. In use

To get enough data, I decided that as much as possible of my web surfing should be performed with Google Chrome, and with the extension activated. This means that the amount of web surfing from mobile phone, tablet or other web browsers should become highly reduced. The period needed to get enough data wasn't decided in advance, so therefore I had a continuous check on the database how many URLs I had visited and how many objects were collected. This was all done through another web browser to avoid that for instance the web page of MongoLab or MongoDB would become the most visited URLs. Since all the functions of extracting and collecting the information needed are done in the background, I did not notice any difference from browsing with or without the extension activated. This was also one of my goals with developing such a system.

After the first day in use, I had spent a lot of time in school writing for this thesis. That means the amount of visited URLs wasn't especially high, only 27. I noticed that the intensity of surfing the web wasn't as high as I first thought, so the days after I did a lot more web surfing. A conclusion to the actual usage is not shocking in any way; the more it gets used, the more interesting the results became. This means that the more URLs and objects that are collected in the database, the more new, and hopefully interesting, content I manage to extract and provide with the extension.

6.2.5. Result of use

The results vary wildly. Some of the results are complete and highly accurate according to my surfing and my own preferences, but some are totally inaccurate. There may be several reasons for this, both regarding the semantic technologies used on the websites, the web service providers knowledge regarding semantic web technologies, and in addition my own extension's limitations and bugs. When the system had found 500 RDFa-objects from my web surfing, I counted how many of these I personally meant were understandable and useful. Of the collected objects, I found 157 of the 500 useful. Meaning the accuracy is approximately 31%. This number would be a lot higher if the semantic markup were more extensive than on the sites I visited. This is further discussed in section "6.6.1 Collecting information".

Considering the results that I find valuable and interesting, they are gathered from my web surfing alone, and are definitely providing me new interesting content. After visiting just the 27 web sites the first day, the recommender tool came up with new articles that I might be interested in. From these 27 web sites, I had visited the Norwegian technology web site ITavisen.no, several Norwegian newspapers and some other web sites. During this surfing I had visited mostly web sites and articles regarding technology, such as iPhone 6, some Android wearable technology, an article about Samsung's new Galaxy S6 Edge and more. From this surfing the extension added objects such as Apple Watch, Engadget and Google's Nexus TV. In my test, I chose to continue with Apple Watch. The recommender part of the extension managed to provide me articles I had not read yet, and contained information about Apple's watch. Specifically my recommender suggested me articles such as: "Apple Watch fungerer ikke på alle håndledd", "Apple forbyr fise-apper" and "Så lenge må du vente på Apple Watch i Norge".

Since there are several ways to mark sites with semantic technologies, I noticed that a big part of the collected RDFa-objects weren't interesting at all. RDFa-objects such as "websites", "article", and more were provided in the list of the most found. In addition we find RDFa-objects that probably are used as an ID for the web providers, or at least not meant to be understandable or used for an everyday-user. Figure 6.3 and Figure 6.4 shows this exact problem. This is far from ideal, and will be discussed further in section "6.6 Future work and improvements of the extension". Figure 6.3 shows the problem of using the markup on some sites.



Figure 6.3: Screenshot of the list of most common objects. Example of RDFa-objects not understandable for users.

As we see in Figure 6.3, the list of most visited objects are consisting of pretty much IDs, numbers and objects that does not give any meaning for a human. Number six in the figure is an ID used by the newspaper Verdens Gang, and is used as an ID for Facebook Insights. It is correctly used, but it isn't useful for my system. Figure 6.4 shows the results retrieved from searching for the object in Figure 6.3.

Suggestions for you:

Here's the suggestion for your search query:

152316758161786

<http://www.vg.no/forbruker/ikea-tilbakekaller-barnegrind-med-fallrisiko/a/23451022/>
<http://www.vg.no/nyheter/meninger/debatt/>
<http://www.vg.no/forbruker/helse/>
<http://www.vg.no/sport/fotball/>
<http://www.vg.no/rampelys/film/>
<http://www.vg.no/sport/hopp/>
<http://www.vg.no/sport/golf/>
<http://www.vg.no/sport/fotball/fotball-vm-2014/>
<http://www.vg.no/sport/friidrett/>

Figure 6.4: Screenshot of the result of searching for the object marked in Figure 6.3

In contrast to the RDFa-objects shown in Figure 6.3, the online video sharing web site YouTube has quite good and informative markup on their site. To give an example, I visited some of the videos by Felix Kjellberg (better known as “PewDiePie”). His videos on YouTube are marked with RDFa-objects such as “pewdiepie”, “pewdie”, “pewds”, “playthrough”, “walkthrough”, “walk through”, and “video games”. These objects are very interesting for a system as mine, and describe the user’s interest very accurately. We can tell that a user that has “pewdiepie” or “walk through” as the most found RDFa-object, are into video games.

6.3. Solved privacy issues

The great focus of this thesis is the informational privacy (see section “2.1.1 Informational privacy”). My goal was to see whether it is possible to provide personal web content or not, without storing any personal information from the users in a central database or similar. The way this was done is shown throughout chapter “5 Implementing the extension”. The answer is: Yes, it is possible, but there are several limitations with this feature. For the results of use, we see that RDFa does have some limitations. These limitations and possible solutions for them are discussed further in section “6.6 Future work and improvements of the extension”.

Some privacy issues are already covered and solved with this extension. In relation to many other web services, the extension doesn't require the user to provide any information at all; it only needs to be used. The information gathered is technological information that the web site owners want their users to use in some way.

6.3.1. Storing information

The extension is made in such a way that the information gathered could be stored almost anywhere. In my development, I chose to store them on a server. This way I could make a website, not only the site provided through the extension, gather the recommendations from the website. It also opens for a possible mobile application since all you actually need to get the information is an Internet connection. Many will be arguing that storing the information on an online web server is against my privacy focus, but; given the fact that this could be stored anywhere, it means you could just as easily store it on your hard drive. The issue then is what happens if the hard drive crashes.

Today more and more people are connecting their external hard drives to their Internet router so they can store and collect elements from it through the Internet. Given the above information; we see that if we make our own personal web server from an external hard drive, you'll actually have the same opportunities as with the server I used in my development.

6.3.2. Reducing personal information

The extension further helps proving that there's not always need for giving away personal information or the need for creating an account. This means my extension is reducing the information you otherwise need to give away to the web service. Further we see that technologies and methods such as the previously mentioned click tracking and cookies (see section "2.3.2 Cookies" and "2.3.3 Click tracking") aren't needed everywhere it otherwise would be used. When reducing the use of these methods, one also reduces information spread, or misuse of information. For instance if one web service installs cookies on your machine, this could be used to provide unwanted advertisement on the web as discussed previously.

In section "2.3.1 User-provided information", "2.3.2 Cookies" and section "2.3.3 Click tracking", I explained how the older methods still have privacy issues. Such a system that I have developed shows how new technologies can contribute in reducing the concern for privacy online. The need of cookies, click tracking or user-provided information isn't as

important as previously to create good web services and applications.

6.4. Unsolved privacy issues

Since the application as of now stores every URL a user visits in a Mongo database, one could see this as a privacy concern. With the use of such an extension, storing objects and URLs a user has visited, clearing the browser's history is not any longer enough. To get rid of all the surf history, the users will also need to clear the extension's storage.

In section "2.3.3 Click tracking" I showed an example of how data, which are seen as anonymous, can easily be used to create an understanding of the person behind the technology. This example is referring to search engines, and is not directly applied to my project and my attempt to resolve some privacy issues, but there is still some of the same problem with the use of my extension.

If the objects found through use of the extension are seen as a whole, it is possible to understand who the person is. The AOL example described a man searching for both a 1986 Porsche 944 and a 1998 Cadillac SLS, in addition to places and universities in one specific place. Seeing all his search history makes other people capable of creating an understanding of who this person actually is in real life. The same could happen with the use of my extension.

If the RDFa markup is done in a correct way, "things" such as Porsche 944, Cadillac SLS, universities and more become objects on websites. When several objects are stored in the same place, it gives an accurate description of who this person is. This is only a problem if the data gets released or in other ways gets out to the crowd. If the data are stored locally on the user's hard drive (as mentioned in section "6.3.1 Storing information"), they should be completely private and without the risk of someone collecting them.

6.5. Why it stands out from the crowd

There are several distinct differences between my developed artifact, and existing technologies and applications; both regarding privacy and the way it collects information. I will give a brief comparison to the other systems described in section "2.4 Other recommenders and content providers": RSS, Facebook Instant Articles and Flipboard.

6.5.1. Compared to RSS

The RSS technology exists of two things: The web sites providing content through RSS, an application to read or collect the feed of web sites. There are a lot of different RSS readers out there today, both for desktops but also mobile devices. The RSS technology does not provide any recommender function, but notices the user of new content on the subscribed feeds. This means that to benefit from the RSS-technology, the user is required to add all the RSS feeds manually to their RSS-reader. In comparison to my system, we see that my approach to collecting information is to collect it in the background. Comparing the way to handle privacy between my system and RSS are difficult due to the fact that each RSS reader have their own privacy policy.

What could be interesting to see is how RSS are using methods to measure “popularity” of their RSS feed. Google have developed something called “FeedBurner”. This analytic tool helps web providers keep track of their traffic. On Google’s FeedBurner site, Google has explained how it works: “...is based on an approximation of how many times your feed has been requested in a 24-hour period.” (Google, 2015b). Further Google explains how they match IP addresses to make additional inferences.

6.5.2. Compared to Facebook Instant Articles

Facebook Instant Articles are a new phenomenon, and are as of today only working on the newest Facebook application on iOS (Apple’s mobile operating system). In addition this is a system that is meant to only work in Facebook’s News feed, meaning it requires a Facebook account.

Facebook explains how they have worked with comScore and the publisher directly to enable referral traffic from Facebook to the publisher’s web site. This means it is quite more complex to understand the privacy concerns from using my system. We would need to take a closer look on Facebook’s, comScore’s and each individual publisher’s privacy policy.

6.5.3. Compared to Flipboard

Flipboard has several almost similar functions as my system. They provide recommendations out of your interests. Facebook differ from my system in one particular way: While Flipboard requires users to mark topics or words of what they find interesting, will my system follow the user on the web and decide from that information what the user is interested in.

Flipboard is open about what they collect from their users. On their page about privacy policy, they describe both the information their users voluntarily give away, and information gathered automatically from use of Flipboard. Further they describe how they use and disclose your information. They are writing several places that they won't give up any personal information except if there are reasons to comply with a law. "... our policy is to provide you with reasonable advance notice under the circumstances unless we're prohibited from doing so by law or court order (e.g., an order under 18 U.S.C. § 2705(b))." (Flipboard, 2014).

What is interesting about this is that right below the above-mentioned citation, they also write that they give up your information for third parties helping them offer and improve their service. They do not provide any description of what these services are, except from that they are "...technical tools and analytics services (including marketing partners) that help us understand how people use Flipboard so we can make it better. We require those companies to observe the limitations in this privacy policy." (Flipboard, 2014).

In addition to all the above-mentioned, they also describe how they can use an anonymous ID to share with advertising partners. Flipboard's privacy policy mentions several places that it is only meant for Flipboard, and that it "... doesn't cover the practices of other services." (Flipboard, 2014). Comparing Flipboard to my system, we see a big difference. While Flipboard collects a lot of information both automatically and manually, my system collects a lot less information.

6.6. Future work and improvements of the extension

I have a vision about how to get this extension to the next level. As a technological application in itself, I would like to see this functioning and providing accurate recommendations on other platforms. This means not being only an extension on Google Chrome, but also as a stand-alone application or at least available from mobile units.

As the dataflow diagram in "Figure 5.3: Data- and Information-flow" shows, the data and information flow today is pretty simple. It goes from users interacting with their web browser, to a database that stores visited URLs, objects and crawled URLs, the extension then calls the database to get the collected information, which are presented in a GUI for the users.

This is quite a simple architecture, and there are several ways to improve or make it more complex. As discussed in the previous sections, for instance collecting the URIs in addition to the objects will help improve the recommending and searching part. It will become more accurate without the need of very complex coding.

If we think of bigger architecture, and having a team to contribute the work, we could think of having a connection to a dataset. This will help extend the recommending part with facts from the dataset. Example of such a dataset is DBpedia. Extending the architecture of the system this way will contribute in a positive way. The connection to a dataset will for instance remove ambiguous words. This is because when the collected objects are stored with unique identifiers for objects in a dataset, there will be an understanding of what the object is referring to. An example of this is shown in the end of section “6.6.1 Collecting information”, where I discuss how “Apple” and “Apple Inc.” is not the same. The recommendations would, with such improvement, only produce recommendations of content that is related to the user’s interests. This also means that the system becomes more precise in the recommending part, the system in a whole will become more reliable. We can also provide new content in the user interface with this extension. If we were to connect the system to the DBpedia, this system could work as a “wiki” for the users. The user interface could be extended with another table including facts about the given topic.

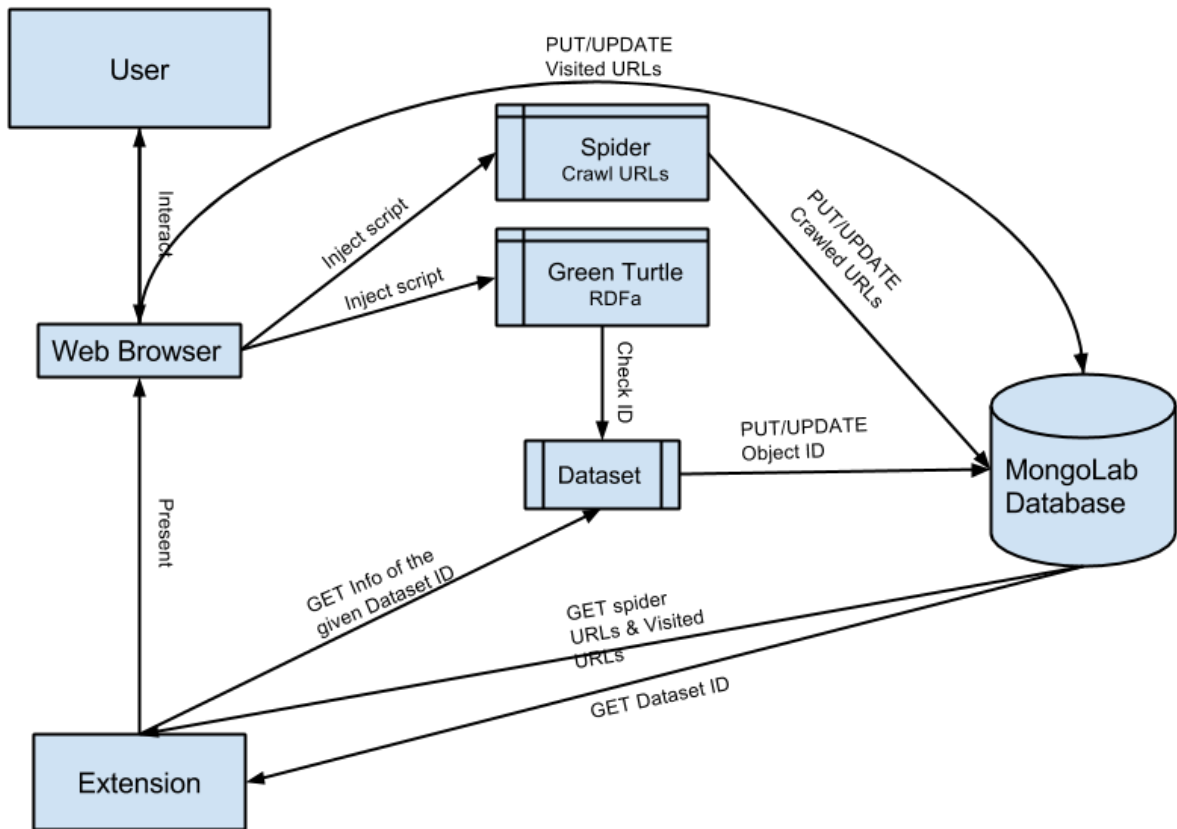


Figure 6.5: Extending the architecture of the system

In addition to above-mentioned benefits, there will also be an improvement on response time when posting to the database. Because of the fact that the extension as it is now posts both visited URLs, spidered URLs and objects to the database at the same time, it do take time for the posting to complete. If the extension first connects to the dataset, all other posting will most likely be completed before requesting a new connection to the database.

6.6.1. Collecting information

Getting the needed information wasn't as straightforward as I first hoped for when I found the RDFa API. As previously mentioned, several of the scripts I needed were very long. My time spent on just modifying the scripts therefore took a lot more time than first intended.

```

{
  "_id": {
    "$oid": "554c9815e4b09e8ca58c8e6e"
  },
  "collectedobject": {
    "object": "kvweathers=3;kvarticleid=",
    "weight": 2
  }
}

```

Snippet 6.1: JSON of RDFa object not understandable for people

The collected information includes a lot of inaccurate objects, such as the one in “Snippet 6.1” above. These objects are not wanted in the database, and do not provide any useful information for neither any user nor my system. If this is an RDFa-object that occurs on many of the visited web sites, it could become one of the most found objects. If such objects are the most found ones, the user will be confused about what it actually is and why it is one of the most popular. This is quite a serious problem since the system relies on these objects found on the visited web sites.

The problem seems to be most common in digital newspapers, and especially Norwegian ones. An example of this is the Norwegian technological newspaper “ITavisen”, which has a triple with the object “article”. See the snippet below, “Snippet 6.2”.

```

{
  "_id": {
    "$oid": "554c99f1e4b00d721e5bb91f"
  },
  "collectedobject": {
    "object": "article",
    "weight": 7
  }
}

```

Snippet 6.2: JSON of the collected "article"-object

Collecting the information does also have other concerning problems. As of now, I have chosen to only store the object of the triple. This is because the way, at least for my surf habits, the web sites I visit are using semantic markup in various ways. Surfing around on Norwegian web sites like VG.no (the newspaper Verdens Gang), BT.no (the newspaper Bergens Tidende) and ITavisen.no (Norwegian technology newspaper), we find triples like:

Subject	Predicate	Object
http://www.itavisen.no/	og:site_name	ITavisen.no
http://www.itavisen.no	og:url	http://www.itavisen.no

Table 6.1: Example of triples collected from ITavisen

The above table is extracted from <http://www.itavisen.no> on May 8. 2015. Even though it actually is correct semantic markup using the Open Graph predicate, it still doesn't help my system very much. To show why this is inaccurate information, I found an example of a marked up web site: <http://iricelino.org/rdfa/sample-annotated-page.html>. The website is developed by Irene Celino, who is an Italian semantic web researcher. On this page we find triples that are related to each other in some form. As we see in the table, Albert Einstein is connected to his name, his date of birth and his birthplace. In addition is his birthplace related the long name of Germany: "Federal Republic of Germany".

Subject	Predicate	Object
dbr:Albert_Einstein	foaf:name	Albert Einstein
dbr:Albert_Einstein	dbp:dateOfBirth	1879-03-14
dbr:Albert_Einstein	dbp:birthplace	dbr:Germany
dbr:Germany	dbp:conventionalLongName	Federal Republic of Germany

Table 6.2: Example of triples collected from Irene Celino's web site

The goal with the thesis was to collect such connected information as this, but since there are very few web sites that actually use this method, we get results as shown in the ITavisen example above. With some minor changes, we could change the entire document to look different. It could include URI and predicates with just changing the way it is stored in the

database. As previously mentioned, it is only about 30% of the collected RDFa-objects that are interesting for a potential user. If sites were marked in such way as Irene Celino has done on her example web site, the amount of interesting objects would become much higher. The test site from Irene Celino consists of 33 triples, and from these there are 14 literals that my extension collects (the entire list of collected literals from Irene Celino looks is shown in appendix “9.4 Appendix 4: List of literals from Irene Celino’s test site”). From this list I will say that “Sue”, “Sample page annotated with RDFa 1.1” and “2015-09-16T16:00:00-05:00” may not be accurate enough for a recommender tool. If we say that 11 of the 14 collected literals are valuable, we see that we have reached 78% accuracy. Note that this is not enough proof to say that the entire system would become as accurate as this, but is showing that the collected literals could potentially become more accurate with good markup.

```
{
  "_id": {
    "$oid":
    "554ca336e4b00d721e5bc831"
  },
  "collectedobject": {
    "uri": "<dbr:Albert_Einstein>",
    "predicate": "foaf:name",
    "object": "Albert Einstein",
    "weight": 2
  }
}
```

Snippet 6.3: JSON of potential improvements on the collected information

Since an object is not a single identifier, one cannot be sure that the results are accurate. If the object provided in the recommended list of objects, and it is an ambiguous word, there’s no check to see if it’s actually the same object. One of the main reasons why semantic technologies are as interesting as they are is because they link data. Semantic technologies link data so that both humans and computer know the context of an object.

Since today's markup vary as much as they do, and some sites are good and some bad, we could aim on creating new tags and attributes to make it easier for the web service providers. This could be for instance be if we were to add a `<topic></topic>` tag on the web site, we could easily know what a given web site actually is about. The problem is that the different providers could understand this in different ways. One could for instance be specific and use the tag such as `<topic>Apple Watch</topic>`, someone else could split this same tag into two: `<topic>Apple</topic>` and `<topic>Watch</topic>`.

If for instance my system were to store this as an object on the database, we would get totally wrong recommendations. Since Apple and Watch are two general and ambiguous words, we would need to provide an exact URI to these as well. If the URI for Apple was connected to a dataset, we would have either the fruit apple or the company Apple Inc. Regarding the example above, Apple Watch, the URI for Apple Inc. is not specific enough if the user was interested in Apple Watch. This means there would be a problem with both unspecific markup, and unspecific URIs. If the dataset doesn't provide a specific URI for Apple Watch at all, we could argue that the URI for Apple Inc. is specific enough for the purpose.

6.6.2. Recommending

Regarding the recommendations part of this thesis, I want it to be more accurate in form of the collected objects and the recommendations provided. As of now the collected information is as accurate as possible, but the technology and those who provide such semantic information are not yet at the level I need (as shown in the previous section). Since it is not only actual objects that are seen on as objects for the RDFa API, things such as "article", "website", and some unique IDs are also provided in the list for recommendations. If the markup was more standard and the providers were more observant of how they marked up their websites, the collected objects would also be much more accurate.

Maybe if I found some way to exclude or be more selective of which objects to actually store in the database, these inaccurate objects wouldn't be as many as they are now. I've also wondered about how one can store URIs and/or predicates. The predicates aren't needed as much as the URI, but can indeed help to improve the recommendations in form of more accurate recommendations. The URI can help create an understanding of what the object actually is in the real world. To continue using the Albert Einstein example above; if we have

stored the object “1879-03-14” in the database, we would know that the user is interested in that exact date as a result of the user’s interest in Albert Einstein.

Taking the above-mentioned in consideration, it means there is need for complete, reliable and big datasets also. Examples of such datasets are as previously mentioned DBpedia. DBpedia is a dataset created by the users of the system. This means that the system is a result of crowd sourcing. On their own web site DBpedia writes: “DBpedia is a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web.” (DBPedia, 2015). For a system such as the extension developed throughout this thesis, a dataset could contribute in several ways.

6.6.3. Searching

Regarding the searching part of the recommender, it’s now only a search for the object on the URL’s you have visited or spidered previously. The search is done through a temporarily download of the given web sites’ HTML. The web sites here are the same ordered list of web sites as used for showing the most visited URLs and spidered URLs in section “5.2.5 Iteration 5 – User Interface”. The lists of URLs in that section are limited to iterate through the ten most visited and spidered URLs. I thought this was too little in order to provide new recommendations, so the limit of iteration on the search-function is raised to 50 of each. 50 spidered URLs and 50 visited URLs, meaning it is iterating through 100 different URLs. The reason of limiting this search is because it will be very time-consuming iterating all the collected URLs. In addition if the object searched for is a common object that exists on many web sites, the list of recommendations could become very long.

The HTML downloaded from these URLs is not stored any places, but is parsed through a script when the user needs to find the object on other web sites. With other words: the script loads the HTML from the URLs, parses it, and if the object is found in the HTML, the recommender adds the URL to the list of recommendations. This means there are no checks to see if it’s actually the same object as you are interested in, as mentioned in section “6.2.1 Limitations”. When a user clicks on an object in the object list, the system checks the URLs (from both visited URLs and crawled URLs) to see if the word exists in the HTML connected to the URL. This should be done more precisely, and should provide an additional check for URI and predicate. This way one would be sure that it is actually the same objects provided in the recommendation as found through the user’s surfing habits. If this could be linked to for

instance DBPedia (<http://wiki.dbpedia.org>) or other big ontologies and datasets, the results would become a lot more accurate and not only a “proof-of-concept”-extension.

It is possible to collect and store the entire triple on the database. Due to the fact that I have no method to check if the crawled web sites have the same triples on them, I found out that the system would work better if I only stored the object. The stored objects look like the one in “Snippet 6.4”.

```
{
  "_id": {
    "$oid":
    "554ca336e4b00d721e5bc831"
  },
  "collectedobject": {
    "object": "Albert Einstein",
    "weight": 2
  }
}
```

Snippet 6.4: JSON of the collected "Albert Einstein"-object

As mentioned in section “6.2.1 Limitations”, the extension only stores the object of a triple, and not the URI and predicates. This way I save both space on the database, false objects, and the system is more time efficient in use. It would be of interest to develop a more complex database, including predicates so that we were able to control if it is talking about the same object. This is previously mentioned in section “6.6.1 Collecting information”.

6.6.4. Posting objects to the database

Since the system now posts both visited URLs, spidered URLs and objects found on each web site, it is quite time consuming. There is a big need for prioritizing what information that gets posted to the database at which time. This is just if there are no other way to store the information except a database. When posting to the database, the objects should be prioritized. This way it will be a continuous update of which objects are the most popular.

In section “5.2.4 Iteration 4 – Database (continues)” I mention that I manage to collect the objects as soon as possible due to the background script-function. The problem is that even though it gets collected effectively and almost before the user interface is done loading, it doesn’t necessarily get posted to the database as effectively. If there are several calls to the database, one connection needs to end before another will open. Now there is no priority, meaning both URLs, spidered URLs and objects are posted as soon as they are done working locally.

6.6.5. Improving User Interface and User experience

Today the prototype’s user interface is divided in four parts, which all are just list of elements. This is not inviting to use, and is not very user friendly at all. I see the potential of expanding the user interface to make it more useful and inviting for the users. The prototype system is developed with Bootstrap, as described in section “5.2.5 Iteration 5 – User Interface”. This was done both to make an interface that is understandable for a user, but mostly so that I could keep track of the actual results of each of the calls to the database. With the calls divided into separate columns and rows, I could more easily check if the actual results from the database were correct or not.

I further have a vision of including the web sites recommended in a “preview-box”. If for instance a user clicks on a recommended web site or article, the extension will provide the article directly in the extensions user interface. This will not work on smaller screens, but I see the usefulness on at least a desktop version of the system.

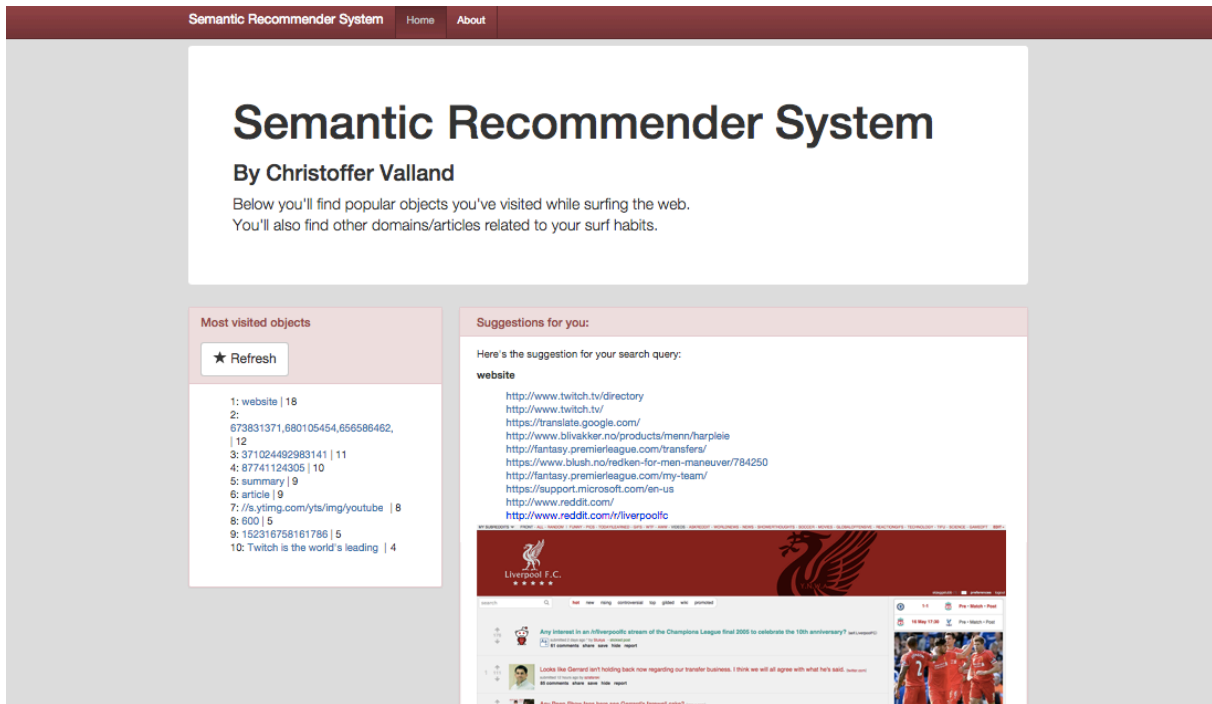


Figure 6.6: Sketch of a suggested new design

The figure above shows how I could include a preview of the wanted article. Instead of providing a wall of recommended URLs, I could include a preview of the articles below the list of URLs. Figure 6.6, shows an example of a suggested new design. The figure looks a little messy in small scale, but as mentioned this is something I thought of for a desktop version. As you see in the figure, I've reduced the list of recommended objects, and increased the size of the list of recommended URLs. This way leaves much more space to include a preview of the recommended web site.

Without any information about the topic, I could see a potential of expanding the user's interaction with the extension. I see the potential of developing a user interface with both a preview of the recommended web site, in addition to the entire web site as a read and interactive part of the extension's user interface. This means that we could get up the entire web site in the extension, without the need of leaving the extension's user interface.

6.7. Expanding the vision of the system

To think bigger for the entire system in general, I think it could potentially be developed as a part of an existing web browser, or as a completely new web browser. Web browsers today store bookmarks, search logs and more on a central server, providing you the same surfing experience on both mobile devices and on a desktop version. If we add the semantic

information gathered through such an application as developed in this thesis, we could manage to give the users new experiences with their web browser. This is also why Google, and others, opens up for their users to produce extensions for their web browser.

If I were to create a vision for the semantic web and its markup methods, I would like to see an improvement or an expansion regarding what is marked as objects on the web. If web sites could get a tag for marking their articles with topics or tags, systems such as the one I have developed would benefit directly from this tag. Some web sites already have similar functionality see Figure 6.7.



Figure 6.7: ITavisen.no's approach to marking articles with topics

The figure above shows how ITavisen.no marks their articles with different topic or tags. They have this field “Stikkord” at the bottom of each article where they mark one or several keywords in the article. The article in the figure explains how Apple Watch can function as a remote for Apple TV. A problem with doing it this way is that the web providers have to mark this manually, and there’s no check to see if it’s accurate in any way. Regarding the article in Figure 6.7, we see that the list of keywords is not precise. It only stands “apple tv” even though the article clearly is about both “apple tv” and “apple watch”.

If though these were to be semantically marked, we could easily get a connection between different keywords or objects on the web. Such linking could be used directly in new applications. As another approach to this exact topic, we could develop more advanced

checks for what objects are collected in the system I have developed. If we had advanced checks that controlled the URIs and the objects before collecting them, we would avoid such problems as discussed previously and shown in Figure 6.3.

Gmail Bilete  

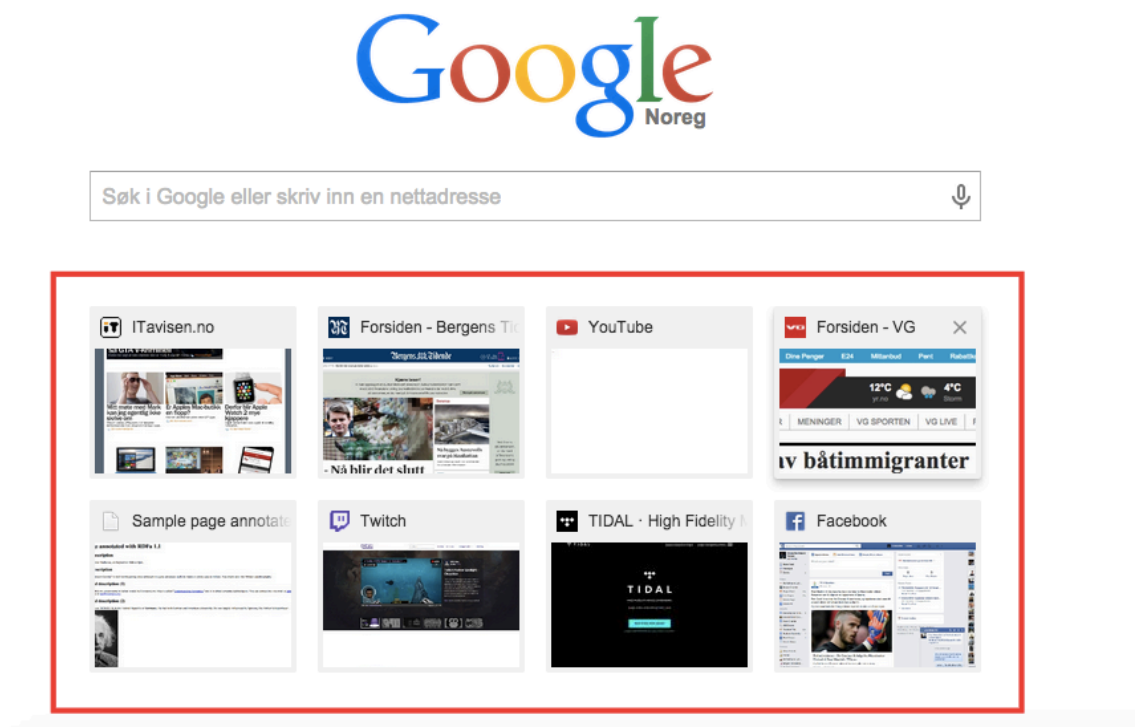


Figure 6.8: Chrome's overview of my "most visited" sites

Chrome, and others, provides a quick access to most visited web sites directly from their “New Tab”-page. Figure 6.8 is an example of this. In the figure you see my most visited web sites displayed. Apple’s Safari browser provides web sites that you have added to your bookmarks above the most visited web sites. Both of these methods are meant to help users quickly access what they spend the most time on when surfing the web. The problem with this is that it doesn’t tell anything about what the user actually is interested in.

If we could fit such an application as my system directly into the web browser’s “New Tab”-page or somewhere else, users could easily get personalized web content without the need of any particular interaction with the browser. More specific we could say that I see a vision for

such a system that I have developed to not only work as an extension to existing web browsers. It could be a part of the browser, or as an entirely new standalone browser. This would though require more from both the development and the technology RDFa.

Expanding the system in such direction wouldn't necessary interfere more with privacy than it does as an extension. This is because the information gathered from surfing still could be stored in the same way as it is: Either on a personal server, or locally on the hard drive as previously discussed.

6.8. Evaluation of research methodology

The methodology used and followed throughout this research, is described in details in chapter "4 Methods". In this section I will describe how the methodology suits my project. I will also evaluate how well and why it suits my project. The methodology is provided by the article "Design science in information systems" by (Hevner et al., 2004). Several of the provided guidelines in the article suit my project directly, and I will describe further why they suit my project.

6.8.1. Solving a problem

The first guideline in the article describes how an IT artifact should solve different problems, and how such research and development can contribute to highlight new possibilities or problems. More of this in section "4.1.1 Design as an Artifact". This thesis keeps a focus on the development of a new system that focuses on privacy on the web. The artifact developed for this thesis is a prototype of a web browser extension that produces personalized recommendation for each user. During the work the focus has been on finding new ways to keep the web personal, and at the same time avoid interfering with the privacy issues addressed with personal information on the web.

The system uses RDFa to extract objects, topics or keywords from different sites a user visits, stores them on a personal storage location, and uses this information to produce recommendation for "new" web content that may be interesting for the user. This system tries to add new ways to think of the problem of giving personalized web content, without the need for giving away any personal information to the system.

6.8.2. Changing the way to solve a problem

The second guideline focuses on how developing IT artifacts can change the way one solves a problem. See section “4.1.2 Problem Relevance“ for more details. The artifact created tries to use information already available on the web to provide personal recommendation for other web content you might be interested in. This means that a problem tried to solve or answer with this thesis is the research question: “*Can semantic web technologies be used to reduce the privacy concerns when recommending web content?*”

The technology used to be able to collect information about users interests are RDFa. Reading RDFa markup on websites a user visits makes me capable of collecting information about certain topics, people or other content the user may be interested in. The RDFa information is the foundation of recommending new information and web content for the users. The way this changes the way one solve a problem is by showing the possibility of developing an IT artifact as a proof-of-concept, meaning the new way to solve the “old” problem of recommending web content could be the use or RDFa markup.

6.8.3. Evaluating the artifact

In the article, we find the third guideline providing information about the need of well-executed evaluation methods. My system hasn’t yet been tested as a stand-alone application with actual users and test objects. The only test is a use case scenario using my own surf habits as described in section “6.2 The extension in use”. The system has also been tested for functionality during development and between iterations. If I had more time, test objects and a more precise result list, I see several methods to conduct a better evaluation.

The most suitable method for evaluating my system would include an observational, testing and descriptive evaluation. Due to lack of time and restriction on the scope, there hasn’t been done any actual evaluation of the system other than the descriptive one (in form of this thesis). If there was to become possible to perform such tests, I would like to do an observational study where I could see the system in use. This could highlight the potential users’ thoughts and meanings about the system.

From the table provided in “9.2 Appendix 2: Design Evaluation Methods”, we see that several others of the provided evaluation methods could be used for evaluating such artifact as I have developed. We find methods such as different analysis of the artifact, experiments and

testing. As already mentioned, I could conduct a case study using potential users of the system. As shown in section “6.6.1 Collecting information” I tested the extension on a web site marked with RDFa by a semantic web researcher. This was done to see if the collection of objects and recommendations became better in a controlled environment, which they did.

6.8.4. Contributing to research

As the fourth guideline in the article says: “What are the new and interesting contributions” This is related to how the development of an artifact may contribute others’ work, with showing novelty, generality and significance of the artifact. My contribution will be the developed artifact, and the documentation done in this thesis. The way I do this is to use seek to prove that already existing technologies can be used for creating a new artifact. The research shows that use of semantic technologies can be used for creating a “user model” for a personal recommendation system.

6.8.5. Research Rigor

The way I created the artifact was with the use of an iterative process (see section “4.2 Development method – RUP (Rational Unified Process)” and section “5.2 Development”). This way I keep a continuous control of functionality. This is an important part because it shows new errors occurring. At the same time it’s making me able to correct these errors in a continuous flow. Working in an iterative process further makes it more flexible and agile when it comes to changes of requirements. For more details about the iterative process and development phase, see section “5 Implementing the extension”.

6.8.6. Search process

This is related to the section above, regarding ”Research Rigor”, and as described has my project been developed in iterations. This has made me able to keep smaller tasks at hand, and also kept me focused on the one given problem. As soon as one problem was solved giving me the wanted result, I continued with another one.

6.8.7. Communicating the research

The result of the research I’ve done through this project will be communicated through this thesis. In addition the coded project is published in the developer community GitHub (see section “3.6 Git and GitHub”). That means anybody can use my project and my code to develop new artifacts, continue developing my system or in any other way use my code. Direct link to my GitHub repository is: <https://github.com/christoffervalland/Semantic>.

6.9. Evaluating the development methodology

Since I'm working alone on this project, I thought that the use of use cases or scenarios was more than I needed. For developing my system, I ended up with creating smaller tasks. I also used a Kanban board to keep track of the minor tasks, and keep track of what are the most important things on the system. Details about this are found in section "5.2.1 Iteration 1 – Pre-programming work".

On my system I've used some existing components. The main component of my system is the already existing RDFa API, which makes me able to collect RDFa elements from the web. This is explained in section "3.5 Green Turtle" and "5.2.2 Iteration 2 – Modifying Green Turtle". In addition to the RDFa API, I have used an already existing JavaScript spider (crawler). Described in section "3.7 Spider" and in the end of section "5.2.2 Iteration 2 – Modifying Green Turtle".

This is also done through my development process. For each module, I've had a continuous control of functionality and quality. Before starting off with a new component, there's been a check that the other components are working together with new modules and components.

Chapter 7

7. Summary and conclusions

The research question provided in section “1.2 Research question” was: "Can semantic web technologies be used to reduce the privacy concerns when recommending web content?". We see that the thesis and development of the extension provide answers to it. The conclusion is therefore: yes, it is possible to recommend “new” web content for users without interfering with privacy and personal information on the web, but new generations does not think of privacy on the web as that important.

The development of a recommender extension has shown that the RDFa technology can be used for recommending web content without interfering with privacy issues. With the use of RDFa, we are able to collect and provide recommendations without any information about the user. Using RDFa has shown the technology’s potential, but also it’s problems. Today’s use of RDFa can, but not always, provide rich markup of web sites. The web sites shown in examples throughout this thesis shows both good and bad markup. This thesis and the extension further show that it is possible to use such markup in other contexts than originally intended. The development of the extension shown in this thesis, can address the way to develop a recommender with privacy in focus.

To give an answer regarding the privacy topic of the research question, we see that the use of semantic web technologies on such level actually can reduce the amount of personal information on the web. This is a result of using only technologically provided information from the web service providers, rather than a personal user account, storing information about the user’s clicks, and similar. During the work with theory for this thesis, I noticed that the importance of privacy on the web is not as central as I thought when starting off with it. It is shown that frequent users of social media are willing to share personal information and location.

The provided semantic information is not yet good enough in many occasions, meaning the result of using such a system isn’t very accurate. As shown in section “6.6.1 Collecting information” the web services that are taught to use semantic markup can help create a very

accurate and rich database of objects for the user. The better the markup is, the better the performance of the extension will be.

There's no doubt that the developed extension in general has both big and minor problems with it. The idea was not to have a complete, working and complex system when I was finished with the development, but to have a working proof-of-concept of a system. Working with this thesis has shown, both through programming and writing, the actual development of such a personal recommender extension using RDFa. Throughout the work with this project, I found out that RDFa has not yet become as widespread as I first thought and hoped for. The limitations shown in section "6.6 Future work and improvements of the extension", show that different websites write triples in different ways. This means that those who spend time on understanding RDFa and semantic web technologies, can produce exact enough information for such a system that I have created. Web systems such as the Norwegian newspapers mentioned throughout this thesis have a lot of flaws in their markup, giving my system inaccurate data, which again give the users inaccurate recommendations.

The limitations with my system are not to hide away. They are of big importance for the entire result of the system. My extension actually is not contributing to the semantic field when talking about the recommender part, but rather highlights the potentials with using semantic markup. The information needed for the system to work is collected through an RDFa-API "Green Turtle", described in section "3.5 Green Turtle".

8. Sources

Alberdeston, R., Dondyk, E., & Zou, C. C. (2014). Click-Tracking Blocker : Privacy Preservation by Disabling Search Engines ' Click-Tracking, 570–575. Retrieved from <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7036868>

Allemang, D., & Hendler, J. (2011). *Semantic Web for the Working Ontologist. Semantic Web for the Working Ontologist* (2nd editio). Waltham: Morgan Kaufmann Publishers. doi:10.1016/B978-0-12-385965-5.10016-0

Best Practice: Use Component Architectures. (2001). Retrieved April 27, 2015, from http://sce.uhcl.edu/helm/rationalunifiedprocess/manuals/intro/im_bp3.htm

Bizer, C., Meusel, R., & Primpeli, A. (2014). Web Data Commons - RDFa, Microdata, and Microformat Data Sets - December 2014. Retrieved May 6, 2015, from <http://webdatacommons.org/structureddata/2014-12/stats/stats.html>

ComScore. (2015). About us. Retrieved May 27, 2015, from <http://www.comscore.com/About-comScore>

Cranor, L. F., Reagle, J., & Ackerman, M. S. (1999). Beyond Concern: Understanding Net Users' Attitudes About Online Privacy. Retrieved April 16, 2015, from <http://arxiv.org/html/cs/9904010/report.htm>

DBPedia. (2015). About. Retrieved from <http://wiki.dbpedia.org/>

Efthimiadis, E. N., & Carlyle, A. (1997). Organizing Internet Resources: Metadata And The Web, (October/November), 4–5. Retrieved from http://onlinelibrary.wiley.com/store/10.1002/bult.68/asset/68_ftp.pdf?v=1&t=i8a0uuxn&s=418d2b5f0262e5a532ab82039923f647161c36f1

Facebook. (2015). Instant Articles. Retrieved May 15, 2015, from https://s0.wp.com/wp-content/themes/vip/facebook-instantarticles/library/docs/FB_IA_FAQS.pdf

Flipboard. (2014). Privacy Policy. Retrieved May 19, 2015, from <https://about.flipboard.com/privacy/>

Flipboard. (2015). Flipboard is your personal magazine. Retrieved May 15, 2015, from <https://flipboard.com/>

Fraser, N. (2011). Google Code - Site-Spider. Retrieved March 5, 2015, from

<https://code.google.com/p/google-site-spider/>

Gantz, J., & Reinsel, D. (2011). *Extracting Value from Chaos State of the Universe : An Executive Summary*, (June), 1–12.

Garfinkel, S., & Spafford, G. (2002). *Web Security, Privacy & Commerce*. (D. Russel & C. Gorman, Eds.) (2nd ed.). Sebastopol: O'Reilly Media Inc. Retrieved from https://books.google.no/books?hl=en&lr=&id=KzabAgAAQBAJ&oi=fnd&pg=PT4&dq=privacy+issues+on+the+web&ots=XGWOXLi7As&sig=sj4ISpq3blFK--Y26kyqn6bkm98&redir_esc=y#v=onepage&q&f=false

Git. (2015). *Git --fast-version-control*. Retrieved March 24, 2015, from <http://git-scm.com>

GitHub. (2015). *Help - Set Up Git*. Retrieved March 24, 2015, from <https://help.github.com/articles/set-up-git/>

Google. (2015a). *About extensions*. Retrieved March 5, 2015, from <https://support.google.com/chrome/answer/154007>

Google. (2015b). *FeedBurner Help*. Retrieved May 19, 2015, from <https://support.google.com/feedburner/answer/78955>

Google. (2015c). *Privacy & Terms*. Retrieved April 7, 2015, from <https://www.google.com/intl/en/policies/privacy/#infocollect>

Google. (2015d). *Privacy & Terms*. Retrieved April 16, 2015, from <http://www.google.com/policies/privacy/example/ads-youll-find-most-useful.html>

Google Developer. (2015a). *Background Pages*. Retrieved March 16, 2015, from https://developer.chrome.com/extensions/background_pages

Google Developer. (2015b). *Chrome Storage*. Retrieved March 9, 2015, from <https://developer.chrome.com/extensions/storage>

Google Developer. (2015c). *chrome.pageAction*. Retrieved March 16, 2015, from <https://developer.chrome.com/extensions/pageAction>

Google Developer. (2015d). *Content Scripts*. Retrieved March 17, 2015, from https://developer.chrome.com/extensions/content_scripts

Google Developer. (2015e). *Declare Permissions*. Retrieved March 10, 2015, from https://developer.chrome.com/extensions/declare_permissions

- Hebeler, J., Fisher, M., Blace, R., & Perez-Lopez, A. (2009). *Semantic Web Programming*. *Journal of experimental psychology General* (Vol. 20). Wiley Publishing, Inc. doi:10.1016/S0022-5371(81)90569-7
- Herman, I. (2009). W3C Semantic Web Frequently Asked Questions. Retrieved April 10, 2015, from <http://www.w3.org/RDF/FAQ>
- Herman, I., Adida, B., Sporny, M., & Birbeck, M. (2015). RDFa 1.1 Primer - Third Edition. Retrieved April 15, 2015, from <http://www.w3.org/TR/xhtml-rdfa-primer/>
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 75–105. doi:10.2307/25148625
- IBM. (1998). Rational Unified Process - Best Practices for Software. *Development*, 1–21. doi:10.1.1.27.4399
- La Monica, P. R. (2006). Google to buy YouTube for \$1.65 billion. Retrieved May 20, 2015, from http://money.cnn.com/2006/10/09/technology/googleyoutube_deal/
- McCullagh, D. (2006). AOL's disturbing glimpse into users' lives. Retrieved May 5, 2015, from http://news.cnet.com/2100-1030_3-6103098.html
- Milowski, A. (2015). Google Code - Green-Turtle. Retrieved March 5, 2015, from <https://code.google.com/p/green-turtle/>
- MongoDB Inc. (2013). Introduction to MongoDB. Retrieved March 9, 2015, from <http://www.mongodb.org/about/introduction/>
- Moor, J. H. (1990). The Ethics of Privacy Protection. In *Library Trends*, vol 39 (1-2) (pp. 69–82). Retrieved from https://www.ideals.illinois.edu/bitstream/handle/2142/7714/librarytrendsv39i1-2h_opt.pdf?se
- RSS.com. (2015). What is RSS? Retrieved May 15, 2015, from <https://www.rss.com/whatisrss>
- SSB. (2014). Statistics Norway - Bruk av IKT i husholdningene, 2014, 2. kvartal. Retrieved May 3, 2015, from <https://www.ssb.no/teknologi-og-innovasjon/statistikker/ikthus/aar/2014-09-17#content>
- Tene, O. (2008). What Google Knows: Privacy and Internet Search Engines. *Utah Law Review*, 2008(4), 1433–1492. Retrieved from http://works.bepress.com/omer_tene/2/

Tessem, B., & Nyre, L. (2013). The Influence of Social Media Use on Willingness to Share Location Information (pp. 161–172). Springer Heidelberg Dordrecht.

W3.org. (2015). Vocabularies. Retrieved May 20, 2015, from <http://www.w3.org/standards/semanticweb/ontology>

W3Schools. (2015a). JavaScript Cookies. Retrieved April 20, 2015, from http://www.w3schools.com/js/js_cookies.asp

W3Schools. (2015b). jQuery Intro. Retrieved March 5, 2015, from http://www.w3schools.com/jquery/jquery_intro.asp

9. Appendix

9.1. Appendix 1: Design-Science Research Guidelines

Table rewritten from the article by (Hevner et al., 2004).

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

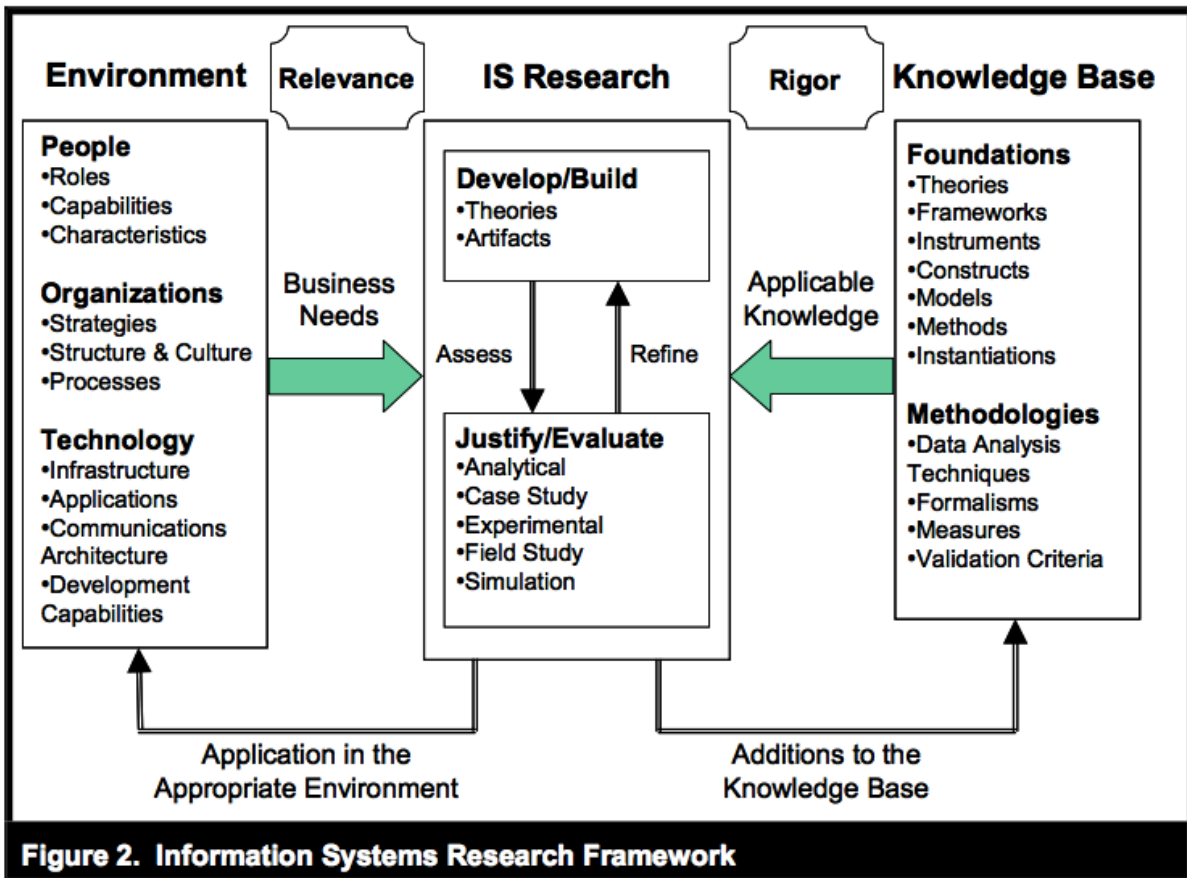
9.2. Appendix 2: Design Evaluation Methods

Table rewritten from the article by (Hevner et al., 2004).

1. Observational	Case Study: Study artifact in depth in business environment
	Field Study: Monitor use of artifact in multiple projects
2. Analytical	Static Analysis: Examine structure of artifact for static qualities (e.g., complexity)
	Architecture Analysis: Study fit of artifact into technical IS architecture
	Optimization: Demonstrate inherent optimal properties of artifact or provide optimality bounds on artifact behavior
	Dynamic Analysis: Study artifact in use for dynamic qualities (e.g., performance)
3. Experimental	Controlled Experiment: Study artifact in controlled environment for qualities (e.g., usability)
	Simulation – Execute artifact with artificial data
4. Testing	Functional (Black Box) Testing: Execute artifact interfaces to discover failures and identify defects
	Structural (White Box) Testing: Perform coverage testing of some metric (e.g., execution paths) in the artifact implementation
5. Descriptive	Informed Argument: Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artifact's utility
	Scenarios: Construct detailed scenarios around the artifact to demonstrate its utility

9.3. Appendix 3: Information systems research framework

Screenshot from the article by (Hevner et al., 2004, p.80)



9.4. Appendix 4: List of literals from Irene Celino's test site

List of literals from Irene Celino's test site
Irene Celino
Sample page annotated with RDFa 1.1
one last summer Barbecue
2015-09-16T16:00:00-05:00
White's autobiography
Giovanni
Understanding Semantics
John Doe
Sue
Albert Einstein
1879-03-14
Spinoza
Federal Republic of Germany
Canteen Cuisine