Technische Universität Berlin

An approach to semantic web annotation aggregation and interlinking via Schema.org

Master's Thesis

am Fachgebiet Agententechnologien in betrieblichen Anwendungen und der Telekommunikation (AOT)

Prof. Dr.-Ing. habil. Sahin Albayrak

Fakultät IV Elektrotechnik und Informatik

Technische Universität Berlin

submitted by

Hannes Rammer

Evaluators: Prof. Dr. Dr. h.c. Sahin Albayrak,
Prof. Dr. habil. Odej Kao

Hannes Rammer
Matrikelnummer: 335371

Erklärung der Urheberschaft

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum Unterschrift

Zusammenfassung

Als Folge des raschen Wachstums des Internets, sind wir in der Lage, über eine große Menge an Informationen im World-Wide-Web zugreifen zu können. Gleichzeitig ist es für einen Benutzer schwierig, einen Überblick über die verfügbaren Informationen zu behalten. Das semantische Web dient dazu, das Internet in ein von Maschinen verarbeitbares Internet zu transformieren. Die Meisten der verfügbaren Informationen auf einer Webseite sind insbesondere nur für den menschlichen Gebrauch und nicht für Maschineninteraktion konzipiert. Daher wurden semantische Standards wie RDF und Metadaten und ein weiter fortgeschritteneren Ansatz von Schema.org unter Verwendung von Mikrodaten entwickelt. Diese Standards ermöglichen es Web-Entwicklern, die Informationen ihrer Web-Seiten besser für die Maschinen-Interaktion zugänglich zu machen. Semantische Daten zu extrahieren erfordert noch eine zusätzliche Menge an Arbeit, da der Web-Entwickler sich über die bestehende Annotations Techniken vertraut machen muss und festzustellen, welche am Besten für seine Anliegen geeignet sind. Extrahieren von Informationen von verschiedene Web-Seiten ist schwierig, wenn sich diese nicht an Konventionen, wie von Schema.org vorgeschlagen, halten. In dieser Arbeit entwickeln wir einen semantischen Browser, der auf Web-Standards und NLP basiert. Er ermöglicht die Extraktion von semantischen Daten wie Authors und Papers auf link.springer.com oder Movies und Actors auf IMDB.com, generiert Verlinkungen zwischen ihnen und stellt diese wieder zur Verfügung ohne weitere Arbeit des Webentwicklers. Die in dieser dieser Studie überprüften Werkzeuge benötigen zu einem großen Teil zusätzlich von Web-Entwicklern zur Verfügung gestellte Informationen, und verlassen sich vor allem auf RDFA oder Mikroformat, die fehleranfällig und nicht überall verfügbar sind. Das in dieser Arbeit entwickelte Tool stützt sich auf Webstandards und NLP zur Inhaltsextraktion auf Web-Seiten mit semantisches Markup. Als solches, dient dieses Tool mit Hilfe der vereinten Datenstruktur von Schema.org zur schnellen Erkennung und Extrahierung großer Mengen von Daten aus Web-Seiten . Die wichtigste Einschränkung ist, dass wir nur einen ersten Machbarkeitsnachweis der vorgeschlagenen Lösung entwickeln. Zukünftige Arbeiten müssen daher weitere Prototypen des Tools erstellen und testen.

Abstract

As a result of the rapid growth of the Internet, we are able to access a massive amount of information via the world-wide-web. At the same time, it is more difficult for a user to keep an overview of available information. The Semantic Web serves to transform the Internet into a machine processable Internet, so machines can perform more of the tedious tasks involved. Notably, most of the available information on a web page is only designed for human usage and not for machine interaction. Therefore, Semantic Standards such as RDF a more advanced approach using microdata standards established by Schema.org have emerged. These standards enable web developers to make information on web pages more accessible for machine interaction. Services, based on techniques that take advantage of dictionaries, databases, and grammar- and pattern matching have recently been developed for gathering and interconnecting information from different web pages. Extracting semantic information currently requires an extra amount of work by web developers, who need to learn about the existing annotation techniques and determine which one is the most suitable for a specific task. Additionally extracting information across different web pages is difficult when these do not employ conventions or a unified data structure such as provided by Schema.org. In this master thesis, we develop a semantic aggregation browser that suggests to effectively supports content extraction, interconnection, and queering of semantic data from Webpages, such as Authors and Papers on link.springer.com, movies actors and ratings www.imdb.com, a product and a price as on ebay or even questions and answers as supported by stackoverflow, without the need for extra efforts in web development. Annotation tools reviewed in this study to a large extent draw on additional information provided by web developers, and mostly rely on RDFA or microformat which are error prone and not widely available. The tool developed in this thesis therefore draws on web standards and NLP to support content extraction on web page. As such, due to new semantic standards and conventions, this tool serves to extract and interlink large amounts of data from webpages by making use of Schema.org's unified data structure. This MSc project provides a tool that addresses these deficiencies in web page data

extraction.

The main limitation is that we only develop a first proof of concept of the proposed solution. Future work will therefore have to further prototype and test the tool designed in this thesis.

Acknowledgments

I would especially like to thank my supervisor Prof. Dr. Dr. h.c. Sahin Albayrak for his constant support, valuable insights and guidance and Dipl.-Inform. Maximilian Kern for the encouraging advices and inspiring ideas.

I would also like to acknowledge the following people for their help and support: Dipl.-Inform. Johannes Fähndrich, my dear mom and step dad, Gabriele Rammer-Romme and Prof. Dr. A.G.L. (Sjoerd) Romme, my aunt Prof. Dr. Viktoria Enzenhofer as well as my uncle Landesschulratspräsident OÖ Fritz Enzenhofer.

Contents

Chapter 1

Introduction

As a result of the rapid growth of the Internet we are able to access a massive amount of information via the world-wide-web (WWW). At the same time, it is more difficult for a user to keep an overview of available information. The Semantic Web has been developed to transform the Internet into a machine processable Internet, so machines can perform more of the tedious tasks involved in finding, combining, and acting upon information on the Internet. In this respect, the Semantic Web extends the current web [1].

In recent years, services have been developed that gather and interconnect information from different web pages via manual, semi-automatic and automatic Information Extraction (IE) techniques, like Wrapper Induction (WI) that utilizes Natural Language Processing (NLP), Ontology Matching (OM) and Content Annotation (CA). These techniques take advantage of dictionaries, databases, and grammar- and pattern matching. In the case of web pages, they also make use of the web page's structure as well as semantic standards like Resource Description Framework (RDF) [2] and microformat provided by web developers.

Notably, most of the available information on a web page is only designed for human usage and not for machine interaction. Therefore, Semantic Standards such as RDF, SPARQL for RDF [3] RDFa [4], OWL [5], micro and meta data (introduced by the W3C [6]) and a more advanced approach using microdata standards established by Schema.org (a cooperation by Google, Microsoft, Yahoo and Yandex) have emerged. These standards enable web developers to make information on web pages more accessible for machine interaction.

1.1 Motivation

There are several reasons for why the Semantic Web has not yet become a reality. As stated by Shadbolt, Berners-Lee and Hall.

"...the Semantic Web is a Web of actionable information–information derived from data through a semantic theory for interpreting the symbols. This simple idea, however, remains largely unrealized. Shop bots and auction bots abound on the Web, but these are essentially handcrafted for particular tasks. They have little ability to interact with heterogeneous data and information types." [7]

Shadbolt et al. also argue that agents can only flourish when standards are well established.

The idea of the Semantic Web has been around for some years now and many approaches have emerged, but there is hardly any use of semantic annotation aggregation on web pages that support Schema.org. Semantic data extraction currently requires an extra amount of work by web developers, who need to learn about the existing techniques and determine which one is the most suitable for a specific task. Additionally extracting information across different web pages is difficult when these do not employ conventions or a unified data structure such as provided by Schema.org. When having access to a unified data structure, web developers can trust that semantic information will be found accoss all web pages using an unified wording. We therefore want to observe current state of the art tools and develop our own approach towards how semantic data can be used when building on conventions and standards.

1.2 Goal

In this master thesis, we develop a tool that draws on semantic conventions, web standards, and NLP to make information on webpages more machine processable. We intend to achieve this by creating a system that extracts data from webpages that provide semantic information, and then want to analyze the extracted information while creating relations and store these data for later use.

To identify and address the weaknesses in how web page data extraction is currently done, we review different annotation tools in this study, and point out deficiencies that a new system can improve on. The main goal is to develop a first proof of concept of the proposed solution. Future work will therefore have to further prototype and test the tool designed in this thesis.

1.3 Outline

The structure of this thesis is as follows. Chapter 2 reviews different annotation tools used for the semantic web and evaluates the usability for our purpose. Chapter 3 specifies the specific problems and challenges to be solved within the development of a new semantic annotation system. Implementation details of the presented approach are described in Chapter 4. A user manual of the created system that includes step by step instructions on how to apply the software can be found in Chapter 5. Chapter 6 concludes this thesis and explores several possible future developments that can improve the current system. Finally, a glossary of all relevant terms is found in Chapter 7.

Chapter 2

State of the Art of Semantic Web Tools

2.1 Introduction

In this chapter we will analyze several semantic approaches, techniques, concepts and tools regarding their functionalities, usability and re-usability. In section 2.2 we will discuss content extraction tools. Section 2.3 provides an overview of semantic search engine browsers. In section 2.4 annotation tools are reviewed. Section 2.5 describes the state of the art of semantic social bookmarking tools, and tools using machine learning approaches are reviewed in section 2.6. Subsequently, we compare these standards and tools and draw conclusions in section 2.7.

2.2 Content Extraction Tool

Content extraction tools are used to scan text based web pages to extract specific information. A good example of content extraction tools is SiteScraper.

2.2.1 SiteScraper

SiteScraper is a semi-automatic tool that is able to extract content from a web page via XPath based patterns.[8] Figure 2.1 provides an example.

Functionalities:
The functionalities are very basic. A user has to provide a sample string and an URL[9] as input. SiteScraper will scrape the URL for the given example string and once similar strings are found in the web page's content, SiteScraper then generates a XPath based location. Now a user can use SiteScraper to scrape another URL and if the scraped web

page contains the already generated XPath, SiteScraper returns all the extracted strings found, related to that given XPath. Since SiteScraper focuses on the visible user content rather than on the HTML code of the web page, it should not matter if the structure of the web page changes. SiteScraper simply adapts the new XPath by itself and returns the correct information.

```
>>> from sitescraper import sitescraper
>>> ss = sitescraper()
>>> url = 'http://www.amazon.com/s/ref=nb_ss_gw?url=search-alias%3Daps&field-keywords=python&x=0&y=0'
>>> data = ["Amazon.com: python", ["Learning Python, 3rd Edition",
     "Programming in Python 3: A Complete Introduction to the Python Language (Developer's Library)",
     "Python in a Nutshell, Second Edition (In a Nutshell (O'Reilly))"]]
>>> ss.add(url, data)
>>> # we can add multiple example cases, but this is a simple example so 1 will do (I generally use 3)
>>> # ss.add(url2, data2)
>>> ss.scrape('http://www.amazon.com/s/ref=nb_ss_gw?url=search-alias%3Daps&field-keywords=linux&x=0&y=0')
["Amazon.com: linux", ["A Practical Guide to Linux(R) Commands, Editors, and Shell Programming",
"Linux Pocket Guide",
"Linux in a Nutshell (In a Nutshell (O'Reilly))",
'Practical Guide to Ubuntu Linux (Versions 8.10 and 8.04), A (2nd Edition)',
'Linux Bible, 2008 Edition: Boot up to Ubuntu, Fedora, KNOPPIX, Debian, openSUSE, and 11 Other Distributions']]
```

Figure 2.1: SiteScraper: Basic example on how to extract book titles from amazon using SiteScraper

Re/Usability:

Even though SiteScraper is only a tool for a simple task, like for example extracting book titles on a product search engine, it provides simple re-usability for a user with programming skills and therefore can be used to automate scraping routines in scripts or integrate them into programs that can talk python as part of a complex analyzer consisting of multiple scraping and analyzing tools. The Documentation of SiteScraper states that, to use SiteScraper, one does not need Programming or HTML skills. But setting up and using SiteScraper does appear to require some level of programming and computer knowledge, especially when working with Linux or in a Console.[8]

2.3 Semantic Search Engine Browser

Semantic search engine browsers are tool that let you browse through aggregated web data provided by RDF or ontology models without the functionality to expand existing information, therefore we will review the tools RelFinder, Piggy Bank, MashQL, Sig.Ma, and Magpie in this section.

2.3.1 RelFinder

RelFinder extracts and visualizes relationships between given objects in datasets and makes these relationships interactively explorable. Highlighting and filtering features support analysis both on a global and detailed level. The RelFinder is based on the open source framework Adobe Flex, easy-to-use and works with any RDF dataset that provides standardized SPARQL access. [10][11]

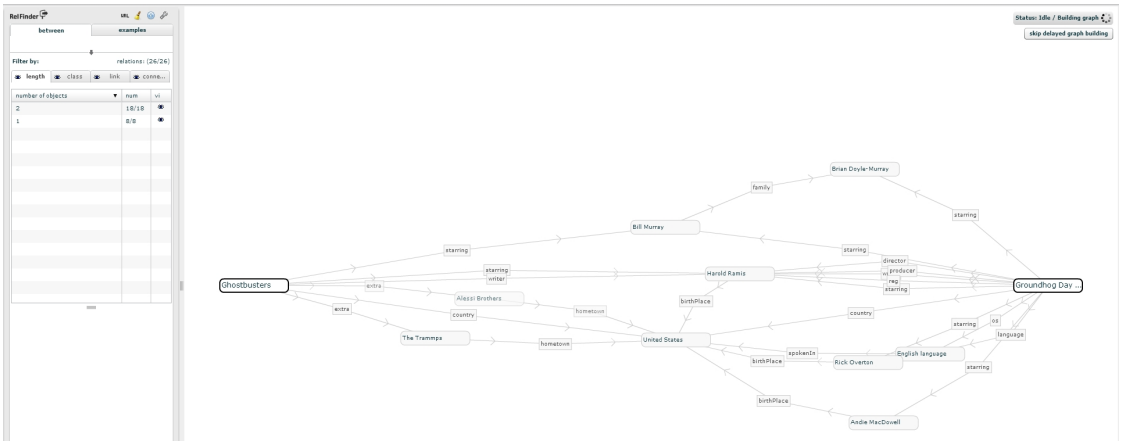

Figure 2.2: RelFinder: The interface

2.3.2 Piggy Bank

Piggy Bank is a Firefox extension that turns your browser into a mashup platform, by allowing you to extract data from different web sites and mix them together. During browsing, it collects semantic information in RDF format, possibly by using automatic mechanisms for extracting semantic annotations from (X)HTML pages. [12]

2.3.3 MashQL

MashQL is a query-by-diagram mashup language. The idea of MashQL is to allow people to query, mash up, and pipeline RDF data intuitively. In the background MashQL queries are automatically translated into and executed as SPARQL queries. The novelty of MashQL is that it allows one to query (and navigate) a RDF graph without any prior knowledge about its schema, vocabulary, structure or technical details; it also supports query pipelines as a built-in concept.[13][14]

2.3.4 Sig.Ma or Semantic Information Mashup

Sig.Ma is an aggregated Semantic Web Search System, that combines the use of Semantic Web querying, logic reasoning, machine learning, external services, responsive user interaction and more, to create rich entity descriptions that form the base for machine oriented as well as data browsing services. Sig.ma builds on top of Sindice.[15][16] Figure 2.3 shows the user interface of Sig.Ma.

Functionalities:

Sig.ma operates as a service and an end user application. Just like normal search engines a user starts with entering a search phrase. The difference is that the presented information is a rich aggregation about the entity identified from the search phrase combined from different vertical searches (images, documents, news, etc.), for example weather forecasts and maps of entered city names. Sig.Ma also offers rich interaction tools to expand and modify the aggregated information about a given entity via accepting and declining (a part of) the whole result. Therefore, it is possible to create Sig.Ma views with persistent URLs for those entities that can be viewed and embedded in external HTML pages or shared with other users. Additionally, once a Sig.Ma view has been created, a user can subscribe to the view to receive alert messages when the view has been modified.

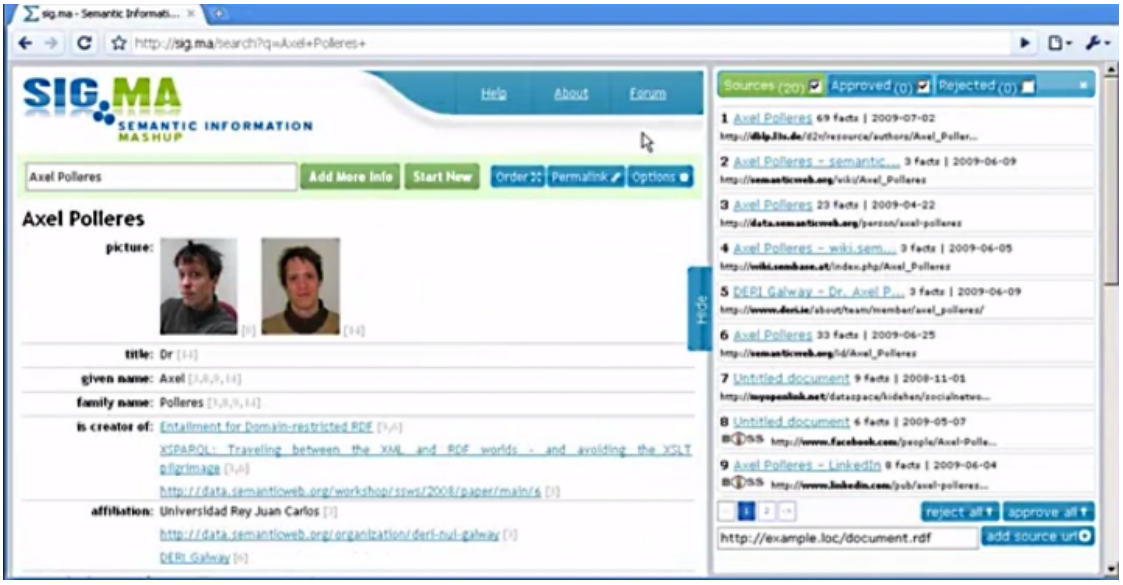

Figure 2.3: Sig.Ma: on the left the aggregated horizontal search results and on the right the sources that represent the results

Re/Usability:

Sig.Ma provides an API so a user or an application can query Sig.Ma for a list of properties and entities such as "address, picture, phone number @ Hannes Rammer, TU Berlin". Sig.Ma relies on RDF, RDFa and micro-formats, which means that if a page does not contain such information it will not return any results. Unfortunately, the Sig.ma homepage is down and no further information is available. [15]

2.3.5 Magpie or Media Access Generator

Magpie is a plug-in for Internet Explorer 4. Magpie's first goal was to create a semantic web browser that allows to navigate through a web page and its created semantic content at the same time. The later version based on Java allows a web of inter-operable applications while enabling bidirectional exchange of information. The exchange is invoked either by a user or triggered automatically by browsing activity.

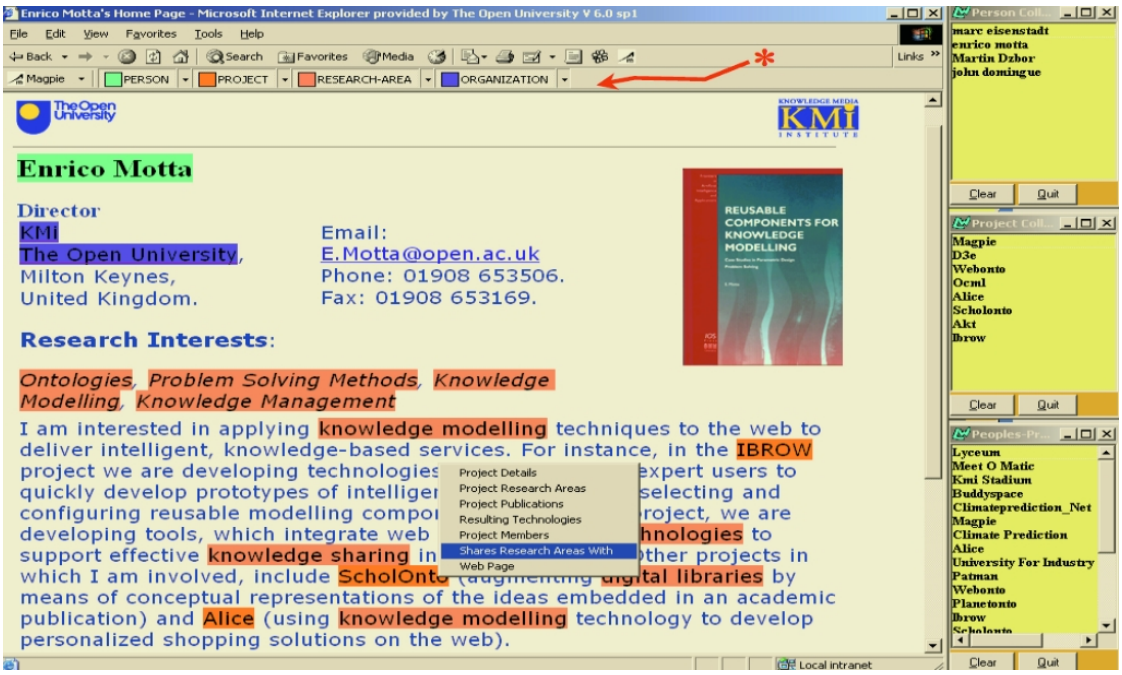

Figure 2.4: Magpie's left web page with extra semantic layer; right side shows the found concepts and their relations (magpie tool-bar marked with *)

Functionalities:

A user loads an ontology map which is a text file of n-tuples describing concepts. After loading a web page, Magpie automatically matches items in the web page via simple lexicon-based parsing to the active concepts from the ontology model. It will then

highlight the found concepts by adapting the HTML code to add an additional layer of semantic information while the original document layout remains unchanged. This enables a user to bring up a menu of services created for the highlighted concepts, like identifying all the members working on a specific project.

Re/Usability:
The usability and integration of annotated concepts seems clean and easy to handle. Magpie supports loading of ontology, which a user has to create by himself via third party tools. Therefore, a restriction is that a user can not update the ontology with new information while visiting a web page. An additional problem of Magpie is Magpie's incompatibility. Magpie 1.0 is an Internet Explorer extension which therefore comes with a few downsides due to Microsoft's restrictions between different product versions: for example Magpie 1 only runs on Internet Explorer 4 and even though Magpie 2 is build as a Java plug-in, it will only work on systems using any Windows version before Windows 7 or Mac-OS 10.4; therefore it is not attractive to use Magpie in new projects.[17][18][19][20] Figure 3 shows the user interface of Magpie.

2.4 Annotation Tool

Annotation tools provide the option to annotate unstructured information using ontology models with the possibility to train the system to improve future information extraction processes. The tools we assess in this section are Melita, MnM, SHOE, Annotea and AeroDAML.

2.4.1 Melita

Melita is a semi-automatic annotation tool using Amilcare, an Adaptive Information Extraction tool based on the Language Processing 2 [1] algorithm to support a user during the annotation of text documents.[21] The interface consists of three areas: [22]

1. The ontology view which shows the existing concepts and relations of the loaded ontology model;
2. The document view in which text will be selected with the mouse;
3. The Information Extraction System (IES) suggestions area, where a user can give feedback to the information extraction system;

[1] LP2 - http://www.cs.rit.edu/ ats/projects/lp2/doc/

Figure 2.5 shows the user interface of Melita.

Functionalities:

Melita has two phases, the continuously performed annotation training and the actual annotation. A user does not need to be aware of these phases, but only has to load an ontology model containing concepts and relations and a text document. Next a user can start annotation by selecting text with the mouse and then click on the suitable concept. While a user annotates the text, the IES analyzes saved annotations and at some point automatically starts suggesting more annotations inside the IES suggestion area until a user approves, corrects or declines those suggestions. Only accepted suggestions are highlighted in the document. Melita does not differ in appearance from other annotation interfaces like Gate or MnM, apart from demonstrating how to integrate user interaction with the IES.

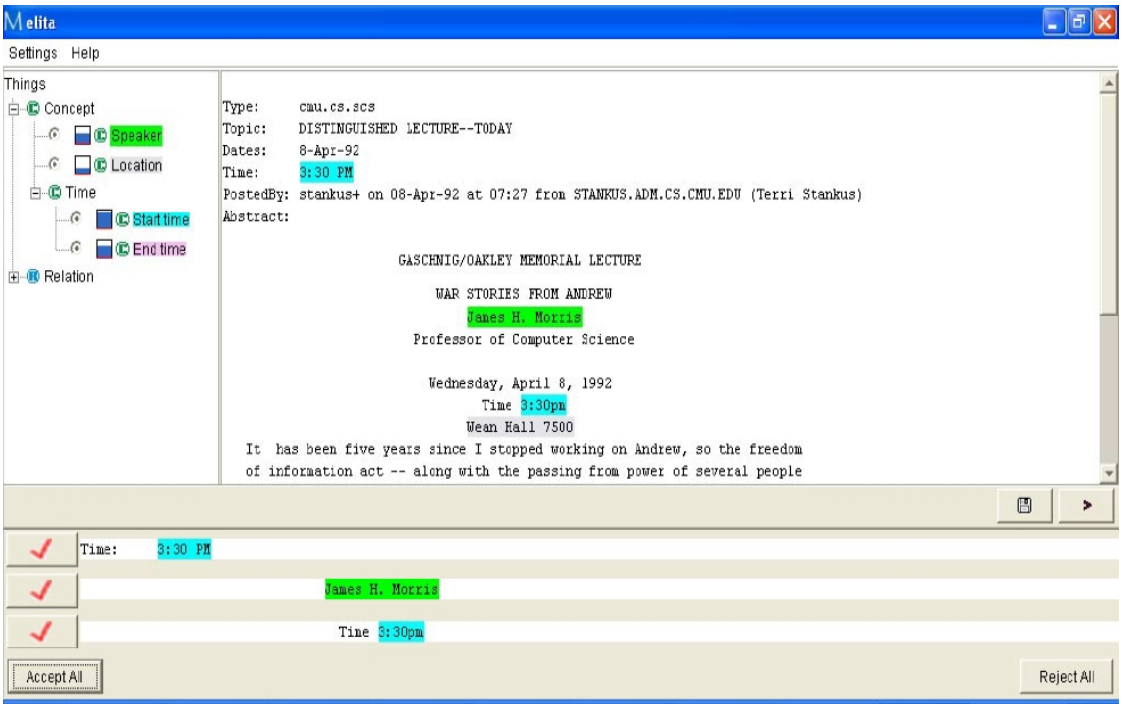

Figure 2.5: Melita: left - ontology model view, middle - text document, bottom - system suggestions

Re/Usability:

Melita's user interface is very simple to use. A user does not need any special computer knowledge. Melita only concentrates on text documents, not on webpages, therefore making the program itself less interesting to reuse. But since Melita only wanted to

demonstrate the benefits of interacting with the IES, it seems very interesting to reuse the concept for future semantic web tools. Melita has been replaced by AKTive Media, but all links related to the latter project [23] point to a dead end.

2.4.2 MnM

MnM is an annotation tool which provides both automated and semi-automated support for annotating webpages with semantic content. MnM integrates a web browser with an ontology editor. MnM can be seen as an early example of the next generation of ontology editors, being web-based, oriented to semantic markup and providing mechanisms for large-scale automatic markup of web pages. MnM makes it possible to access ontology servers through APIs, such as OKBC, and also to access ontologies specified in a markup format, such as RDF and DAML + OIL. MnM can handle multiple ontologies at the same time, which makes it very easy to switch from one to another, and also allows inherited definitions to be displayed for ontology editing and browsing. MnM's annotations are stored both as markup on a page and as items in a knowledge base.[24]

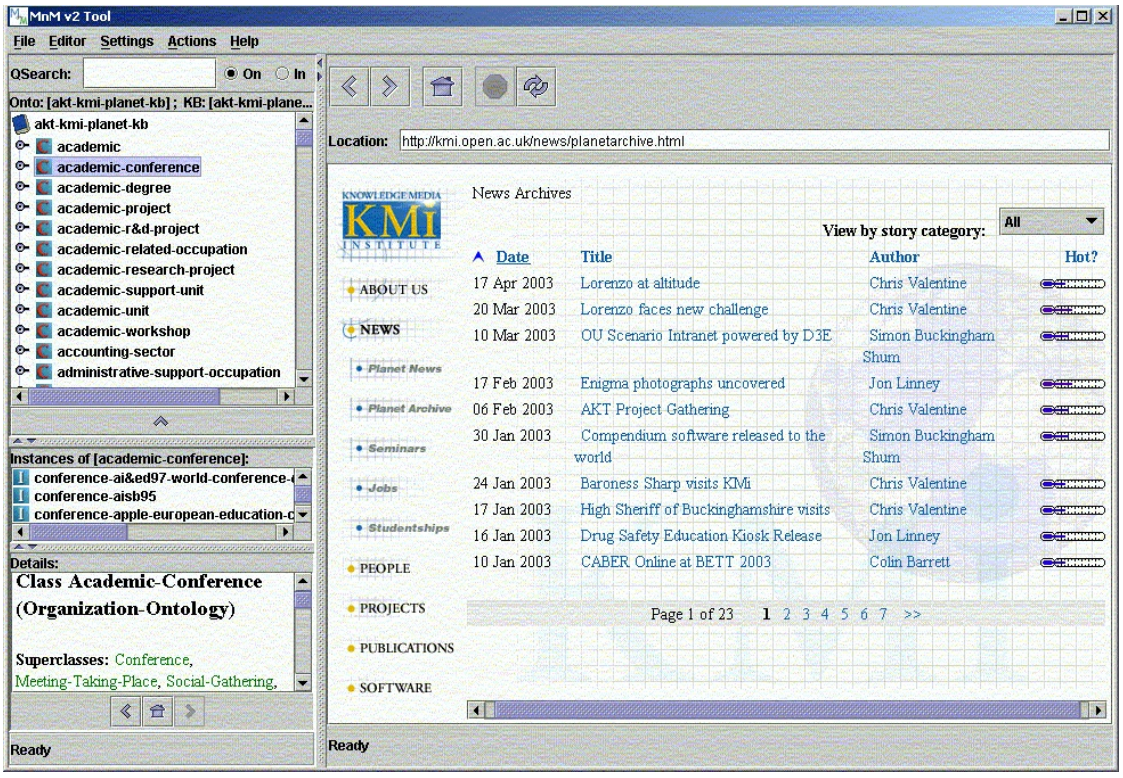

Figure 2.6: MnM: user interface of the ontology brwoser

2.4.3 AeroDAML

AeroDAML is available as a web page. The user simply enters a URL and the system automatically returns DAML annotations on a web page using a predefined ontology based on WordNet.[25][26]

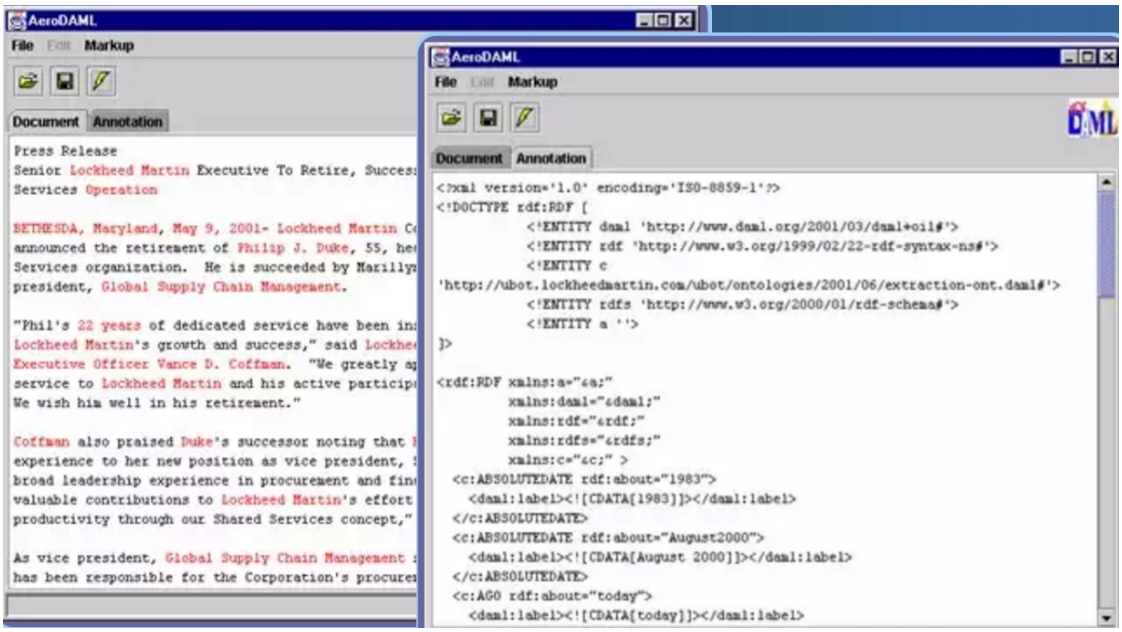

Figure 2.7: AeroDAML: user interface of the annotation tool

2.4.4 SHOE

SHOE is a knowledge annotator that allows users to mark up pages guided by ontologies available locally or via a URL. These marked up pages can be reasoned about by SHOE-aware tools such as SHOE Search, tool that allows you to query SHOE information that has been discovered by a web robot written in Java which searches out web pages with SHOE entries called Expose.[27]

2.5 Semantic Social Bookmarking

Social bookmarking serves to give people the chance to create a set of semantic information via links and then share them with each other. In this section we take a closer look at Delicious, the renewed tool for cross-platform semantic social bookmarking.

2.5.1 Delicious

Delicious [2] is a social bookmark annotation platform created in 2003, with bookmarking features such as commenting a bookmark so people can have conversations about the web page and even tag bookmarks to group or mark them for later re-usage. In 2013 it has been renewed to enrich a user experience and ease the use of bookmarking. Delicious stores bookmarks online for either public or private viewing. Figure 2.8 illustrates the web view of Delicious.

Functionalities:

Delicious is a very basic yet complex bookmarking tool. It allows a user to save links inside an online bookmark system that other user's can search, find or even comment and share on. A user also has the possibility to follow other users bookmark streams, or get bookmark suggestions depending on the tags a user has already stored. For example, one user stores a picture of a shirt, another user can comment on this picture's bookmark that he would like to know where to buy that shirt, and a third user might just respond with a link to a shop. All these bookmarks are stored as a bookmark stack, thus allowing users to interact on an additional layer of information.

[2] Delicious - https://delicious.com

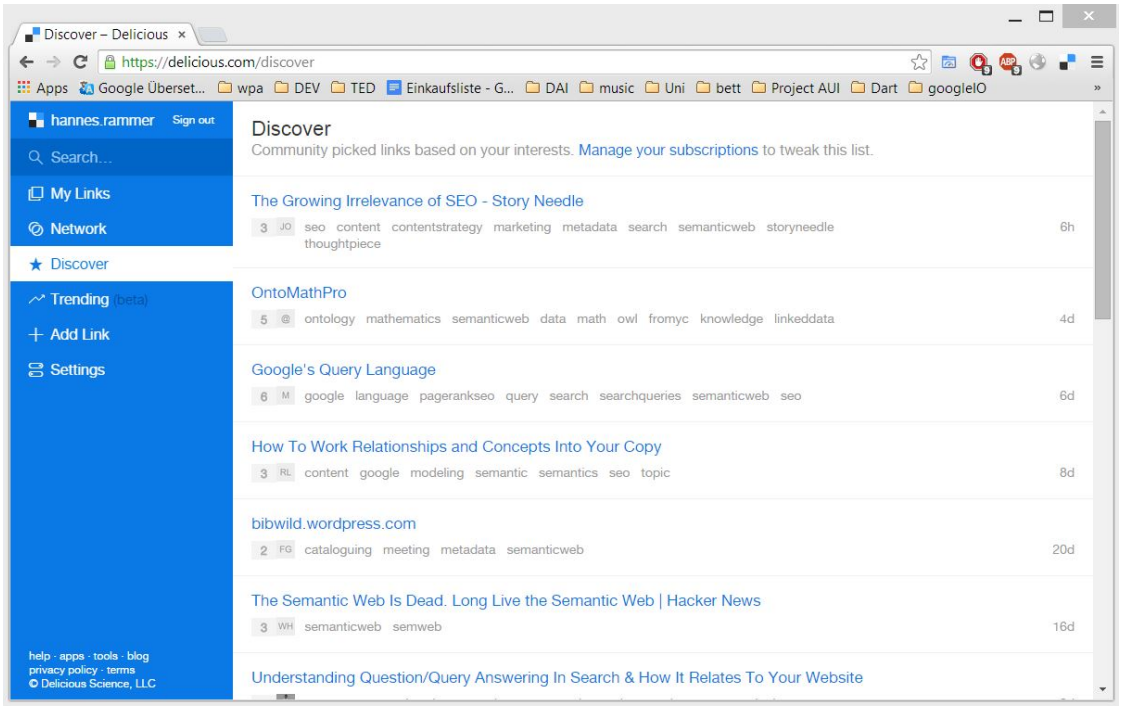

Figure 2.8: Delicious: web-view of social online bookmarking system

Re/Usability:

Delicious is a tool that has recently been updated, unlike most other tools that stopped working after a few years. After 10 years, this tool is now supported on iPhone, iPad, Android for phone and tablets, Firefox Social API integration, a Firefox OS APP and a Chrome browser extension. It even provides a HTML tag save button for developer integration and a bookmarklet link in case none of the above tools will work on your device. Even though this tool is for personal use only, Delicious also offers an API to reuse all those functionalities inside a third party service. Delicious surely got around the incompatibility issues that many other tools failed to solve.

2.5.2 Annotea

Annotea provides RDF-based markup but it does not support information extraction nor is it linked to an ontology server. It does, however, have an annotation server which makes annotations publicly available.[28]

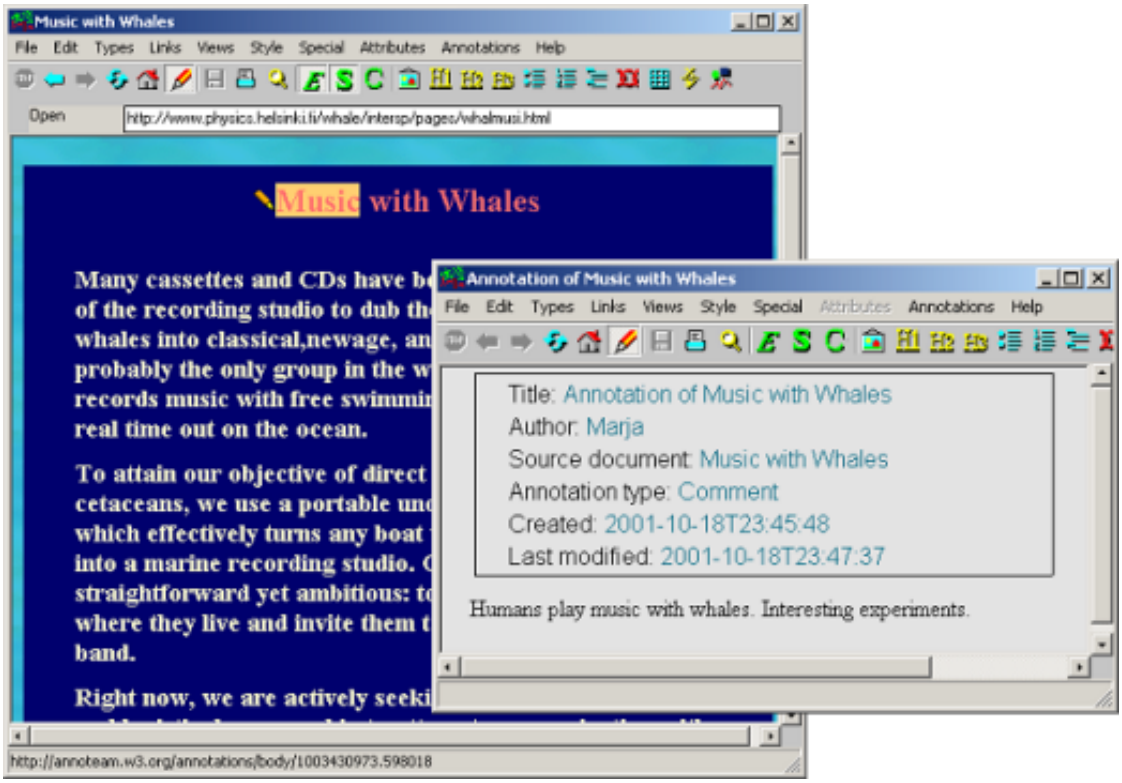

Figure 2.9: Annotea User interface

2.6 Machine Learning approach

In this section we review tools that make use of machine learning like WIEN, Rapier, Whisk, and SoftMealy.

2.6.1 WIEN

WIEN stands for Wrapper Induction Environment that serves to build wrappers for labeling web pages, with low human engagement needed. WIEN application draws on the following 4 steps, illustrated with building a wrapper for the Lycos search engine: [29]

1. **Domain specification**: The user can specify attributes to be extracted;

2. **Gathering & labeling examples**: Using a standard Internet browser, the user gives WIEN a set of example pages as well as the text to be extracted from the pages by using the mouse. Building the wrapper: Once the examples have been gathered, the user simply invokes a Build Wrapper command. Once learned, the

wrapper can be tested on additional examples; if the wrapper makes mistakes, the user only needs to correct them and re-invoke the learner;

3. **Source mode**: In some cases (like extracting URLs), the text fragments that are aimed to be extracted are not rendered by the browser; WIEN handles such cases with its integrated HTML source mode;

4. **Recognizers** : To simplify the task of labeling the examples, WIEN provides a facility to automate this process. When started, WIEN loads a dynamically maintained library of recognizers. A recognizer is a procedure for examining a page and identifying "interesting" text fragments;

2.6.2 Rapier

Rapier is a system to learn rules for extracting information from documents. Rapier can learn rules without the need of prior parsing or post-processing. It uses pairs of sample documents and filled templates to induce pattern-match rules that directly extract fillers for the slots in the template.

Rapier is a bottom-up learning algorithm that incorporates techniques from several inductive logic programming systems and acquires unbounded patterns that include constraints on the words, part-of-speech tags, and semantic classes present in the filler and the surrounding text.[30]

2.6.3 Whisk

Whisk is a supervised learning algorithm that generates content based rules in case of text, and delimiters in case of structured documents and can even handle a combination of both. Whisk is designed to handle text styles ranging from highly structured to free text, including text that is neither rigidly formatted nor composed of grammatical sentences. Such semi-structured text has largely been beyond the scope of previous systems. When used in conjunction with a syntactic analyzer and semantic tagging, Whisk can also handle extraction from free text such as news stories by learning text extraction rules automatically.[31]

2.6.4 SoftMealy

SoftMealy is a non-deterministic finite state atomata using bottom up induction learning rules to learn extraction rules. SoftMealy is a novel wrapper representation formalism.

This representation is based on a finite state transducer (FST) and contextual rules. This approach can wrap a wide range of semi-structured Web pages because a FST can encode each different attribute permutation as a path. A SoftMealy wrapper can be induced from a handful of labeled examples using a generalization algorithm.[32]

2.7 Resume

The internet accumulates a huge amount of unstructured data that is not directly available for machine processing. To change that, around 2000 the semantic web was introduced. Since then many semantic methods, tools, and standards have been developed to give unstructured data on the web meaning by applying semantic information to it. But one important aspect of the semantic web has been neglected for a while, as stated by Berners-Lee in 2006:

"There are many hurdles to overcome before the concept the Semantic Web can become a success. One is that the Semantic Web must be based on a unified data structure which does not currently exist in the database technology employed.".[33]

It sure is difficult to automatically extract and interlink information across a large scale of web pages when these do not employ a unified data structure. But even though Berners-Lee made his statement already in 2006, no one tried to tackle the problem of vocabulary incompatibility till 2010, when Schema.org was launched by Google, Microsoft and Yahoo because there where too many different versions of what a person would look like, thus creating a massive fragmentation of vocabularies (N versions of person).[34]

Not having a unified data structure made it more difficult for web masters to decide which vocabulary to use, and therefore harder for search engine web crawlers to scan web pages for the most reasonable information that represents the web page's content. For example, without conventions in naming entities, different web developers might use different schema's for their webpages representing the same instances of concepts. Having multiple concepts of a person, one concept where the person name is divided into two properties, surname and first_name and another concept where its just one property called name. This will result in extra work needed by developers to find similarities, and semantic connections between similar items.

We therefore concentrated to find the most appropriate and newest tools available, but in this respect, it was difficult to find tools that where published after the creation of Schema.org.

Table 2.1: Comparison table of reviewed semantic tools

Name	Alive	UDS	OOTB	First Release	Tech used	Available for
SiteScraper	yes	no	yes	2010	XPath	JavaScript
RelFinder	yes	no	yes	2010	RDF,SPARQL	FLASH, Website
Piggy Bank	no	no	yes	2005	RDF	Not Available
MashQL	yes	no	yes	2008	RDF,SPARQL	Fire Fox 3.5
Sig.Ma	no	no	?	2011	RDF,RDFa	Not Available
Magpie	no	no	no	2002	OM, NLP	Internet Explorer 4
Melita	no	no	no	2002	NLP,OM	Not Available
MnM	no	no	no	2002	RDF,DAML	Win 2000
Annotea	no	no	no	2001	RDF	—-
AeroDAML	no	no	yes	2001	OM,DAML	Website
Delecious	yes	no	yes	2003/2013	Tags	Website, Android, iOS

The following key terms are used in the Table 2.1;

- **Alive**: If the source code or the running project is somehow still available;
- **Unified Data Structure (UDS)**: If the tool uses vocabulary conventions such as provided by Schema.org;
- **First Release**: When the tool was released;
- **OOTB**: If the tool works without any extra configuration;
- **Tech used**: The semantic technologies used ;
- **Available for**: The Software requirements for the tool;

When looking at Table 2.1 that compares all reviewed tools excluding the machine leaning approaches, we can see that none of the tools makes use of conventions for an unified data structure. This is a huge limitation to the systems reviewed. In addition to vocabulary incompatibilities, another downside for current systems is that most of them rely on RDF/RDFa and microdata which have a high percentage of bad or wrong markup.[35] Moreover RDF has a slow adoption rate, with coverage of less than 100k sites by 2013 compared to the adoption rate for Schema.org. Since its establishment, the vocabulary of Schema.org has spread to over 10 Million websites and covers over 25 Billion entity references by 2013. Therefore, adopting Schema.org into semantic tools will allow us to have access to a very interesting source of structured data.[35]

In the case of massive amounts of reliable and correctly structured data, this approach could allow to extract semantic information from webpages without semantic markup in a much more precise manner then when working with sources that are as error prone and limited as RDFa.

Interestingly, Schema.org was somehow a threat to RDF since the three major search providers came together to not only create a new system, but they created a model that is significantly simpler than the model used in RDF and ignores the RDFa syntax. Schema.org also did not imply the RDF standards like RDF Schema and OWL nor the proposed way of implementing non-information resources. [36]

In this MSc thesis, we therefore want to move beyond the reviewed approaches by envisioning a semantic aggregation system that can perform better when building on state-of-the-art conventions in semantic markup using Schema.org.

We will Develop such a system and then test how it performs in recognizing entities on websites without markup, after it has been trained by a large consistent set of web-

sites that provide good semantic markup. In this MSc thesis, we focus on a subset of categories and properties provided by Schema.org.

We explore and explain the issues and challenges arising from this project more closely in the next chapter.

Chapter 3

Problem Specification

3.1 Introduction

In this section we identify the problems we want to tackle with our approach. In Section 3.2 we will talk about fragmentation of vocabularies and some of the consequences that current annotation tools and standards like RDF have to deal with. Section 3.3 states the proposed solution to the found problems.

3.2 Vocabulary incompatibilities

Till the establishment of Schema.org there where too many different versions of what a person would look like, thus creating a massive fragmentation of vocabularies (N versions of person) which makes it hard for web developers to decide which vocabulary to use. the incompatibility is not just on items but on their properties as well. one might call a person human instead. or use multiple properties for the name instead of a single one. Thus making recognition of similar items more difficult without the use of ontology matching. Annotation also still requires a huge amount of work by web developers, who need to learn not only about the Vocabularies but about the existing annotation techniques and methods and determine which one is the most suitable for a specific task. This is also a reason why many individual web-crawlers have to be created for individual web pages, which again is time consuming and unreliable.

To solve these problems, Schema.org provides an interconnected unified vocabulary of terms that can be added to the HTML markup of a Web page to communicate the meaning of concepts. [37]

To motivate web developers to use the unified vocabulary from Schema.org, and to boost accuracy of their own search engines, Google, Microsoft, and Yahoo have been promising a future of better web page ranking if those conventions will be used.[35]

However it seems that even though Schema.orgs's adoption rate in the web is remarkable, it seems that that it has not been included by scientific approaches yet. Even Though we tried to find the newest annotation tools available in 2015, only 27 % of the tools reviewed in Table 2.1 where created in 2010 or later. Only 9% where created after the introduction of Schema.org, but there still is no tool to make use of the unified data structure available.

With this being such a potential source of reliable data it it should not be neglected by science.

3.2.1 Incorrectly marked information

Moreover, the annotation tools currently available rely on information (mainly RDF and RDFa) provided by web developers and on recognizing relations from looking at the structural features of a web page. Additionally, a problem with tools that rely on microformat and RDF/RDFa data is the percentage of bad or wrong markup. Microformat has about 25% incorrectly marked information while RDFa even has about 40% of wrongly marked data, making it a unreliable source of information. In addition, both RDFa and microformat appear to have a slow adoption rate by developers with less than 100.000 sites. Also none of the previewed tools make use of the conventions and standards introduced by Schema.org which compared to RDFa by 2013 covers over 10 Million websites and over 25 Billion entity references. [35]

3.2.2 Out of the box

The tools reviewed earlier in this thesis that use OM and NLP do not appear to provide a way to create new concepts, but only to recognize existing concepts loaded from an external model. This forces a user to combine different tools, which add to the problem described in Section 3.2 as well as making the whole annotation process unnecessarily more complex.

The downside of the annotation tools discussed in this thesis is that many of the tools are not working any longer because they are deprecated by the developer team, sold to private companies or due to hardware/software incompatibilities that came with newer versions of operating systems and web browsers. When looking at the tools reviewed in Table 2.1 there also appears to be a connection between projects being still alive or

working and the creation of a closed system. In this respect it also is more difficult to keep a system running when it relies on many different system components.

3.3 Proposed solution

In view of the problem analysis in the previous sections, this master thesis will serve to explore *how a semi-automated annotation system would perform in extracting and interconnecting, when using a reliable source of data and software standards.* We will draw on the most reliable source and standard currently available, that is, Schema.org conventions and standards to develop a new tool that combines these conventions with long term compatibility in mind. As such, we want to see how good the solution developed in this thesis can overcomes the problems of vocabulary incompatibilities and incorrectly marked information.

A detailed approach towards developing this tool is described in the next Chapter.

Chapter 4

Approach

4.1 Introduction

In section 4.2 we discuss the overall approach towards creating a semi automatic semantic aggregation and annotation system. including the proposed system workflow in terms of three components: analyzer, storage, and mapper. Sections 4.3 to 4.6 then outlines the design of the analyzer, storage, mapper, and browser components. In section 4.7 we describe the technologies used. Finally, section 4.9 provides a evaluation of the developed system as well as a conclusion of this chapter.

4.2 System

The overall goal for this thesis is to create a semi-automatic semantic annotation aggregation Browser that builds on semantic conventions using Schema.org's unified data structure and relying on web standards for longterm software and storage compatibility. This approach involves the following principles:

1. Building an analyzer to extract microdata from webpages with structured semantic information;
2. Create relationship mappings for
    (a) Item Scopes and their property Item Scopes, e.g. between a movie and its actors property;
    (b) Identical Item Scopes from different websites, e.g. two different definitions of the same actor;

(c) Related property Item Scopes from similar Item Scopes.

3. Store extracted concepts, their properties and the relationship mappings in a storage system for data accumulation;

4. Enable browsing through accumulated semantic Item Scopes:

4.2.1 Workflow

The workflow of the system is outlined in Figure 4.1. The workflow is as follows:

Workflow

1. A webpage is loaded;

2. In case the webpage provides semantic markup, in the form of Schema.org microdata, the system will extract and process the information found;

3. The semantic information is analyzed, relations and stored in a storage system;

4. The aggregated information is then made available through our implementation of a semantic browser;

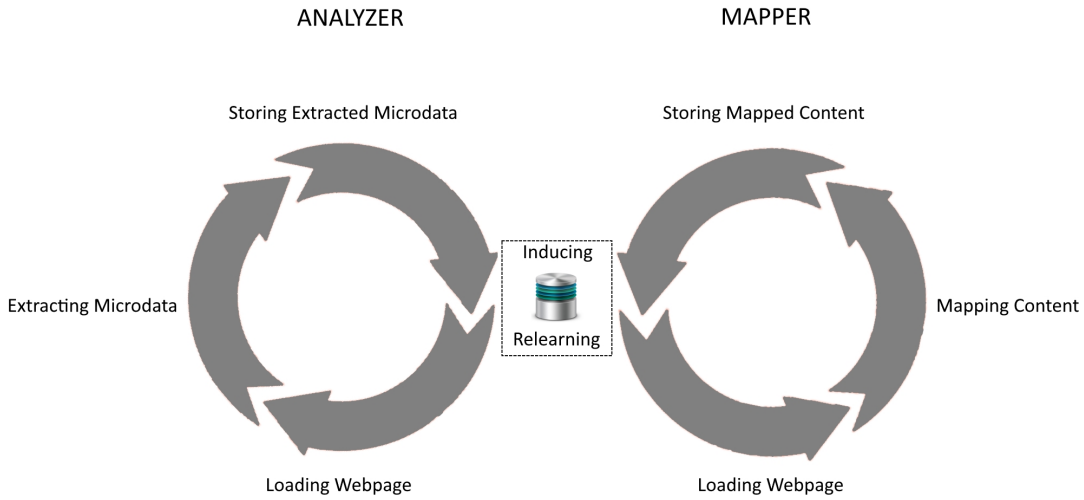

Figure 4.1: Workflow

4.2.2 Components

The system consists of 4 components: the analyzer, the mapper, the storage and the semantic browser, a graphical user interface (GUI). Components 1-3 are shown in Figure 4.2.

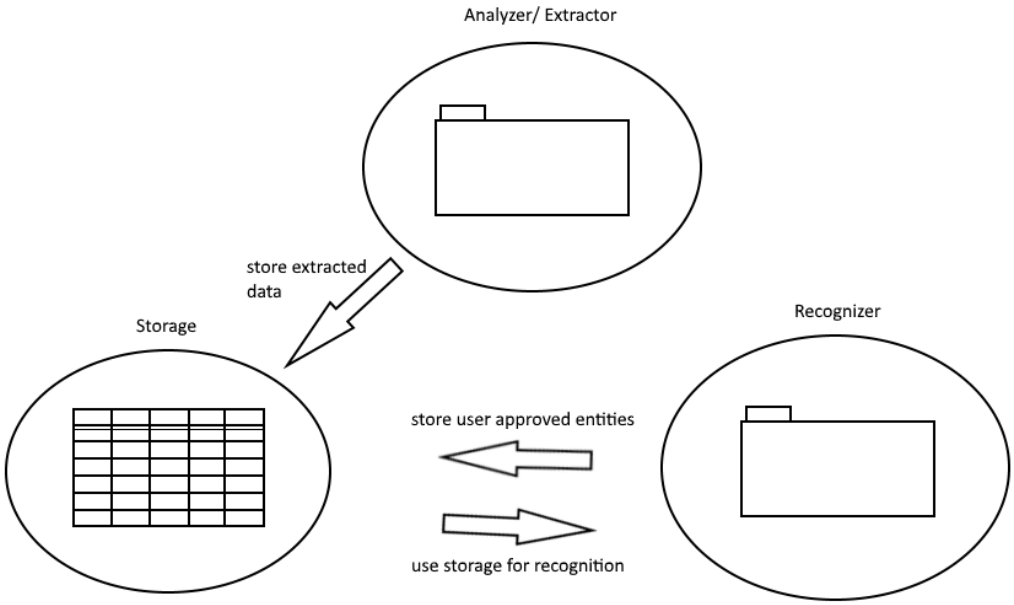

Figure 4.2: System components

4.3 Analyzer

The first Step is to build an analyzer that is able to extract Schema.org specific markup from a webpage's HTML code. Following the Schema.org documentation, we need to scan the code for three microdata tags. The first tag is Itemscope that defines the start of a new Schema.org item inside the HTML code. The ItemType tag specifies the Schema.org type. Figure 4.3 illustrates this for a Movie as the item contained in the div, which is specified in the same tag as Itemscope. And the ItemProperty tag, which is included using additional span tags, defines a property of the specific item, such as the name of the movie or its director.

```
<div itemscope itemtype ="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <span>Director: <span itemprop="director">James Cameron</span> (born August 16, 1954)</span>
  <span itemprop="genre">Science fiction</span>
  <a href="../movies/avatar-theatrical-trailer.html" itemprop="trailer">Trailer</a>
</div>
```

Figure 4.3: HTML code for creating a simple Itemscope - ItemProperty connection

In addition, in the process of analyzing the item properties, we have to take into account that an item property itself can be defined as another item scope with its own set of properties, as seen in Figure 4.4 and 4.5. In Figure 4.4 the movie director is now specified as an Itemscope itself of the Person type, containing the properties name and birth date. These properties need to be treated in the same way as normal item scopes, with the addition that we now also have a relation between two items. [38]

```
<div itemscope itemtype ="http://schema.org/Movie">
  <h1 itemprop="name"&g;Avatar</h1>
  <div itemprop="director" itemscope itemtype="http://schema.org/Person">
  Director: <span itemprop="name">James Cameron</span> (born <span itemprop="birthDate">August 16
  </div>
  <span itemprop="genre">Science fiction</span>
  <a href="../movies/avatar-theatrical-trailer.html" itemprop="trailer">Trailer</a>
</div>
```

Figure 4.4: HTML code for creating a layered Itemscope - ItemProperty connection

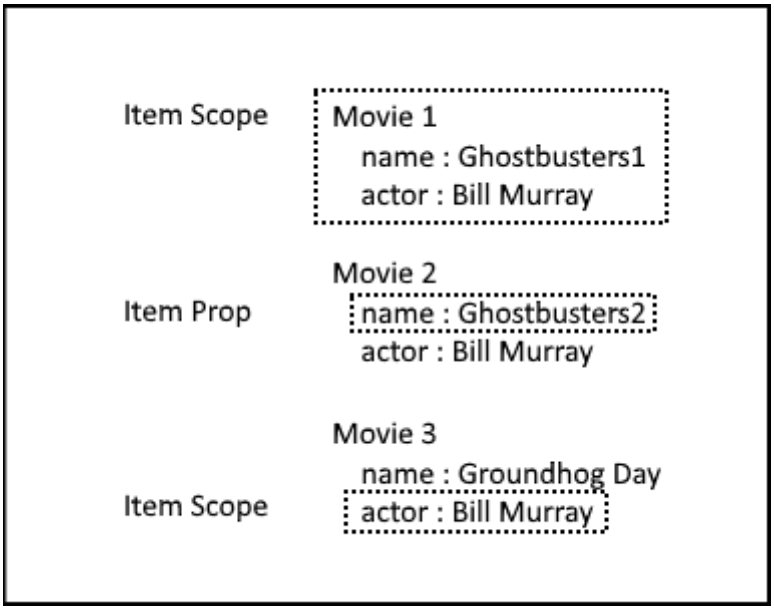

Figure 4.5: extracted semantic information from three website, illustrating ItemScope and ItemProperty

After data has been extracted from the webpage it is ready to be processed and stored.

4.4 Mapper

There are several ways to identify similar instances of a concept on a webpage. In this study we concentrate on the direct text comparison which thanks to Schema.org's unified data structure should allows to create reliable connections between the ItemScope, ItemType and ItemProperty properties of recognized objects.

The relations we want to create are as follows:

1. Item Scopes and their property Item Scopes, e.g. between a movie and its actors property;

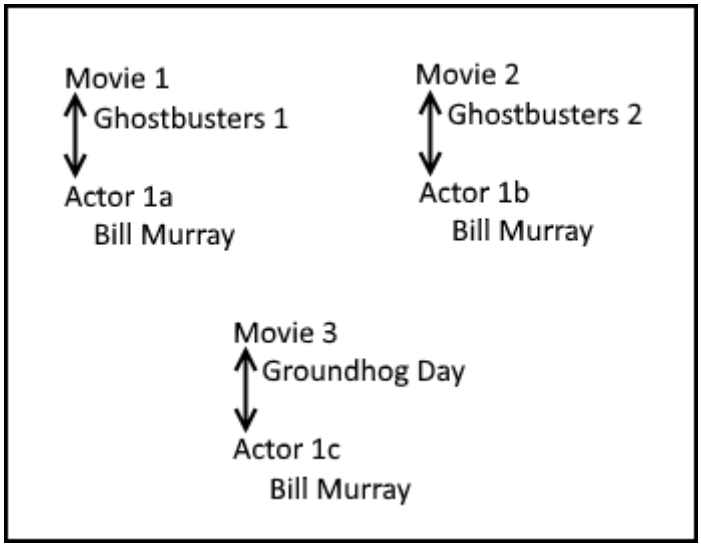

Figure 4.6: Mappings created between Item Scope property property (Actor) and Item Scope (Movie)

   To generate a connection between the given semantic information provided by the website. Figure 4.6 shows a mapping of 3 Movies and their direct related Actor ItemScopes.

2. Identical Item Scopes from different websites, e.g. multiple definitions of the same person;

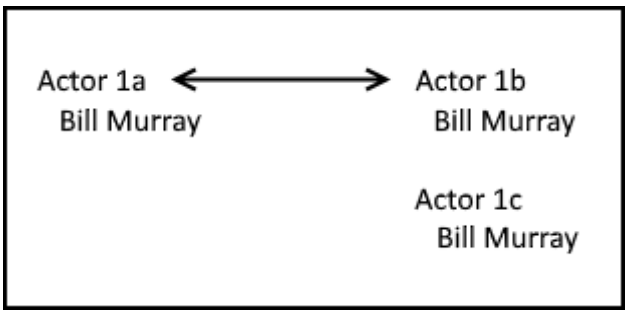

Figure 4.7: Direct connection mapping view

Next we want to create a link between similar items, this can be done in many ways, but instead of creating a connection between every single pair of items like in Figure 4.7 we chose to connect similar items by using a cross_connection_id as seen in Figure4.8. This will keep the database much smaller, even though storage nowadays is normally not a limitation, and in addition speed up queering the data.

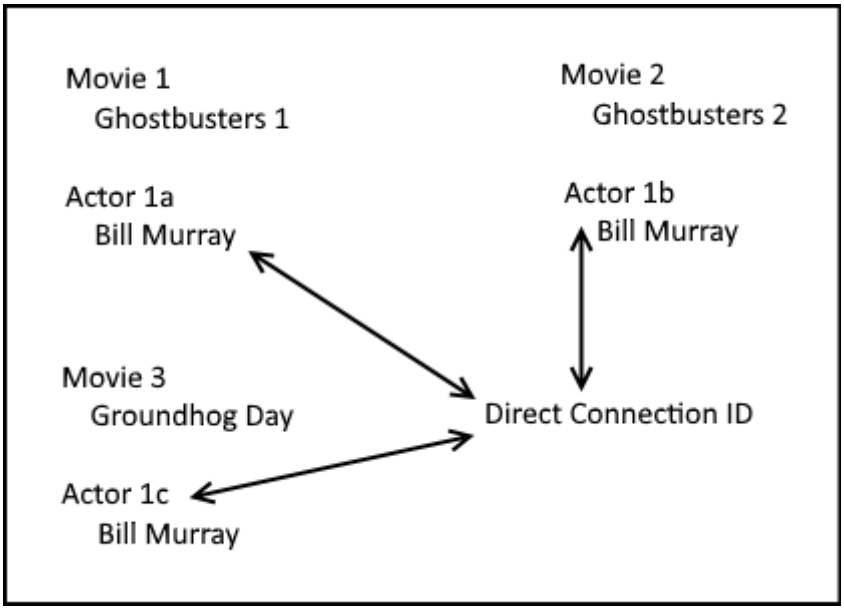

Figure 4.8: Direct connection mapping between the same Item Scopes from different websites

We can calculate the number of possible unique sets by using the formula below, while using our approach, the maximum number of database entries for all similar items will be N. :
Lets say

$$N = number\_of\_identical\_Item\_Scopes$$

Then

$$\frac{(N) * (N-1)}{2} = nr\_database\_entries$$

When having 30 similar Item Scopes of the same instance, for example the same actor in different movies, a product with different prices, or even publishers of books, instead of storing 435 database entries we only store one list of 30 entries.

To give an idea on how this will look with larger numbers:

Say we have 10.000 connected Item Scopes, we would have to generate 49.995.000 database entries.

When dealing with a huge datasets like the 25 billion Schema.org entities available, by using a cross connection id, we can prevent a lot of overhead.

Compared to a simple one to one relation, this approach of interconnecting similar information already shows advantages when only dealing with sets that include four similar instances of an Item Scope

Once the database holds more than 4 similar item by using a separate identification we can reduce entries in the database, which will resolve in a faster system.

3. Related property Item Scopes from similar Item Scopes;

   Last we want to be able to find indirectly related objects, meaning objects that are related through similar ItemScopes.

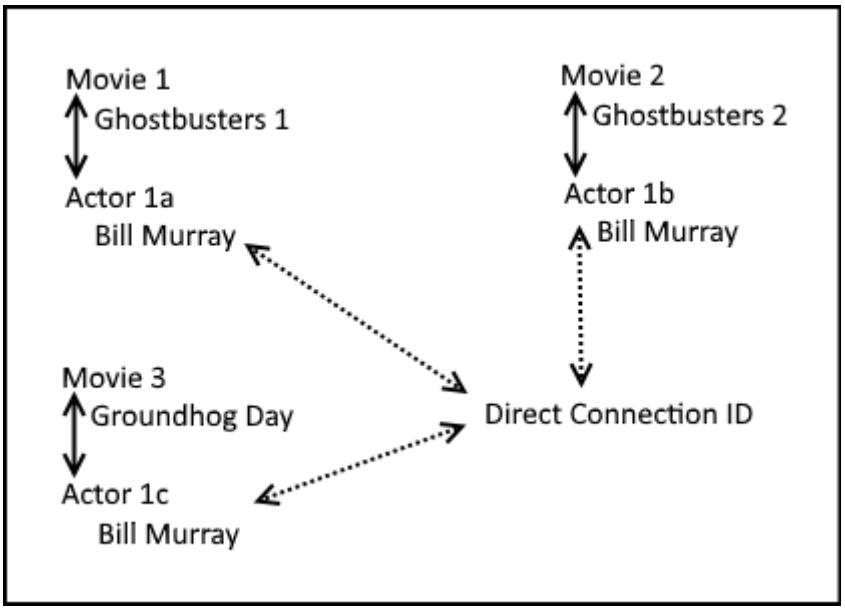

Figure 4.9: Indirectly related data

Creating such relations allows us to browse through the semantic data, like finding other movies of the same actor or different prices of the same products. Illustrations of this is shown in Figure 4.9 and Figure 4.10

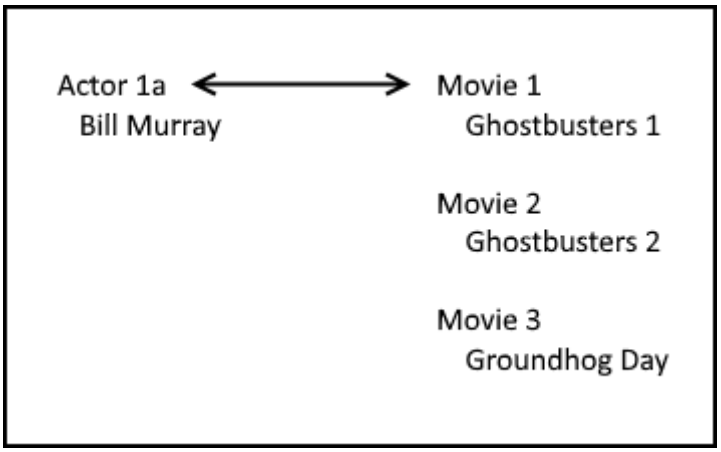

Figure 4.10: Cross connected relations between an actor and movies of other instances from the same actor

The Mapper uses the extracted information to compare and interlink aggregated content, returning all found entities and relations.

Moreover, in extracting information from webpages, it is important to avoid problems that may occur due to structural changes on webpages. Therefore, the approach adopted here will only inspect structural relations as provided by Schema.org, thus marking information related to one another by context and not by structural location. For example, within the scope of a movie, properties provided by the movie‘s property list will be recognized, independent of whether the order of the movie properties changes from for example 1.title, 2.actors, 3.release to 1.title, 2.release, 3.actors.

Another way is to generate a fixed regular expression that will find certain results, like phone numbers or credit card providers. This kind of mapping often provides a faster way to recognize elements. The research area of regular expression provides the opportunity to automatically generate regular expressions, where a regular expression will be generated from analyzing a list of positive and negative examples.[39] But, since this process is very time consuming and rather complex, we will not further explore this option.

4.4.1 Threshold

By comparing each type and value of all ItemScope properties from ItemScopes of the same type, we are able to find every connection that exists. But once the relations are created, we want to be able to specify to what extend the objects relate, thus we do not always want to know that two objects are related. As seen in Figure 4.11 two different actors are only directly connected by the jobTitle. But to be able to see how strong two items are related we need to calculate a relationship percentage. We then can apply the threshold to filter only ItemScopes that are related by 50% or more.

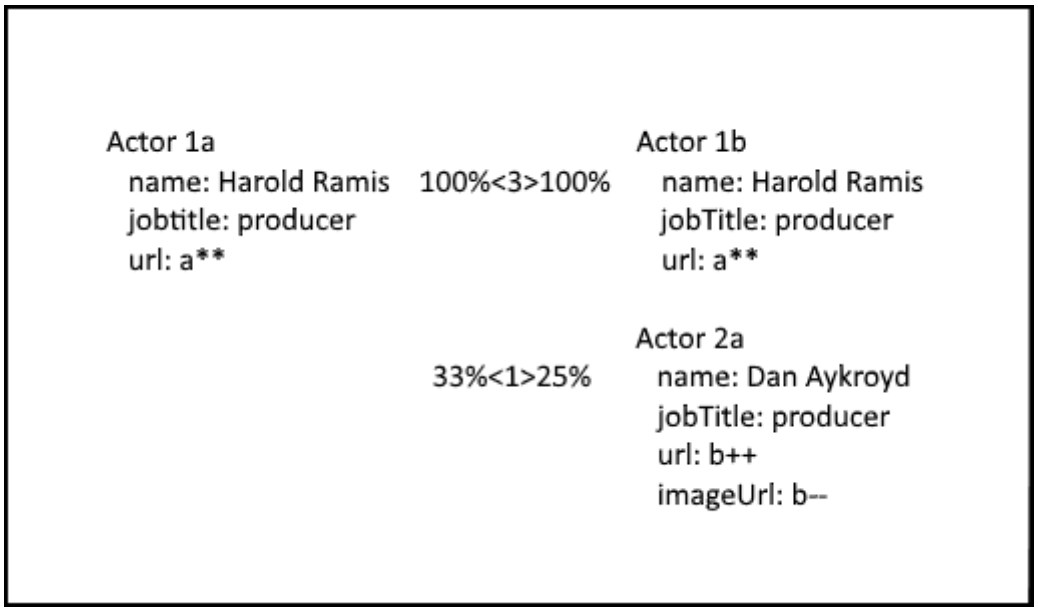

Figure 4.11: Shows the similarity percentage of connected ItemScopes

The Similarity Percentage is currently the only one used. it seems as if the same approach could be useful on comparing the content of values, by comparing and interconnecting properties of ItemScopes of the exact same type when the item It is possible to change different types of Threshold, currently it is set to find identical Items but it can easily be adapted to recognize relations between for example different ItemTypes.

4.5 Storage

Each of the extracted Itemscope entities will be separately saved in a JSON object inside the variable analyzer.jsonHash, including its textual content, the webpage's source URL and a unique scope and/or property Id. Itemscope properties which themselves

that are marked as Itemscope entities will too be stored as an separate entry with reference, while a reference between the connected Itemscopes is stored inside the variable ItemScopesToContainedItemScopes.

After retrieving and mapping multiple microdata entities of the same instance from different webpages, the storage system stores the interconnected ItemScopeIds and their cross connection id inside the variables directCrossConnectionToItemScopeHash, to retrieve all similar Itemscopes via using one single id.

Vise verse, we also stored the reversed connection between ItemScopes and crossConnectionId inside directItemScopeToCrossConnectionHash Finally the connected ItemProperties, mapping two ItemScopes together, are stored in PossibleItemScopeRelationHash.

The Variables including an Example are illustrated below:

- jsonHash : stores each found microdata entity of Itemscopes, that have been extracted from webpages in a JSON database-like structure, additionally a unique id is generated for each ItemScope and item property

- itemScopesToContainedItemScopes : Stores the relation between Items and their directly related ItemScope properties.

  For Example : {"0":[1],"1":[0]} means ItemScope with id 0 and id 1 are related.

- directCrossConnectionToItemScopeHash : Creates a new id that is used to map multiple items together. For Example : {"0":[4,5,9]} means ItemScope 4, 5 and 6 are connected via direct connection id 0 and one are related.

- directItemScopeToCrossConnectionHash : Additionally we create a connection between an Item Scope and its direct_connection_id to generate a one on one relation.

  For Example : {"4":0,"5":0,"9":0} is the reversed connection of directItemScopeToCrossConnectionHash and means ItemScope 4, 5 and 6 are connected via direct_connection_id 0.

- possibleItemScopeRelationHash : Stores all found relations between ItemScopes through their ItemScope properties

  For Example : {"4-5":["3-1"]} means that ItemScope with id 4 is connected through property_id 3 with ItemScope five's property with id 1.

Additionally there are **directCrossConnectionToItemScopeHashWithT**, **directItemScopeToCrossConnectionHashWithT** and **possibleItemScopeRelationHashWithT**, which are equivalent to their counterparts without the ending "WithT". The difference is that they only store connections that receive a bigger relationship percentage than the set threshold. Currently the processed information can be accessed and extracted via the browsers LocalStorage within the used web browser.

4.6 Semantic Browser

The extracted meta data is displayed in an extra window (to keep the webpage clean from unnecessary markers) and, in addition, works with the extracted information. More detailed information is given in Chapter 5.

4.7 Technologies Used

To support long-term and cross-platform compatibility, the systems will be build using web standards like HTML5, CSS3 and JavaScript. Moreover, we utilizes the W3C semantic standards and conventions provided by Schema.org.

For our system the content storage system will be build using HTML5 LocalStorage, mainly to keep our storage system compatible to all major browsers (chrome4+, firefox3.5+, safari4+, Opera10.5+, Internet Explorer8+, iOS Safari3.2+, android2.1+,and Opera Mobile11+). Due to security standards, stored information can only be accessed by pages from the same web domain.

While analyzing data on the same web domain we are able to aggregate a vast amount of information. The LocalStorage standards suggest that it is possible to store complex objects inside the key value pairs, but the majority of browsers only supports text values to be stored. In order to store complex object structures, JSON.stringify() and JSON.parse() methods can be used.

Converting between objects and string brings some downsides, like slowing down the analysis and recognition process due to poor performance when working with large data structures.

LocalStorage is only used as a storage point until the user decides that enough data of the specific web domain is collected.

4.8 Evaluation

To test the system, we have extracted a number of ItemScopes from http://www.imdb.com, interlinked them and analyzed problems that we where able to observe.

	set1	T=0%	T>30%	T>50%	T>80%
pages	187				
ItemScopes	3752				
property relations		19207	9454	1540	34
similar items		497 - 99	477 - 99	437 - 96	36 - 3

	set2	T=0%	T>30%	T>50%	T>80%
pages	unknown				
ItemScopes	630				
property relations		952	125	2	2
similar items		195 - 39	138 - 55	114 - 44	3 - 1

Figure 4.12: Above: Automatized tests set; Below: Manual test set

In Figure 4.12 we see the results for 2 test sets, the first set automatically extracted by using the developed IMDB autoextractor, the second one manual selection via a user. The wording is described below:

- pages: Number of visited pages.
- ItemScopes: Number of extracted ItemScopes.
- property relation: Number of relations between two ItemScopes via a property.
- similar items: Left: Number of ItemScopes that have a similarity with another ItemScope; Right: number of ItemScopes after treating connected ItemScopes like a single one.
- t=x%: percentage of threshold, that specifies how many similarities two objects need to be treated similar.

After testing the system it seem that having a unified data structure shows potential. Even though Schema.org's conventions and standards provide Solutions for incompatibility in vocabulary, there still seems to be a lot of confusion on how to use the markup

correctly. Either there are no conventions or web developers implementing the markup are just to unexperienced.

With the following examples we demonstrate some aspects where the unified data structure from Schema.org fails due to wrong or inadequate markup and therefore does currently not provide a way around using OM.

4.8.1 Unclear Markup Definition

Schema.org currently allows multiple structural definitions for the same Instance. In Figure 4.13 when looking at the person objects with id 0, 6, 9, and 312, we find 4 totally different implementations, which make it very difficult to extract easily data without adoption of the crawler for specific objects.

Figure 4.13: Example of unclear markup since multiple instances describe the same object.

For example we can see that:

- Object 0 defines a single person ItemScope with multiple properties
- Object 6 defines an actors property list with two properties for each person in the list, an url and the persons name
- Object 9 defines a single actor property with two properties , an url and the persons name
- Object 312 defines a single creator property as a list, with two properties for each person in the list, an URL and the persons name

Having that many possibilities makes it nearly impossible to extract every object correctly. A possibility to check if the instance is supposed to be a list of person or a single person, one could see if there are only two properties, and if half the properties are an URL and if the other half is the name property. But already in the example provided in Figure 4.13 we see that Person with id 6 has an incorrect number of properties, since it has a seventh property called URL.

4.8.2 Bad Implementation

Another problem is that Schema.org can not take care of the possibility that the data gets understood wrong or simply interpreted the wrong way by the user. It is good to tag as many items possible with Schema.org, one should take care they do structure them correctly, since when looking at Figure 4.14

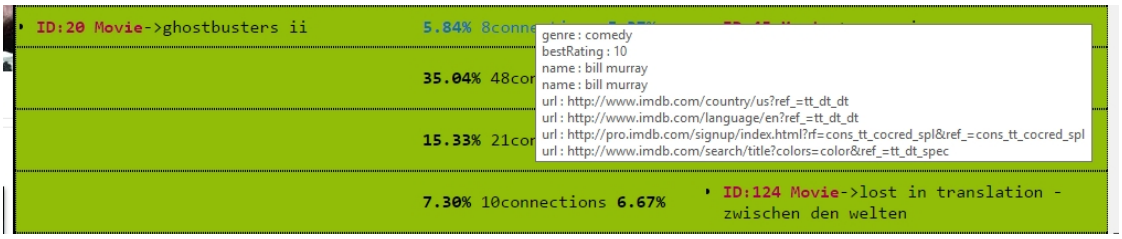

Figure 4.14: Example of bad implementation of Schema.org structure

We can see that two movies are related not just by actor but by the webpages language and country settings, they also provide a search URL and a signup URL, which definitely had nothing to do with the the movie object. it is also questionable on how HTML parameters can get ignored. IMDB for example always adds a random String as reference from where one came from. thus blocking our approach to map entities together, since after visiting an already visited page from another predecessor page, we can not be sure that the reference will be the same again. thats why we created a Filter, as seen in Figure 4.15.

```
959     cleanUrl: function (string) {
960         var ignoredUrls = ["www.imdb.com/plugins", "www.imdb.com/offsite",
961             "www.imdb.com/video", "www.imdb.com/language",
962             "www.imdb.com/country", "www.imdb.com/search", "pro.imdb.com/signup"];
963         for (var i = 0; i < ignoredUrls.length; i++) {
964             if (string.indexOf(ignoredUrls[i]) > -1) {
965                 return "";
966             }
967         }
968         if (string.indexOf("//www.imdb.com/") > -1) {
```

Figure 4.15: needed addaption of code to be able to cope with objects from IMDB

In addition we have to ignore specific ItemScope properties, since they seem to have invalid data for interconnecting items. See Figure 4.16 for example, thumbnails provided would have some kind of token attached that would change after time. or the provider of the news headline, not of the movie.

the same counts for Ajax calls like on play.google.com where the analyzer has to be executed after loading the application.

```
if (propertyObjectNodeName !== "thumbnailUrl" &&
    propertyObjectNodeName !== "image" &&
    propertyObjectNodeName !== "provider") {
```

Figure 4.16: ignore specific attributes on IMDB

4.8.3 Wrong Markup

Obviously, no matter how good a system works, it will not behave correctly if the provided information is not consistent. We can now have a look at Figure 4.17

```
ID:62 author->http://link.springer.com/search?facet-creator=%22sahin+albayrak%22
http://schema.org/Person
namehttp://link.springer.com/search?facet-creator=%22sahin+albayrak%22
```

Figure 4.17: Wronly marked markup, not as proposed by Schema.org

Here we can see that instead of using a URL and a name property, the URL gets stored in the name property instead. An additional problem with this example apart from the wrong markup is, that the URL only directs to the search page from link.springer.com, searching for the author but without there being any schemantic information available about the author itself.

In Figure 4.18 we have the same product 5 different times. Apparently ebay uses the products URL for the product name, thus we will never find the same product name twice when comparing product names.

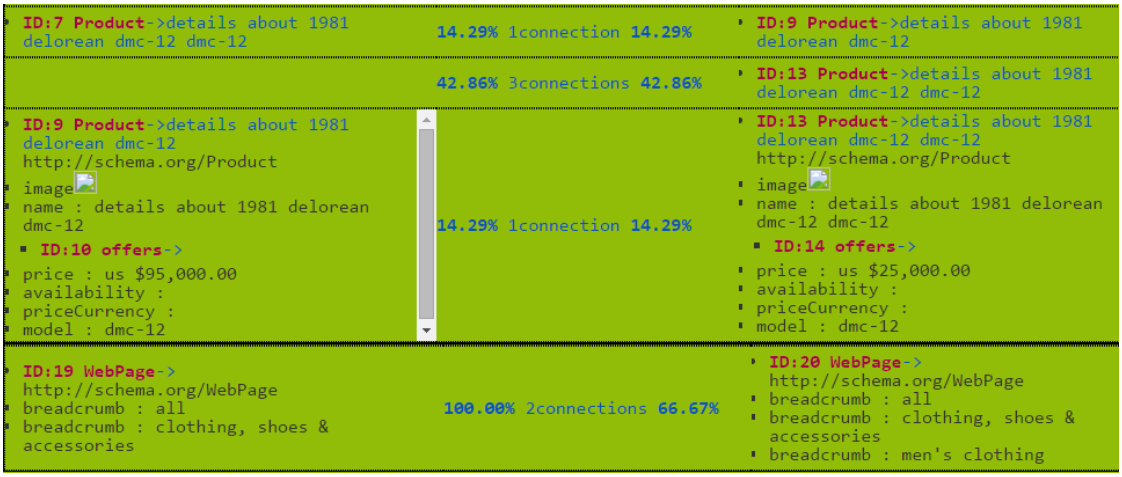

Figure 4.18: shows wrongly marked semantic data found on

Additionally they do not provide enougth additional properties, since the only properties available for any car are the model, the price, and the name. No additional semantic information about the color, about the number of doors, or the odometer, even though the information was provided on the product site.

4.8.4 Technical Limitations

Even though we overcame the limitations of browser incompatibility and other problems by using web based technology for the whole system workflow, this lead to another limitations. Currently known technical limitations are the size limitation of the LocalStorage in web browsers, it is limited to 5MB giving us the possibility to store 5 million single characters, since Javascript though uses UTF16, meaning using 2 bytes per character, we can only store 2.5 million characters.

Another technical limitation is the rendering of HTML elements inside the document view. Thus, this is only partially a limitation of the browser.

In case the user wants to use another web domain with the gathered information, one has to manually select the stringified version in the web developer consoles of the browser, and copy and paste it into the web developer console of this web domain.

LocalStorage is only used as a storage point until the user decides that enough data of the specific web domain is collected.

4.9 Resume

In this chapter, we have developed our approach towards creating a semi-automatic semantic aggregation and annotation system. We outlined the key design principles of this approach, and then proposed a system workflow in terms of three components (analyzer, storage, and mapper). Subsequently, we described the design of the analyzer, storage, mapper and browser components. Finally, we summarized the technologies we draw on in this project and evaluated some of Schema.org's aspects with the developed system. The next chapter will provide a detailed user manual.

Chapter 5

User Manual

5.1 Introduction

The user manual gives help in Setting up and using the developed tool, and while giving step by step instructions demonstrates of the key principles proposed (see 4.1). The purpose of this section is to verify that these principles have the potential of being used. The system has been tested working in following browsers and the set of websites[40]. below each website a list of extractable ItemScopes is listed.
Browsers: Chrome 44, FireFox 41, Edge 21, Internet Explorer 11
Webpages:

- https://addons.mozilla.org/de/firefox
  WebApplication, Offer, Aggregated Ratings
- https://chrome.google.com/webstore
  WebApplication, Offer, Aggregated Ratings
- http://link.springer.com
  ScholarlyArticle, Person (Author, Editor)
- http://www.ebay.de
  Product, Offers
- http://stackoverflow.com
  Question, acceptedAnswer
- http://www.imdb.com
  Movie, Person (Actor, Creator, Director), Rating

5.2 How to set up and execute the script

The next steps illustrate how to set up the system:

5.2.1 Manual Extraction

5.2.1.1 Simple

1. Open the javascript file that holds the systems code inside a text editor (see Figure 5.1);

```
var analyzer = {
    itemScopeList: [],
    itemRootScopeList: [],
    specialIdList: [],
    scopeIdList: [],
    itemList: [],
    relationList: [],
    jsonList: [],
    loadItemsIntoItemList: function () {
        //  var items2 = analyzer.getAllItemScopes();
        var items = analyzer.getAllRootItemScopesFromDocument();

        for (var i = 0; i < items.length; i++) {
            var item = items[i];
            var jsonJSTreeJsonScope = analyzer.createJSONListFromRootScopeHTML(item);
            analyzer.itemList.push(jsonJSTreeJsonScope);
        }

    }, renderItemsFromItemList: function () {
        var display = recognitionDisplay.getDisplay();

        var JSONlist = analyzer.itemList;
        for (var i = 0; i < JSONlist.length; i++) {
            var item = JSONlist[i];
            var htmlUl = analyzer.createHTMLFromJSONScope(item);
            display.appendChild(htmlUl);
        }
    },
    getAllRootItemScopesFromDocument: function () {
        analyzer.itemRootScopeList = [];
```

Figure 5.1: A screen shot of the systems code.

2. Select all text from the file and copy it;

3. Start the web browser, in this example:
   chrome v 44.0.2403.157 m;

4. Open a webpage with microdata, in this example: http://www.imdb.com/title/tt1663202/;

5. Inside the browsewr, press the F12 key once to open web developer tools as seen in Figure 5.2);

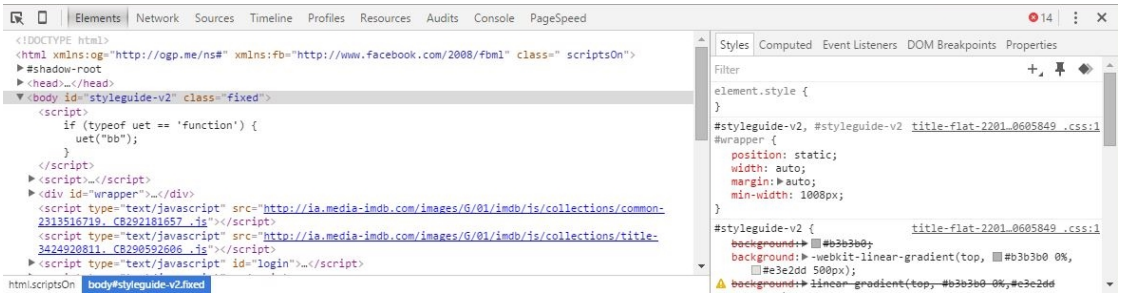

Figure 5.2: A view of the chrome web browser with the web developer tools opened

6. Now switch to the console tab;

7. Paste the code from the javascript file into the console tab and execute it.

8. The script will be executed The webpage will be analyzed for semantic data and compare it to already stored entries, mainly comparing the name property, the source URL and if available the destination URL

9. A new HTML div element named WebPageBox appears on the right top side of the window. (see Figure 5.3);

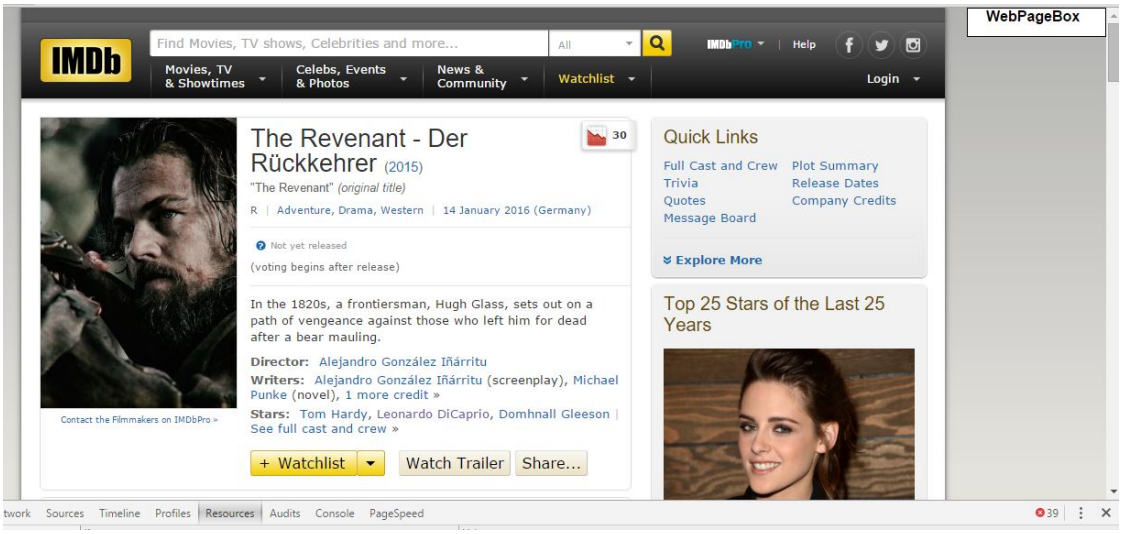

Figure 5.3: The webpage view after code execution.

10. Clicking on the new HTML div will show all the provided Schema.org microdata found by the analyzer. The WebPage tab holds all the semantic data found on the current website (see Figure 5.4)

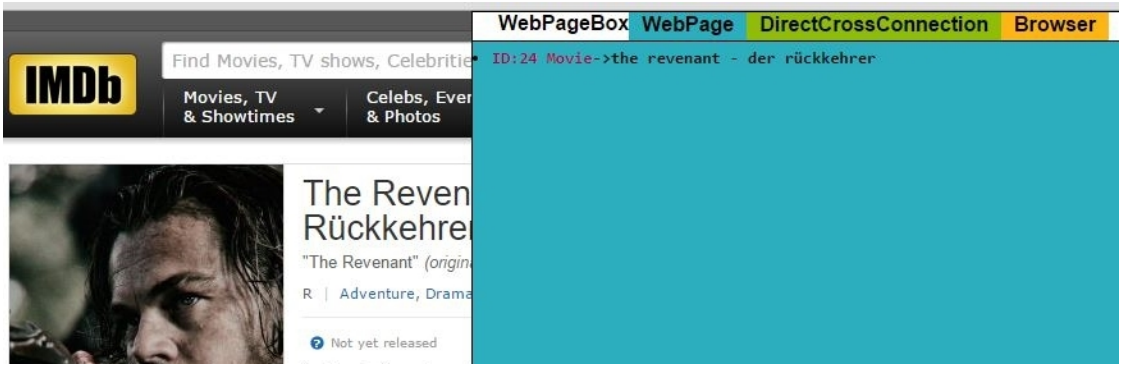

Figure 5.4: View of the open GUI of the developed tool, with information about the current website.

11. The DirectConnection tab shows connections between similar items, as-well as the properties the items are related by. when hovering over the word connection a pop up window appears showing all the properties that create the relation (see Figure 5.5)

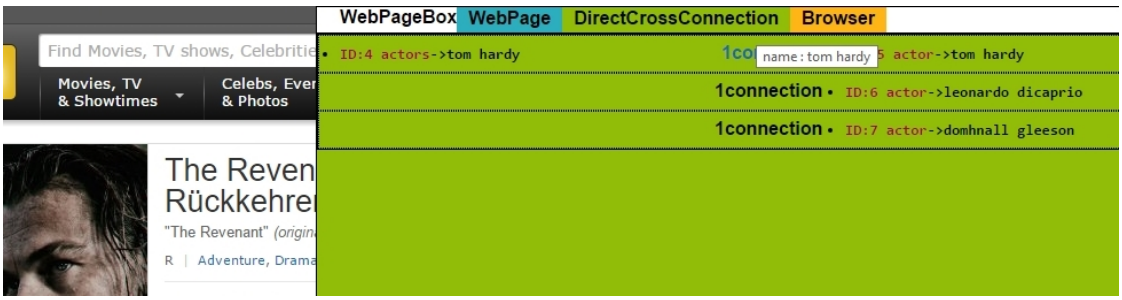

Figure 5.5: View of the open GUI, showing information about ItemScopes connection reasons.

12. The red links represent a single ItemScope. When clicking on them, they will expand, showing the ItemScopes Properties (see Figure 5.6)

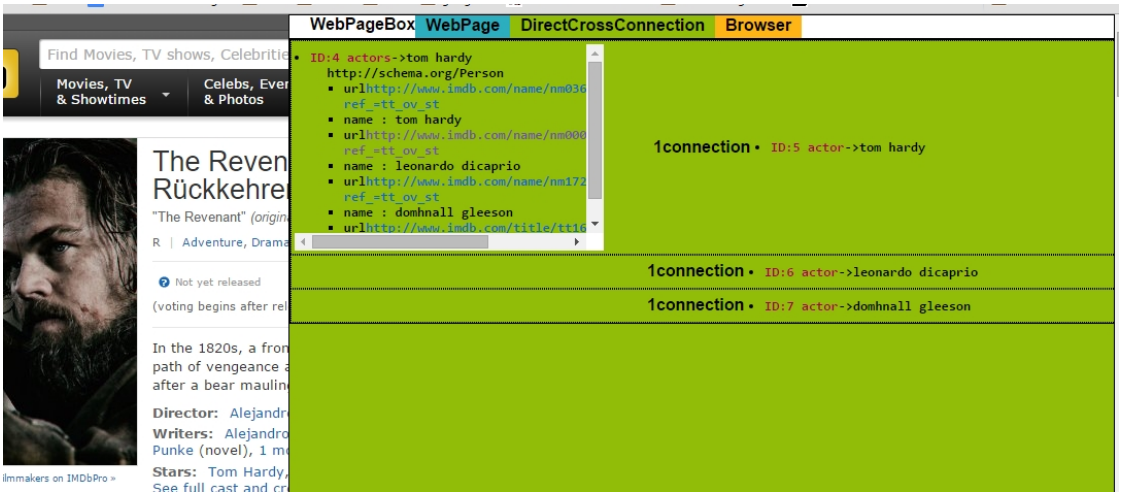

Figure 5.6: View of the open GUI, showing information about similar ItemScopes and their connection reasons. With expandable Items.

13. Lets have a look at the storage, to do this, please click on the resources tab in the chrome-developer-console (see Figure 5.7).

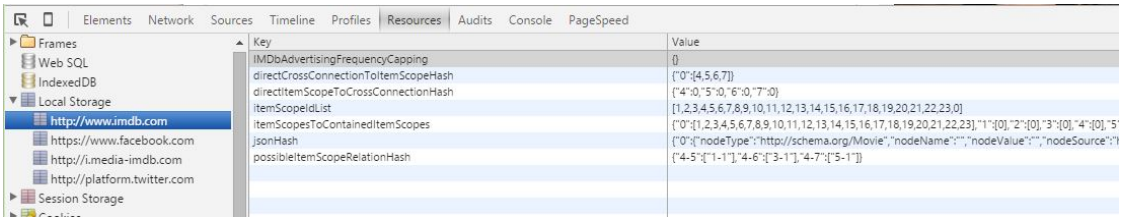

Figure 5.7: LocalStorage in the chrome web browser resources tab

14. We can see that a relation has been found between ItemScope 4, 5, 6, and 7 and we can see which property Ids connect them.

15. After collecting more data from different websites, we can try to use the semantic browser

16. When clicking on the arrow next to the red colors link named ID:4, the semantic browser tab will open, showing all directly and indirectly connection ItemScopes (see Figure 5.8).

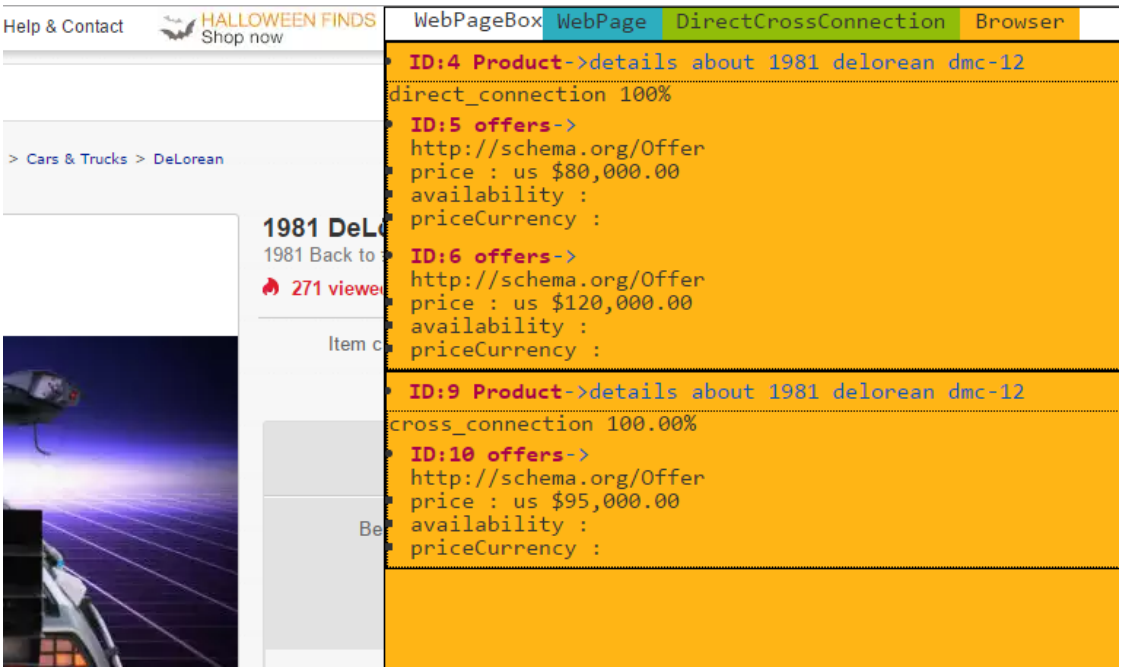

Figure 5.8: A view of the semantic browser tab showing all related Items of Item id 4

17. Now we are able to, for example explore other instances of the same product, like in the example in Fig 5.8 where we see a Delorean and that there are different offers for that car available, thus showing us there is a cheaper car available that might interest us, only by interconnection the semantic information provided.

5.2.1.2 Advanced

There are four main execution types available:

1. run.run(false, true, true, true):
   This will run the same execution as the simple version
   ->1. load storage, 2.analyze webpage, 3.map data, 4.render data, 5.update storage

2. run.run(false, true, false, false):
   this will run only the analyzer, including the steps.
   ->1. load storage, 2.analyze webpage, 3.update storage

3. run.run(false, false, true, false):
   this will run only the mapper, including the steps.
   ->1. load storage, 2.map data, 3.update storage

4. run.run(false, false, false, true):
   this will run only the renderer
   ->1. load storage, 2.render data

5. run.run(true, true, false, false):
   this will run the automatized extraction script,
   To collect testdata fast, we provide a sample snippet that will create a string for a movie on IMDB, redirect there, and then incrementally starts extracting the webpage data.

so the best approach for using these methods is: To use the analyzer on its own to be able to scan webpages quickly. To use the mapper only after all items have been extracted, since this is a time consuming task it makes sence to do it only once at the end of aggregation. To use the renderer only when all Items have been mapped and you want to display them.

5.2.2 Semi Automatic Extraction

To speed up data aggregation, the script provided can be used in combination with tool such as the Tampermonkey extension for Chrome Browser, or any other tool that supports automatic execution of javascript code. Depending on the automated execution tool used, it is important that the script is executed after the content is fully loaded. So, when extracting data from webpages that update content via Ajax calls, the script has to be executed after each content update, which might not be supported by the automation execution tool.

Chapter 6

Conclusion

Current web page annotation tools to a large extent rely on additional information provided by web developers, and mostly rely on RDFA or microformat which are error prone and not widely available. Moreover, due to the rapid development of web technologies, many of these tools are deprecated or for other (unknown) reasons no longer accessible.

In this MSc project, we developed a tool that addresses these deficiencies in web page data extraction. This tool draws on web standards and Natural Language Processing to support content extraction on web pages with semantic markup. As such, due to new semantic standards and conventions, the tool developed in this thesis serves to mark and extract large amounts of data from webpages by making use of Schema.org's unified data structure. It provides interlinking via threshold and access to the generated data via browser and console.

6.1 Future Work

The unclear markup definition problem 4.8.1 seems to be a problem that is known for years, and also has been worked on. A discussion in 2012 initiated a proposed solution by 2014, but there is no information about whether and when this approach will be adopted. Having many possibilities makes it nearly impossible to extract every object correctly. Currently the only way to check if the instance is supposed to be a list of persons or a single person, is by determining whether there are only two properties, and whether half the properties are an URL and the other half involve the name property. But already in the example provided in Figure 4.13 we see that Person with id 6 has an incorrect number of properties, since it has a seventh property called URL.

The problem of bad implementation can possibly be reduced by improving the Schema.orgs website, by providing more examples. These implementation problems may also arise from applying Schema.org to existing systems, where no structural changes of the running system are allowed. The local storage limitations inevitably force one to move to another storage, destroying the OOTB dream, by implementing an external database, or by dropping cross browser support by swithching to webSQL.

With regard to the rendering of HTML elements, there also is a lot of potential for improvement, possibly in combination with changes in the type of storage and the storage's structure which will provide faster access or have less mapping queries. While the solution proposed in this MSc thesis is limited, it offers basic functionalities which can be further developed to be capable of dealing with more complex scenarios and provide more functionality.

Chapter 7

Glossary

Automated data extraction
Automated data extraction is also called unsupervised Wrapper generation, makes use of unsupervised learning to find and extract hidden structures in unlabeled data.[41]

DARPA Agent Markup Language (DAML)
DAML is an emerging knowledge representation for the Semantic Web. DAML can encode the semantics of a document for use by agents on the web.[42]

Inductive logic programming (ILP)
ILP is a subfield of machine learning which uses logic programming as a uniform representation for examples, background knowledge and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical database of facts, an ILP system will derive a hypothesized logic program which entails all the positive and none of the negative examples.[43]

Information extraction (IE)
IE is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. In most cases, this activity concerns processing human language texts by means of natural language processing.[44]

Information retrieval (IR)
IR is the activity of obtaining information relevant to an information need from a collection of resources. Searches can be based on metadata or on full-text (or other content-based) indexing. Automated information retrieval systems are used to reduce what has been called "information overload". Web search engines are the most visible application of IR.[45]

Machine learning (ML)
ML is a scientific discipline that explores the construction and study of algorithms that can learn from data. Such algorithms operate by building a model from example inputs and using that to make predictions or decisions, rather than following static program instructions. ML is a subfield of computer science stemming from research into artificial intelligence and is employed in a range of computing tasks where designing and programming explicit, rule-based algorithms is infeasible. Example applications include spam filtering, optical character recognition (OCR), search engines and computer vision. When employed in industrial contexts, machine learning methods may be referred to as predictive analytics or predictive modelling.[46]

Natural language processing (NLP)
NLP is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and human (natural) languages. As such, NLP is related to the area of human-computer interaction. Many challenges in NLP involve natural language understanding, that is, enabling computers to derive meaning from human or natural language input, and others involve natural language generation.[47]

Ontology Inference Layer or Ontology Interchange Language (OIL)
OIL can be regarded as an ontology infrastructure for the Semantic Web. OIL is based on concepts developed in Description Logic and frame-based systems and is compatible with RDFS.[48]

Open Knowledge Base Connectivity (OKBC)
OKBC is a protocol and an API for accessing knowledge in knowledge representation systems such as ontology repositories and object-relational databases. It is somewhat complementary to the Knowledge Interchange Format that serves as a general representation language for knowledge. It is developed by SRI International's Artificial Intelligence Center for DARPA's High Performance Knowledge Base program (HPKB).[49]

Out of the Box
The term, Out of the Box, is used for a software feature or functionality of a product, that the product works without any extra configuration or modification.[50]

RDF Schema
RDF Schema is a set of classes with certain properties using the RDF extensible knowledge representation data model, providing basic elements for the description of

ontologies, intended to structure RDF resources. These resources can be saved in a triplestore to reach them with the query language SPARQL.[51]

Resource Description Framework (RDF)
RDF is a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it supports the evolution of schemas over time without requiring all the data consumers to be changed. 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?). This allows structured and semi-structured data to be mixed, exposed, and shared across different applications.[52]

Semantic Web
Semantic Web is a web of data that can be processed by machines. It extends the Web by providing a common framework that allows data to be shared and reused across application, enterprise, and community boundaries through standards by the World Wide Web Consortium (W3C). These standards promote common data formats and exchange protocols on the Web, most fundamentally the Resource Description Framework attribute (RDFa).[53]

SPARQL (sparkle)
SPARQL is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework format.[54]

Supervised learning
Supervised learning is the machine learning task of inferring a function from labeled training data, consisting of a set of training examples. Each example involves an input object (typically a vector) and a desired output value (also called the supervisory signal). The supervisory signal can then be used for mapping new examples (e.g. to correctly determine the class labels for unseen instances), by generalizing from the training data in a "reasonable" way.[55]

Unsupervised learning
Unsupervised learning is the process of trying to find hidden structure in unlabeled data. Since the examples given to the learner are unlabeled, there is no error or reward signal to evaluate a potential solution. This distinguishes unsupervised learning from supervised learning and reinforcement learning. Unsupervised learning also

encompasses many techniques that seek to summarize and explain key features of the data.[56]

Web Ontology Language (OWL)
OWL is a semantic web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL is a computational logic-based language such that knowledge expressed in OWL can be exploited by computer programs, e.g., to verify the consistency of that knowledge or to make implicit knowledge explicit. OWL documents, known as ontologies, can be published in the World Wide Web and may refer to or be referred from other OWL ontologies. OWL is part of the W3C's semantic web technology stack, which includes RDF, RDFS, SPARQL, etc.[5]

Web Data Extraction/Web Scraping
Web Data Extraction/Web Scraping is a computer software technique of extracting information from websites by simulating human exploration of the World Wide Web via low-level Hypertext Transfer Protocol (HTTP), or embedding a fully-fledged web browser. Web scraping focuses on the transformation of unstructured or semi-structured data on the web, into structured data that can be stored and analyzed in a central local storage system.[57]

Web Data Extraction Systems
Web Data Extraction Systems is a platform implementing a sequence of procedures (for example, Web wrappers) that extract information from web sources.[58]

Web wrapper
Web wrapper is a procedure that implements one or many different classes of algorithms, which seeks and finds data required by a human user, extracting them from unstructured (or semi-structured) web sources, and transforming them into structured data, merging and unifying this information for further processing, in a semi-automatic or fully automatic way. Web wrappers are characterized by a life-cycle involving three phases: generation, execution and maintenance.[59]

Wrapper generation
Wrapper generation is a process of generating a web wrapper via wrapper induction or automated data extraction using mainly the following three types of approaches: requlars expressions, wrapper programming languages and tree-based approaches.[59]

Wrapper induction

Wrapper induction also called supervised wrapper generation, makes use of supervised learning to learn data extraction rules from manually labeled training examples. Wrapper induction can be used for mapping new examples.[60]

Bibliography

[1] BERNERS-LEE, TIM, JAMES HENDLER and ORA LASSILA: *The Semantic Web*. Scientific American, 284(5):34–43, May 2001. 1

[2] W3: *Resource Description Framework*. `http://www.w3.org/2001/sw/wiki/RDF`, visited 2015-05-20. 1

[3] W3: *SPARQL Protocol And RDF Query Language*. `http://www.w3.org/2001/sw/wiki/SPARQL`, visited 2015-05-25. 1

[4] W3: *Resource Description Framework attribute*. `http://www.w3.org/2001/sw/wiki/RDFa`, visited 2015-05-22. 1

[5] W3: *Web Ontology Language*. `http://www.w3.org/2001/sw/wiki/OWL`, visited 2015-05-14. 1, 53

[6] W3: *W3C Semantic Web*. `http://www.w3.org/2001/sw/wiki/Main_Page`, visited 2015-06-18. 1

[7] SHADBOLT, NIGEL, TIM BERNERS-LEE and WENDY HALL: *The Semantic Web Revisited*. IEEE Intelligent Systems, 21(3):96–101, May 2006. 2

[8] PENMAN, BARON RICHARD, TIMOTHY BALDWIN and DAVID MARTINEZ: *Web Scraping Made Simple with SiteScraper*. 2010. 4, 5

[9] WIKIPEDIA: *Uniform Resource Locator*. `http://en.wikipedia.org/wiki/Uniform_resource_locator`, visited 2015-07-30. 4

[10] HEIM, PHILIPP, SEBASTIAN HELLMANN, JENS LEHMANN, STEFFEN LOHMANN and TIMO STEGEMANN: *RelFinder: Revealing Relationships in RDF Knowledge Bases*. In *Proceedings of the 4th International Conference on Semantic and Digital Media Technologies (SAMT 2009)*, pages 182–187, Berlin/Heidelberg, 2009. Springer. 6

[11] SEMANTICWEB.ORG: *RelFinder*. `http://semanticweb.org/wiki/RelFinder`, visited 2015-04-10. 6

[12] HUYNH, DAVID, STEFANO MAZZOCCHI and DAVID KARGER: *Piggy Bank: Experience the semantic web inside your Web browser*. pages 413–430, 2005. 6

[13] JARRAR, MUSTAFA and MARIOS D. DIKAIAKOS: *MashQL: a query-by-diagram topping SPARQL*. In *ONISW '08: Proceeding of the 2nd international workshop on Ontologies and nformation systems for the semantic web*, pages 89–96. ACM, November 2008. 6

[14] WIKIPEDIA: *MashQL*. `https://en.wikipedia.org/wiki/MashQL`, visited 2015-06-30. 6

[15] SINDICE: *SIG.MA*. `http://blog.sindice.com/category/sigma/`, visited 2015-07-25. 7, 8

[16] TUMMARELLO, GIOVANNI, SZYMON DANIELCZYK, RICHARD CYGANIAK, RENAUD DELBRU, MICHELE CATASTA, ECOLE POLYTECHNIQUE FEDERALE and STEFAN DECKER: *Sig.ma: Live views on the Web of data*. In *In Proc. WWW-2010*, pages 1301–1304. ACM Press, 2010. 7

[17] DOMINGUE, JOHN and MARTIN DZBOR: *Magpie: supporting browsing and navigation on the semantic web*. In *IUI '04: Proceedings of the 9th international conference on Intelligent user interface*, pages 191–197, New York, NY, USA, 2004. ACM Press. 9

[18] DZBOR, MARTIN: *Magpie Homepage*. `http://kmi.open.ac.uk/projects/magpie`, visited 2015-07-10. 9

[19] DZBOR, MARTIN, JOHN DOMINGUE and ENRICO MOTTA: *Magpie - Towards a Semantic Web Browser*. In FENSEL, DIETER, KATIA P. SYCARA and JOHN MYLOPOULOS (editors): *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 690–705. Springer, 2003. 9

[20] DZBOR, MARTIN, ENRICO MOTTA and JOHN DOMINGUE: *Opening Up Magpie via Semantic Services*. In MCILRAITH, SHEILA A., DIMITRIS PLEXOUSAKIS and FRANK VAN HARMELEN (editors): *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 635–649. Springer, 2004. 9

[21] SHEFFIELD, UNIVERSITY OF: *Melita an annotation interface*. `http://nlp.shef.ac.uk/melita/`, visited 2015-08-10. 9

[22] CIRAVEGNA, FABIO, ALEXIEI DINGLI and DANIELA PETRELLI: *Melita: Active Document Enrichment using Adaptive Information Extraction from Text*. In *The First International Semantic Web Conference (ISWC2002)*, 2002. 9

[23] *AKTive Media research link*. `http://www.dcs.shef.ac.uk/~ajay/html/cresearch.html`, visited 2015-06-17. 11

[24] VARGAS-VERA, M., E. MOTTA, J. DOMINGUE, M. LANZONI, A. STUTT and F. CIRAVEGNA: *MnM: Ontology Driven Tool for Semantic Markup*. In *Proceedings of the ECAI Workshop on Semantic Authoring, Annotation & Knowledge Markup (SAAKM)*, Lyon France, 2002. 11

[25] KOGUT, PAUL and WILLIAM HOLMES: *AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages*. In *First International Conference on Knowledge Capture (K-CAP 2001). Workshop on Knowledge Markup and Semantic Annotation*, 2001. 12

[26] HOLMES, WILLIAM and PAUL KOGUT: *AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages*. 2001. 12

[27] HEFLIN, JEFF and JAMES HENDLER: *A Portrait of the Semantic Web in Action*. IEEE Intelligent Systems, 16(2):54–59, March 2001. 12

[28] KAHAN, JOSÉ and MARJA R. KOIVUNEN: *Annotea: an open RDF infrastructure for shared Web annotations*. In *Proceedings of the 10th International World Wide Web Conference*, pages 623–632, 2001. 14

[29] KUSHMERICK, NICHOLAS and BRETT GRACE: *Wrapper Induction Environment*. AAAI Technical Report WS-98-10, 1998. 15

[30] CALIFF, MARY ELAINE, RAYMOND J. MOONEY and DAVID COHN: *Bottom-Up Relational Learning of Pattern Matching Rules for Information Extraction*. pages 328–334, 2003. 16

[31] SODERLAND, STEPHEN: *Learning Information Extraction Rules for Semi-Structured and Free Text*. Mach. Learn., 34(1-3):233–272, February 1999. 16

[32] HSU, CHUN NAN and MING-TZUNG DUNG: *Generating Finite-State Transducers For Semi-Structured Data Extraction From The Web*, 1998. 17

[33] BERNERS-LEE, TIM, J HANDLER and ORA LASSILA: *The semantic web*. Database and Network journal, 36(3):7, 2006. 17

[34] SCHEMA.ORG: *An initiative launched by Google, Microsoft and Yahoo*. `http://Schema.org`, visited 2015-08-16. 17

[35] GUHA, RAMANATHAN V.: *Light at the End of the Tunnel*. 2013. 19, 22

[36] WIKIPEDIA: *The Ultimate Problem of RDF and the Semantic Web*. `http://milicicvuk.com/blog/2011/07/19/ultimate-problem-of-rdf-and-semantic-web/`, visited 2015-9-22. 19

[37] WWW.TECHNOLOGYREVIEW.COM: *Google, Microsoft, and Yahoo Team Up to Advance Semantic Web*. `http://www.technologyreview.com/news/424259/google-microsoft-and-yahoo-team-up-to-advance-semantic-web/`, visited 2015-08-16. 21

[38] SCHEMA.ORG: *Getting started with schema.org using Microdata*. `http://www.schema.org/docs/gs.html`, visited 2015-08-06. 27

[39] BARTOLI, ALBERTO, ANDREA DE LORENZO, ERIC MEDVET and FABIANO TARLAO: *Playing Regex Golf with Genetic Programming*. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 1063–1070, New York, NY, USA, 2014. ACM. 31

[40] HTTP://BINARYPARK.ORG: *List of websites using Schema.org*. `http://getschema.org/index.php/List_of_websites_using_Schema.org/`, visited 2015-8-2. 41

[41] DILL, STEPHEN, NADAV EIRON, DAVID GIBSON, DANIEL GRUHL, R. GUHA, ANANT JHINGRAN, TAPAS KANUNGO, SRIDHAR RAJAGOPALAN, ANDREW TOMKINS, JOHN A. TOMLIN and JASON Y. ZIEN: *SemTag and Seeker: Bootstrapping the Semantic Web via Automated Semantic Annotation*. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 178–186, New York, NY, USA, 2003. IBM, ACM. 50

[42] WIKIPEDIA: *DARPA*. `http://de.wikipedia.org/wiki/DARPA_Agent_Markup_Language`, visited 2015-07-13. 50

[43] WIKIPEDIA: *Inductive Logic Programming*. http://en.wikipedia.org/wiki/Inductive_logic_programming, visited 2015-06-28. 50

[44] WIKIPEDIA: *Information Extraction*. http://en.wikipedia.org/wiki/Information_extraction, visited 2015-06-15. 50

[45] WIKIPEDIA: *Information Retrieval*. http://en.wikipedia.org/wiki/Information_retrieval, visited 2015-07-03. 50

[46] WIKIPEDIA: *Machine Learning*. http://en.wikipedia.org/wiki/Machine_learning, visited 2015-06-27. 51

[47] WIKIPEDIA: *Natural Language Processing (NLP)*. http://en.wikipedia.org/wiki/Natural_language_processing, visited 2015-06-27. 51

[48] WIKIPEDIA: *Ontology inference Layer*. http://en.wikipedia.org/wiki/Ontology_Inference_Layer, visited 2015-08-05. 51

[49] WIKIPEDIA: *Open Knowledge Base Connectivity*. http://en.wikipedia.org/wiki/Open_Knowledge_Base_Connectivity, visited 2015-08-05. 51

[50] WIKIPEDIA: *Out of the box*. https://en.wikipedia.org/wiki/Out_of_the_box_feature, visited 2015-10-22. 51

[51] WIKIPEDIA: *RDF Schema*. http://en.wikipedia.org/wiki/RDF_Schema, visited 2015-05-30. 52

[52] W3: *Resource Description Framework (RDF)*. http://www.w3.org/RDF/, visited 2015-05-29. 52

[53] WIKIPEDIA: *The Semantic Web*. http://en.wikipedia.org/wiki/Semantic_Web, visited 2015-06-12. 52

[54] WIKIPEDIA: *SPARQL (sparkle)*. http://en.wikipedia.org/wiki/SPARQL, visited 2015-06-11. 52

[55] WIKIPEDIA: *Supervised Learning*. http://en.wikipedia.org/wiki/Supervised_learning, visited 2015-06-14. 52

[56] WIKIPEDIA: *Unsupervised Learning*. http://en.wikipedia.org/wiki/Unsupervised_learning, visited 2015-06-15. 53

[57] WIKIPEDIA: *Web Scraping*. http://en.wikipedia.org/wiki/Web_scraping, visited 2015-07-10. 53

[58] LAENDER, ALBERTO H. F., BERTHIER A. RIBEIRO-NETO, ALTIGRAN S. DA SILVA and JULIANA S. TEIXEIRA: *A Brief Survey of Web Data Extraction Tools*. SIGMOD Rec., 31(2):84–93, June 2002. 53

[59] FERRARA, EMILIO, PASQUALE DE MEO, GIACOMO FIUMARA and ROBERT BAUMGARTNER: *Web Data Extraction, Applications and Techniques: A Survey*. CoRR, abs/1207.0246, 2012. 53

[60] WIKIPEDIA: *Data Mining*. http://en.wikipedia.org/wiki/Wrapper_(data_mining), visited 2015-06-11. 54

List of Figures

List of Tables

Anhang

Hier befinden sich der komplette Quelltext des entwickelten programms.

A Anhang: Quelltexte

Annotaion.js

```
// ==UserScript==
// @name         Semantic annotation aggregation Browser
// @namespace    http://your.homepage/
// @version      1.0
// @description  Scan webpages for semantic structure as provided by Schema.org, interconnect found semantic information, and make them available through webbrowser or javascript variables
// @author       Hannes Rammer
// @match
// @grant        none
// ==/UserScript==
var analyzer = {
    timesTaken: "",
    percent: 50,
    itemScopesToContainedItemScopes: {},
    withIncludedProperties: true,
    specialIdList: [],
    JSONRootItemScopeList: [],
    HTMLRootItemScopeList: [],
    itemScopeIdList: [],
    jsonHash: {},
    possibleItemScopeRelationHash: [],
    possibleItemScopeRelationHashWithT: [],
    directCrossConnectionToItemScopeHash: {},
    directItemScopeToCrossConnectionHash: {},
    directCrossConnectionToItemScopeHashWithT: {},
    directItemScopeToCrossConnectionHashWithT: {},
    propertyScopeLength: 0,
    extract: {
        /**
         * create a json list from all found root scope items
         * @effect adds json scope item to analyzer.JSONRootItemScopeList
         */
```

```
32          JSONRootItemScopesFromWebpage: function (scopeList) {
33              //var HTMLRootItemScopes = analyzer.extract.helper.
                    HTMLRootItemScopesFromWebpage();
34              var HTMLRootItemScopes = scopeList;//analyzer.extract.helper.
                    HTMLRootItemScopesFromWebpage();
35              for (var i = 0; i < HTMLRootItemScopes.length; i++) {
36                  var HTMLRootItemScope = HTMLRootItemScopes[i];
37                  var jsonJSTreeJsonScope = analyzer.transformHTMLItemScopeToJSON(
                        HTMLRootItemScope, analyzer.itemScopeIdList.length);
38                  analyzer.JSONRootItemScopeList.push(jsonJSTreeJsonScope);
39              }
40          },
41          helper: {
42              HTMLRootItemScopesFromWebpage: function () {
43                  analyzer.HTMLRootItemScopeList = [];
44                  analyzer.extract.helper.findHTMLRootItemScopes(document.body);
45                  return analyzer.HTMLRootItemScopeList;
46              },
47              findHTMLRootItemScopes: function (item) {
48                  var isScope = item.hasAttribute('itemscope');
49                  if (isScope) {
50                      analyzer.HTMLRootItemScopeList.push(item);
51                  } else {
52                      var children = item.children;
53                      var hasChildren = children.length > 0;
54                      if (hasChildren) {
55                          for (var i = 0; i < children.length; i++) {
56                              var newItem = item.children[i];
57                              analyzer.extract.helper.findHTMLRootItemScopes(newItem);
58                          }
59                      }
60                  }
61              }
62          }
63      },
64
65      /**
66       * compares single scope items with all scope Items in List to find possible
             connections
67       * only comparing items from different http sources
68       * @param currentItem
69       * @prerequisite same itemSope, different nodeSource and at least 1 identical
             attribute
70       * @effect adds possible connection id string eg. '0_1' to analyzer.
             possibleItemScopeRelationList list  TODO UPDATE
71       */
72      createPossibleConnectionsBetweenScopeItemAndList: function (currentItem) {
73          //for (var i = 0; i < analyzer.jsonList.length; i++) {
74          Object.keys(analyzer.jsonHash).forEach(function (storageItemKey) {
75              var storageItem = analyzer.jsonHash[storageItemKey];
76              //if (currentItem['nodeSource'] !== storageItem['nodeSource']) {
77              if (currentItem['scopeId'] !== storageItem['scopeId']) {
78                  if (currentItem['nodeType'] === storageItem['nodeType']) {
79                      //when same name -> check fore more similarities
80                      var cNodeName = currentItem['nodeName'];
```

```
81                      var sNodeName = storageItem['nodeName'];
82                      var emptyName = cNodeName === "" || sNodeName === "";
83                      var sameName = cNodeName === sNodeName;
84
85                      var isPlural = helper.isPlural(cNodeName, sNodeName);
86                      var createConnection = (emptyName || sameName);//&& isPlural;
87                      //do not create relations for plurals eg connecting all actors of
                            an actor list to the list and therefore to each other
88
89                      if (createConnection) {
90                          //when same value -> check fore more similarities
91                          var cleanCurrentString = helper.toLowerCaseTrimmedSpaces(
                                currentItem['nodeValue']);
92                          var cleanStorageString = helper.toLowerCaseTrimmedSpaces(
                                storageItem['nodeValue']);
93                          if (cleanCurrentString === cleanStorageString) {
94                              analyzer.createDirectConnection(currentItem, storageItem)
                                    ;
95                          }
96                      }
97                  }
98
99              }
100
101         });
102     },
103
104     createDirectConnection: function (currentItem, storageItem) {
105         if ((currentItem.children !== undefined) && ( storageItem.children !==
                undefined)) {
106             for (var k = 0; k < currentItem.children.length; k++) {
107                 var currentItemChild = currentItem.children[k];
108                 var currentItemChildNodeName = currentItemChild['nodeName'];
109                 for (var l = 0; l < storageItem.children.length; l++) {
110                     var storageItemChild = storageItem.children[l];
111                     if (storageItemChild !== undefined) {
112                         //if (currentItemChildNodeName !== "thumbnailUrl" &&
                                currentItemChildNodeName !== "url" &&
                                currentItemChildNodeName !== "image" &&
                                currentItemChildNodeName !== "provider") {
113                         if (currentItemChildNodeName !== "thumbnailUrl" &&
                                currentItemChildNodeName !== "image" &&
                                currentItemChildNodeName !== "provider") {
114                             if (currentItemChildNodeName === storageItemChild['
                                    nodeName']) {
115                                 var cleanCurrentString = helper.
                                        toLowerCaseTrimmedSpaces(currentItemChild['
                                        nodeValue']);
116                                 var cleanStorageString = helper.
                                        toLowerCaseTrimmedSpaces(storageItemChild['
                                        nodeValue']);
117                                 if (cleanCurrentString !== "" && cleanCurrentString
                                        === cleanStorageString) {
118                                     //Sort numbers in an array in ascending order
```

```
119                 var sortedScopeConnectionIds = helper.
                        sortAscending([currentItem.scopeId,
                        storageItem.scopeId]);
120                 var propertyConnectionIds = [];
121                 if (sortedScopeConnectionIds[0] === currentItem.
                        scopeId) {
122                     propertyConnectionIds = [currentItemChild['
                            propertyId'], storageItemChild['propertyId
                            ']];
123                 } else {
124                     propertyConnectionIds = [storageItemChild['
                            propertyId'], currentItemChild['propertyId
                            ']];
125                 }
126                 var joinedScopeConnectionId =
                        sortedScopeConnectionIds.join('-');
127                 var joinedPropertyConnectionId =
                        propertyConnectionIds.join('-');
128                 var currentCrossId = -1;
129                 var newCrossId = Object.keys(analyzer.
                        directItemScopeToCrossConnectionHash).length;
130                 var hasCurrentItemScopeId = analyzer.
                        directItemScopeToCrossConnectionHash.
                        hasOwnProperty(currentItem.scopeId);
131                 var hasStorageItemScopeId = analyzer.
                        directItemScopeToCrossConnectionHash.
                        hasOwnProperty(storageItem.scopeId);
132                 //find existing cross id
133                 if (hasCurrentItemScopeId) {
134                     currentCrossId = analyzer.
                            directItemScopeToCrossConnectionHash[
                            currentItem.scopeId];
135                 } else {
136                     if (hasStorageItemScopeId) {
137                         currentCrossId = analyzer.
                                directItemScopeToCrossConnectionHash[
                                storageItem.scopeId];
138                     }
139                 }
140                 if (currentCrossId === -1) {
141                     currentCrossId = newCrossId;
142                 }
143                 analyzer.directItemScopeToCrossConnectionHash[
                        currentItem.scopeId] = currentCrossId;
144                 analyzer.directItemScopeToCrossConnectionHash[
                        storageItem.scopeId] = currentCrossId;
145                 var hasCurrentCrossConnectionId = analyzer.
                        directCrossConnectionToItemScopeHash.
                        hasOwnProperty(currentCrossId.toString());
146                 var itemScopeIds = analyzer.
                        directCrossConnectionToItemScopeHash[
                        currentCrossId];
147                 if (hasCurrentCrossConnectionId) {
148                     if (itemScopeIds.indexOf(currentItem.scopeId)
                            < 0) {
```

```
149                         itemScopeIds.push(currentItem.scopeId);
150                     }
151                     if (itemScopeIds.indexOf(storageItem.scopeId)
                            < 0) {
152                         itemScopeIds.push(storageItem.scopeId);
153                     }
154                 } else {
155                     itemScopeIds = [currentItem.scopeId,
                        storageItem.scopeId];
156                 }
157                 analyzer.directCrossConnectionToItemScopeHash[
                    currentCrossId] = helper.sortAscending(
                    itemScopeIds);
158                 if (analyzer.possibleItemScopeRelationHash.
                    hasOwnProperty(joinedScopeConnectionId)) {
159                     var possiblePropertyRelationList = analyzer.
                        possibleItemScopeRelationHash[
                        joinedScopeConnectionId];
160                     if (possiblePropertyRelationList.indexOf(
                        joinedPropertyConnectionId) < 0) {
161                         possiblePropertyRelationList.push(
                            joinedPropertyConnectionId);
162                     }
163                 } else {
164                     analyzer.possibleItemScopeRelationHash[
                        joinedScopeConnectionId] = [
                        joinedPropertyConnectionId];
165                 }
166                 ////////With Threshold
167                 var percents = helper.percents(currentItem,
                    storageItem);
168                 if (percents[0] >= analyzer.percent || percents
                    [1] >= analyzer.percent) {
169                     var currentCrossIdWithT = -1;
170                     var newCrossIdWithT = Object.keys(analyzer.
                        directItemScopeToCrossConnectionHashWithT)
                        .length;
171                     var hasCurrentItemScopeIdWithT = analyzer.
                        directItemScopeToCrossConnectionHashWithT.
                        hasOwnProperty(currentItem.scopeId);
172                     var hasStorageItemScopeIdWithT = analyzer.
                        directItemScopeToCrossConnectionHashWithT.
                        hasOwnProperty(storageItem.scopeId);
173                     //find existing cross id
174                     if (hasCurrentItemScopeIdWithT) {
175                         currentCrossIdWithT = analyzer.
                            directItemScopeToCrossConnectionHashWithT
                            [currentItem.scopeId];
176                     } else {
177                         if (hasStorageItemScopeIdWithT) {
178                             currentCrossIdWithT = analyzer.
                                directItemScopeToCrossConnectionHashWithT
                                [storageItem.scopeId];
179                         }
180                     }
```

```
181                                 if (currentCrossIdWithT === -1) {
182                                     currentCrossIdWithT = newCrossIdWithT;
183                                 }
184                                 analyzer.
                                        directItemScopeToCrossConnectionHashWithT[
                                        currentItem.scopeId] = currentCrossIdWithT
                                        ;
185                                 analyzer.
                                        directItemScopeToCrossConnectionHashWithT[
                                        storageItem.scopeId] = currentCrossIdWithT
                                        ;
186                                 var hasCurrentCrossConnectionIdWithT =
                                        analyzer.
                                        directCrossConnectionToItemScopeHashWithT.
                                        hasOwnProperty(currentCrossIdWithT.
                                        toString());
187                                 var itemScopeIdsWithT = analyzer.
                                        directCrossConnectionToItemScopeHashWithT[
                                        currentCrossIdWithT];
188                                 if (hasCurrentCrossConnectionIdWithT) {
189                                     if (itemScopeIdsWithT.indexOf(currentItem
                                            .scopeId) < 0) {
190                                         itemScopeIdsWithT.push(currentItem.
                                                scopeId);
191                                     }
192                                     if (itemScopeIdsWithT.indexOf(storageItem
                                            .scopeId) < 0) {
193                                         itemScopeIdsWithT.push(storageItem.
                                                scopeId);
194                                     }
195                                 } else {
196                                     itemScopeIdsWithT = [currentItem.scopeId,
                                             storageItem.scopeId];
197                                 }
198                                 analyzer.
                                        directCrossConnectionToItemScopeHashWithT[
                                        currentCrossIdWithT] = helper.
                                        sortAscending(itemScopeIdsWithT);
199                                 if (analyzer.
                                        possibleItemScopeRelationHashWithT.
                                        hasOwnProperty(joinedScopeConnectionId)) {
200                                     var possiblePropertyRelationListWithT =
                                            analyzer.
                                            possibleItemScopeRelationHashWithT[
                                            joinedScopeConnectionId];
201                                     if (possiblePropertyRelationListWithT.
                                            indexOf(joinedPropertyConnectionId) <
                                            0) {
202                                         possiblePropertyRelationListWithT.
                                                push(joinedPropertyConnectionId);
203                                     }
204                                 } else {
205                                     analyzer.
                                            possibleItemScopeRelationHashWithT[
```

```
                                                    joinedScopeConnectionId] = [
                                                    joinedPropertyConnectionId];
206                                             }
207                                         }
208                                     }
209                                 }
210                             }
211                         }
212                     }
213                 }
214             }
215         },
216         mapAllForSemanticReference: function () {
217             //for (var i = 0; i < analyzer.jsonList.length; i++) {
218             var currentItemCount = 0;
219             Object.keys(analyzer.jsonHash).forEach(function (currentItemKey) {
220                 var currentItem = analyzer.jsonHash[currentItemKey];
221                 analyzer.createPossibleConnectionsBetweenScopeItemAndList(currentItem);
222                 currentItemCount += 1;
223             });
224         },
225
226         transformHTMLItemScopeToJSON: function (itemScope, scopeId) {
227             var jsonScope = analyzer.newJSONTreeNode();
228             if (analyzer.itemScopeIdList.indexOf(scopeId) > -1) {
229                 console.log("duplicate Scope Id -> something went wrong");
230             }
231             jsonScope.scopeId = scopeId;
232             jsonScope.nodeSource = helper.cleanUrl(document.location.toString());
233             jsonScope.nodeType = itemScope.getAttribute("itemtype");
234             if (itemScope.getAttribute("itemprop") !== null) {
235                 jsonScope.nodeName = itemScope.getAttribute("itemprop");
236             }
237             var HTMLItemScopeProperties = analyzer.getItemScopeProperties(itemScope);
238             for (var i = 0; i < HTMLItemScopeProperties.length; i++) {
239                 var HTMLItemScopeProperty = HTMLItemScopeProperties[i];
240                 var spcecialid = HTMLItemScopeProperty.getAttribute("specialid");
241                 var index = analyzer.specialIdList.indexOf(parseInt(spcecialid));
242                 if (analyzer.withIncludedProperties || (index >= 0)) {
243                     analyzer.specialIdList.splice(index, 1);
244                     //if property is new scope
245                     if (HTMLItemScopeProperty.getAttribute("itemScope") !== null) {
246                         analyzer.propertyScopeLength += 1;
247                         var propertyScope = analyzer.transformHTMLItemScopeToJSON(
                                HTMLItemScopeProperty, analyzer.itemScopeIdList.length +
                                analyzer.propertyScopeLength);
248                         propertyScope.propertyId = i;
249                         //TODO see if needed
250                         jsonScope.children.push(propertyScope);
251                     } else {
252                         var propertyName = HTMLItemScopeProperty.getAttribute("itemprop")
                                .toString();
253                         var propertyValue = "";
254                         var valueType = "TEXT";
255                         if (HTMLItemScopeProperty.tagName === "IMG") {
```

```
256                         propertyValue = "#";//HTMLItemScopeProperty.src;
257                         valueType = "IMG";
258                     } else if (HTMLItemScopeProperty.tagName === "A") {
259                         propertyValue = HTMLItemScopeProperty.href;
260                         valueType = "A";
261                     } else {
262                         if (typeof HTMLItemScopeProperty.textContent !== "undefined")
                                {
263                             propertyValue = HTMLItemScopeProperty.textContent;
264                         } else {
265                             propertyValue = HTMLItemScopeProperty.innerText;
266                         }
267                     }
268                     var jsonProp = analyzer.newJSONTreeNode();
269                     jsonProp.propertyId = i;
270                     jsonProp.nodeName = propertyName;
271                     propertyValue = helper.toLowerCaseTrimmedSpaces(propertyValue);
272 
273                     if (propertyValue.indexOf("http://") > -1 || propertyValue.
                           indexOf("https://") > -1) {
274                         propertyValue = helper.cleanUrl(propertyValue);
275                     }
276                     jsonProp.nodeValue = propertyValue;
277                     jsonProp.valueType = valueType;
278                     jsonScope.children.push(jsonProp);
279                 }
280             }
281         }
282         if (helper.objectDoesNotExist(jsonScope)) {
283             var scopeId = jsonScope.scopeId;
284             if (analyzer.jsonHash.hasOwnProperty(scopeId)) {
285             } else {
286                 analyzer.jsonHash[scopeId] = jsonScope;
287             }
288             if (analyzer.itemScopeIdList.indexOf(scopeId) == -1) {
289                 analyzer.itemScopeIdList.push(scopeId);
290             }
291             for (var j = 0; j < jsonScope.children.length; j++) {
292                 var jsonScopeProperty = jsonScope.children[j];
293                 if (jsonScopeProperty.scopeId !== undefined) {
294                     var propertyScopeId = jsonScopeProperty.scopeId;
295                     if (analyzer.itemScopeIdList.indexOf(propertyScopeId) == -1) {
296                         analyzer.itemScopeIdList.push(propertyScopeId);
297                     }
298                     if (analyzer.itemScopesToContainedItemScopes.hasOwnProperty(
                            scopeId)) {
299                         var relationList = analyzer.itemScopesToContainedItemScopes[
                                scopeId];
300                         if (relationList.indexOf(propertyScopeId) < 0) {
301                             relationList.push(propertyScopeId);
302                         }
303                     } else {
304                         analyzer.itemScopesToContainedItemScopes[scopeId] = [
                                propertyScopeId];
305                     }
```

```
306                     if (analyzer.itemScopesToContainedItemScopes.hasOwnProperty(
                            propertyScopeId)) {
307                         var relationList = analyzer.itemScopesToContainedItemScopes[
                                propertyScopeId];
308                         if (relationList.indexOf(scopeId) < 0) {
309                             relationList.push(scopeId);
310                         }
311                     } else {
312                         analyzer.itemScopesToContainedItemScopes[propertyScopeId] = [
                                scopeId];
313                     }
314                     if (analyzer.jsonHash.hasOwnProperty(propertyScopeId)) {
315
316                     } else {
317                         analyzer.jsonHash[propertyScopeId] = jsonScopeProperty;
318                     }
319                 }
320             }
321             analyzer.propertyScopeLength = 0;
322         }
323         return jsonScope;
324     },
325     getItemScopeProperties: function (itemScope) {
326         var specialId = 0;
327
328         var allIncludingDuplicates = itemScope.querySelectorAll('[itemprop]');
329         for (var i = 0; i < allIncludingDuplicates.length; i++) {
330             allIncludingDuplicates[i].setAttribute("specialId", specialId.toString())
                    ;
331             analyzer.specialIdList.push(specialId);
332             specialId += 1;
333         }
334         return allIncludingDuplicates;
335     },
336
337     newJSONTreeNode: function () {
338         return {
339             //id          : "string" // will be autogenerated if omitted
340             nodeType: "", // node text
341             nodeName: "",
342             nodeValue: "",
343             nodeSource: "",
344             scopeId: undefined,
345             propertyId: undefined,
346             children: []//,  // array of strings or objects
347         };
348     }
349
350 };
351
352 var visual = {
353     setListStyle: function (htmlElement) {
354         htmlElement.style.fontFamily = "monospace";
355         htmlElement.style.margin = 0;
356     },
```

```
357     createHTMLFromJSONScope: function (jsonScope) {
358         if (jsonScope === undefined) {
359             console.log("something went wrong: undefined jsonSope");
360         }
361         var ul = document.createElement("ul");
362         var ulType = document.createElement("ul");
363         var liType = document.createElement("li");
364         var liName = document.createElement("li");
365         var liValue = document.createElement("li");
366         ul.style.margin = "0";
367         ul.style.padding = "5px 15px";
368         liType.style.display = "none";
369         liType.className = "liType";
370         liName.className = "liName";
371         liValue.className = "liValue";
372         visual.setListStyle(liType);
373         visual.setListStyle(liName);
374         visual.setListStyle(liValue);
375         var nodeName = jsonScope.nodeName;
376         if (nodeName === "") {
377             var splitList = jsonScope.nodeType.split("/");
378             nodeName = splitList[splitList.length - 1];
379         } else {
380
381         }
382         nodeName = "ID:" + jsonScope.scopeId + " " + nodeName;
383         var htmlContent = document.createElement("a");
384         var browserLink = document.createElement("a");
385         htmlContent.style.cursor = "pointer";
386         browserLink.style.cursor = "pointer";
387         //htmlContent.title = "ItemScopeId:" + jsonScope.scopeId;
388         htmlContent.title = JSON.stringify(jsonScope);
389         htmlContent.style.color = "#a9014b";
390         htmlContent.style.fontWeight = "bold";
391         if (typeof liType.textContent !== "undefined") {
392             htmlContent.textContent = nodeName;
393             browserLink.textContent = "->";
394             liType.textContent = jsonScope.nodeType;
395             liValue.textContent = jsonScope.nodeValue;
396         } else {
397             htmlContent.innerText = nodeName;
398             browserLink.innerText = "->";
399             liType.innerText = jsonScope.nodeType;
400             liValue.innerText = jsonScope.nodeValue;
401         }
402         liName.appendChild(htmlContent);
403         liName.appendChild(browserLink);
404         htmlContent.onclick = function () {
405             visual.toggleItemScopeView(liType);
406         };
407         ulType.appendChild(liType);
408         liName.appendChild(ulType);
409         ul.appendChild(liName);
410         browserLink.onclick = function () {
411             console.time('browser');
```

```
412
413             visual.render.relationsForItemScope(jsonScope);
414             console.timeEnd('browser');
415
416         };
417         var properties = jsonScope.children;
418         var newScope = true;
419         for (var i = 0; i < properties.length; i++) {
420             var property = properties[i];
421             if (property.nodeName === "name" && newScope) {
422                 if (nodeName.indexOf("actors") === -1) {
423                     if (typeof browserLink.textContent !== "undefined") {
424                         browserLink.textContent += property.nodeValue;
425                     } else {
426                         browserLink.innerText += property.nodeValue;
427                     }
428                 }
429                 newScope = false;
430             }
431             //if property is new scope
432             if (property.nodeType !== "") {
433                 var htmlPropertyScope = visual.createHTMLFromJSONScope(property);
434                 liType.appendChild(htmlPropertyScope);
435                 //console.log(htmlPropertyScope);
436             } else {
437                 var ulPropName = document.createElement("ul");
438                 var liPropName = document.createElement("li");
439                 visual.setListStyle(liPropName);
440                 var ulPropValue = document.createElement("ul");
441                 if (typeof liPropName.textContent !== "undefined") {
442                     liPropName.textContent = property.nodeName;
443                 } else {
444                     liPropName.innerText = property.nodeName;
445                 }
446                 if (property.valueType === "IMG") {
447                     var img = document.createElement("IMG");
448                     img.src = "#";//property.nodeValue;
449                     //liPropValue.appendChild(img);
450                     liPropName.appendChild(img);
451                 } else if (property.valueType === "A") {
452                     var a = document.createElement("a");
453                     a.href = property.nodeValue;
454                     if (typeof a.textContent !== "undefined") {
455                         a.textContent = property.nodeValue;
456                     } else {
457                         a.innerText = property.nodeValue;
458                     }
459                     liPropName.appendChild(a);
460                 } else {
461                     if (typeof liPropName.textContent !== "undefined") {
462                         liPropName.textContent = liPropName.textContent + "␣:␣" + ❯
                            property.nodeValue;
463                     } else {
464                         liPropName.innerText = liPropName.innerText + "␣:␣" + ❯
                            property.nodeValue;
```

```
465                     }
466                 }
467                 ulPropName.appendChild(liPropName);
468                 ulPropName.appendChild(ulPropValue);
469                 liType.appendChild(ulPropName);
470             }
471         }
472         return ul;
473     },
474
475     /**
476      * function used on scope Elements liType div onclick function to toggle display properties
477      * @param li
478      */
479     toggleItemScopeView: function (li) {
480         if (li.className === "liType") {
481             if (li.style.display === "none") {
482                 li.style.display = "block";
483             } else {
484                 li.style.display = "none";
485             }
486         }
487
488     },
489     render: {
490         currentWebPageItems: function () {
491             var webPageBox = visual.getDisplay("WebPage");
492             for (var i = 0; i < analyzer.JSONRootItemScopeList.length; i++) {
493                 var item = analyzer.JSONRootItemScopeList[i];
494                 var htmlUl = visual.createHTMLFromJSONScope(item);
495                 webPageBox.appendChild(htmlUl);
496             }
497         },
498         directConnections: function () {
499             var directConnectionBox = visual.getDisplay("DirectConnection");
500             Object.keys(analyzer.possibleItemScopeRelationHash).forEach(function (possibleJoinedScopeRelationId) {
501                 var possibleJoinedScopeRelationIds = possibleJoinedScopeRelationId.split('-');
502                 var possiblePropertyRelationList = analyzer.possibleItemScopeRelationHash[possibleJoinedScopeRelationId];
503                 var possiblePropertyRelationListLength = possiblePropertyRelationList.length;
504                 for (var i = 0; i < possiblePropertyRelationListLength; i++) {
505                     var possiblePropertyRelationIds = possiblePropertyRelationList[i].split('-');
506                     var itemScopePropertyConnectionReason = document.createElement("div");
507                     var JSONItemScope1 = analyzer.jsonHash[possibleJoinedScopeRelationIds[0]];
508                     var JSONItemScope2 = analyzer.jsonHash[possibleJoinedScopeRelationIds[1]];
509                     var children1 = JSONItemScope1.children;
510                     var children1Length = children1.length;
```

```
511                     for (var j = 0; j < children1Length; j++) {
512                         var property = children1[j];
513                         if (property['propertyId'] === parseInt(
                                possiblePropertyRelationIds[0])) {
514                             if (typeof itemScopePropertyConnectionReason.textContent
                                    !== "undefined") {
515                                 itemScopePropertyConnectionReason.textContent =
                                        property.nodeName + " : " + property.nodeValue;
516                             } else {
517                                 itemScopePropertyConnectionReason.innerText =
                                        property.nodeName + " : " + property.nodeValue;
518                             }
519                         }
520                     }
521                     var box = document.createElement("div");
522                     box.style.border = "1px solid black";
523                     box.style.maxHeight = "450px";
524                     box.style.width = "100%";
525                     box.style.overflow = "auto";
526                     var htmlUl1 = visual.createHTMLFromJSONScope(JSONItemScope1);
527                     var htmlUl2 = visual.createHTMLFromJSONScope(JSONItemScope2);
528                     box.appendChild(itemScopePropertyConnectionReason);
529                     box.appendChild(htmlUl1);
530                     box.appendChild(htmlUl2);
531                     directConnectionBox.appendChild(box);
532                 }
533             });
534         },
535         directCrossConnection: function () {
536             var directCrossConnectionBox = visual.getDisplay("DirectCrossConnection")
                    ;
537             Object.keys(analyzer.directCrossConnectionToItemScopeHash).forEach(
                    function (directCrossConnectionId) {
538                 var box = document.createElement("div");
539                 box.style.border = "1px solid black";
540                 box.style.maxHeight = "450px";
541                 box.style.width = "100%";
542                 box.style.overflow = "auto";
543                 var itemScopeIds = analyzer.directCrossConnectionToItemScopeHash[
                        directCrossConnectionId];
544                 var itemScopeIdsDublicate = JSON.parse(JSON.stringify(itemScopeIds));
545                 for (var i = 0; i < itemScopeIdsDublicate.length;
                        itemScopeIdsDublicate.splice(0, 1)) {
546                     var firstItemScopeId = itemScopeIdsDublicate[i];
547                     var firstItemScopeCount = 0;
548                     for (var j = i + 1; j < itemScopeIdsDublicate.length; j++) {
549                         var secondItemScopeId = itemScopeIdsDublicate[j];
550                         var sorted = helper.sortAscending([firstItemScopeId,
                                secondItemScopeId]);
551                         var connectionString = sorted[0] + "-" + sorted[1];
552                         //console.log(connectionString);
553                         if (analyzer.possibleItemScopeRelationHash.hasOwnProperty(
                                connectionString)) {
554                             var propertyConnections = analyzer.
                                    possibleItemScopeRelationHash[connectionString];
```

```
555                             var connectionTable = document.createElement("table");
556                             var connectionTR = document.createElement("tr");
557                             var firstItemScopeTD = document.createElement("td");
558                             var connectionTD = document.createElement("td");
559                             var secondItemScopeTD = document.createElement("td");
560                             firstItemScopeTD.width = "37%";
561                             connectionTD.width = "26%";
562                             secondItemScopeTD.width = "37%";
563                             connectionTable.style.width = "100%";
564                             connectionTable.style.border = "1px␣dotted␣black";
565                             connectionTR.appendChild(connectionTR);
566                             connectionTR.appendChild(firstItemScopeTD);
567                             connectionTR.appendChild(connectionTD);
568                             connectionTR.appendChild(secondItemScopeTD);
569                             var div1 = document.createElement("div");
570                             var div2 = document.createElement("div");
571                             div1.style.overflow = "auto";
572                             div2.style.overflow = "auto";
573                             //div1.style.width = "400px";
574                             //div2.style.width = "400px";
575                             div1.style.maxHeight = "200px";
576                             div2.style.maxHeight = "200px";
577                             var firstItem = analyzer.jsonHash[firstItemScopeId];
578                             var secondItem = analyzer.jsonHash[secondItemScopeId];
579                             var firstItemScopeUL = visual.createHTMLFromJSONScope(
                                    firstItem);
580                             var secondItemScopeUL = visual.createHTMLFromJSONScope(
                                    secondItem);
581                             div1.appendChild(firstItemScopeUL);
582                             div2.appendChild(secondItemScopeUL);
583                             var propertyTitleList = "";
584                             var countedPropertyCount = 0;
585                             for (var k = 0; k < propertyConnections.length; k++) {
586                                 var propertyConnectionId = propertyConnections[k].
                                        split("-")[0];
587                                 var propertyObject = analyzer.jsonHash[
                                        firstItemScopeId].children[propertyConnectionId];
588                                 var propertyObjectNodeName = propertyObject['nodeName
                                        '];
589                                 //if (propertyObjectNodeName !== "thumbnailUrl" &&
                                        propertyObjectNodeName !== "url" &&
                                        propertyObjectNodeName !== "image" &&
                                        propertyObjectNodeName !== "provider") {
590                                 if (propertyObjectNodeName !== "thumbnailUrl" &&
                                        propertyObjectNodeName !== "image" &&
                                        propertyObjectNodeName !== "provider") {
591                                     countedPropertyCount++;
592                                     propertyTitleList += propertyObject.nodeName + "␣
                                            :␣" + propertyObject.nodeValue + "\n";
593                                 }
594                             }
595                             var perc1 = ((100 / firstItem.children.length ) *
                                    countedPropertyCount).toFixed(2);
596                             var perc2 = ((100 / secondItem.children.length ) *
                                    countedPropertyCount).toFixed(2);
```

```
597                             var connectionLink = document.createElement("a");
598                             var connectionLinkText = "<b>" + perc1 + "%</b>␣" +
                                    countedPropertyCount + "connection";
599                             if (propertyConnections.length !== 1) {
600                                 connectionLinkText += "s";
601                             }
602                             connectionLinkText += "␣<b>" + perc2 + "%</b>";
603                             connectionLink.innerHTML = connectionLinkText;
604                             connectionLink.title = propertyTitleList;
605                             if (firstItemScopeCount === 0) {
606                                 firstItemScopeTD.appendChild(div1);
607                             }
608                             firstItemScopeCount++;
609                             connectionTD.appendChild(connectionLink);
610                             secondItemScopeTD.appendChild(div2);
611                             box.appendChild(connectionTable);
612                         }
613                     }
614                 }
615                 directCrossConnectionBox.appendChild(box);
616             });
617         },
618         relationsForItemScope: function (itemScope) {
619             var browserBox = visual.getDisplay("Browser");
620             var listManager = [];
621             browserBox.innerHTML = "";
622             helper.toggleTab("Browser");
623             var itemScopeId = itemScope.scopeId;
624             //render direct connection
625             visual.render.renderBox(browserBox, itemScopeId, "direct_connection",
                    100, listManager);
626             //render cross connection
627             var hasCrossConnection = analyzer.
                    directItemScopeToCrossConnectionHashWithT.hasOwnProperty(itemScopeId);
628             if (hasCrossConnection) {
629                 var crossConnectionId = analyzer.
                        directItemScopeToCrossConnectionHashWithT[itemScopeId];
630                 var itemScopeIdList = analyzer.
                        directCrossConnectionToItemScopeHashWithT[crossConnectionId];
631                 var listLength = itemScopeIdList.length;
632                 for (var i = 0; i < listLength; i++) {
633                     var percents = helper.percents(itemScope, analyzer.jsonHash[
                            itemScopeIdList[i]]);
634                     if (percents[0] >= analyzer.percent || percents[1] >= analyzer.
                            percent) {
635                         visual.render.renderBox(browserBox, itemScopeIdList[i], "
                                cross_connection", percents[1], listManager);
636                     }
637                 }
638             }
639         },
640         renderBox: function (browserBox, itemScopeId, name, percent, listManager) {
641             var innerBox = document.createElement("div");
642             innerBox.style.border = "1px␣dotted␣black";
643             var outerBox = document.createElement("div");
```

```
644             outerBox.style.border = "1px␣solid␣black";
645             var itemScope = analyzer.jsonHash[itemScopeId];
646             //browserBox.appendChild(visual.createHTMLFromJSONScope(itemScope));
647             if (typeof innerBox.textContent !== "undefined") {
648                 innerBox.textContent = name + "␣" + percent + "%";//+ " via :";
649             } else {
650                 innerBox.innerText = name + "␣" + percent + "%";//+ " via :";
651             }
652             outerBox.appendChild(visual.createHTMLFromJSONScope(itemScope));
653             outerBox.appendChild(innerBox);
654             var directRelations = analyzer.itemScopesToContainedItemScopes[
                    itemScopeId];
655             if (directRelations !== undefined) {
656                 for (var j = 0; j < directRelations.length; j++) {
657                     var relItemScope = analyzer.jsonHash[directRelations[j]];
658                     if (relItemScope !== undefined) {
659                         if (listManager.indexOf(relItemScope.scopeId) < 0) {
660                             listManager.push(relItemScope.scopeId);
661                             var relItemScopeUL = visual.createHTMLFromJSONScope(
                                    relItemScope);
662                             innerBox.appendChild(relItemScopeUL);
663                         }
664                     }
665                 }
666             }
667             if (typeof innerBox.textContent !== "undefined") {
668                 if (innerBox.textContent !== "cross_connection␣" + percent + "%") {
669                     browserBox.appendChild(outerBox);
670                 }
671             } else {
672                 if (innerBox.innerText !== "cross_connection␣" + percent + "%") {
673                     browserBox.appendChild(outerBox);
674                 }
675             }
676         }
677     },
678
679     /**
680      * creates the analyzer display and returns the wanted display tab
681      * @param nameVar : id of display tab div as string
682      * @returns {HTMLElement}
683      */
684     getDisplay: function (nameVar) {
685         var name = nameVar + "Box";
686         if (document.getElementById(name) === null) {
687             var display = document.createElement("div");
688             display.id = "semantic_container";
689             display.style.width = "150px";
690             display.style.height = "30px";
691             display.style.position = "fixed";
692             display.style.border = "1px␣solid␣black";
693             display.style.background = "white";
694             display.style.top = "0";
695             display.style.right = "0";
696             display.style.zIndex = "9999999999";
```

```
697             display.style.fontSize = "16px";
698             display.style.fontFamily = "monospace";
699             //display.style.fontWeight = "bold";
700
701             var webPageButton = helper.createButton("WebPage", "#2daebf");
702             var directConnectionButton = helper.createButton("DirectConnection", "#91
                    bd09");
703             var directCrossConnectionButton = helper.createButton("
                    DirectCrossConnection", "#91bd09");
704             var browserButton = helper.createButton("Browser", "#ffb515");
705             var webPageTab = helper.createTab("WebPage", "#2daebf");
706             var directConnectionTab = helper.createTab("DirectConnection", "#91bd09")
                    ;
707             var directCrossConnectionTab = helper.createTab("DirectCrossConnection",
                    "#91bd09");
708             var browserTab = helper.createTab("Browser", "#ffb515");
709             var b0 = document.createElement("div");
710             b0.style.paddingLeft = "20px";
711             b0.style.float = "left";
712             b0.style.height = "20px";
713             if (typeof b0.textContent !== "undefined") {
714                 b0.textContent = name;
715             } else {
716                 b0.innerText = name;
717             }
718             b0.onclick = function () {
719                 var container = document.getElementById("semantic_container");
720                 if (container.style.width === "150px") {
721                     container.style.width = "1024px";
722                     container.style.height = "800px";
723                     webPageButton.style.display = 'block';
724                     directConnectionButton.style.display = 'none';
725                     directCrossConnectionButton.style.display = 'block';
726                     browserButton.style.display = 'block';
727                     webPageTab.style.display = 'block';
728                 } else {
729                     container.style.width = "150px";
730                     container.style.height = "30px";
731                     webPageButton.style.display = 'none';
732                     directConnectionButton.style.display = 'none';
733                     directCrossConnectionButton.style.display = 'none';
734                     webPageTab.style.display = 'none';
735                     directConnectionTab.style.display = 'none';
736                     directCrossConnectionTab.style.display = 'none';
737                     browserButton.style.display = 'none';
738                 }
739             };
740             display.appendChild(b0);
741             display.appendChild(webPageButton);
742             display.appendChild(directConnectionButton);
743             display.appendChild(directCrossConnectionButton);
744             display.appendChild(browserButton);
745             display.appendChild(webPageTab);
746             display.appendChild(directConnectionTab);
747             display.appendChild(directCrossConnectionTab);
```

```
748             display.appendChild(browserTab);
749             var body = document.getElementsByTagName("body")[0];
750             body.appendChild(display);
751             return document.getElementById(name);
752         } else {
753             return document.getElementById(name);
754         }
755     }
756 };
757 var storage = {
758     /**
759      * deletes all entries by resetting the local storage
760      */
761     resetLocalStorage: function () {
762         localStorage.setItem("jsonHash", JSON.stringify({}));
763         localStorage.setItem("itemScopeIdList", JSON.stringify([]));
764         localStorage.setItem("itemScopesToContainedItemScopes", JSON.stringify({}));
765         localStorage.setItem("possibleItemScopeRelationHash", JSON.stringify({}));
766         localStorage.setItem("possibleItemScopeRelationHashWithT", JSON.stringify({})
            );
767         localStorage.setItem("directCrossConnectionToItemScopeHash", JSON.stringify
            ({}));
768         localStorage.setItem("directItemScopeToCrossConnectionHash", JSON.stringify
            ({}));
769         localStorage.setItem("directCrossConnectionToItemScopeHashWithT", JSON.
            stringify({}));
770         localStorage.setItem("directItemScopeToCrossConnectionHashWithT", JSON.
            stringify({}));
771
772     },
773     /**
774      * load the local storage into the JavaScript environment as JSON objects
775      * eg: done before every web page is loaded
776      */
777     readFromLocalStorage: function () {
778         analyzer.jsonHash = JSON.parse(localStorage.getItem("jsonHash")) || {};
779         analyzer.itemScopeIdList = JSON.parse(localStorage.getItem("itemScopeIdList")
            ) || [];
780         analyzer.itemScopesToContainedItemScopes = JSON.parse(localStorage.getItem("
            itemScopesToContainedItemScopes")) || {};
781         analyzer.possibleItemScopeRelationHash = JSON.parse(localStorage.getItem("
            possibleItemScopeRelationHash")) || {};
782         analyzer.possibleItemScopeRelationHashWithT = JSON.parse(localStorage.getItem
            ("possibleItemScopeRelationHashWithT")) || {};
783         analyzer.directCrossConnectionToItemScopeHash = JSON.parse(localStorage.
            getItem("directCrossConnectionToItemScopeHash")) || {};
784         analyzer.directItemScopeToCrossConnectionHash = JSON.parse(localStorage.
            getItem("directItemScopeToCrossConnectionHash")) || {};
785         analyzer.directCrossConnectionToItemScopeHashWithT = JSON.parse(localStorage.
            getItem("directCrossConnectionToItemScopeHashWithT")) || {};
786         analyzer.directItemScopeToCrossConnectionHashWithT = JSON.parse(localStorage.
            getItem("directItemScopeToCrossConnectionHashWithT")) || {};
787     },
788     /**
789      * stores the JSON objects into the local storage system
```

```
790       */
791     writeToLocalStorage: function () {
792         //create json object from local storage
793         localStorage.setItem("jsonHash", JSON.stringify(analyzer.jsonHash));
794         localStorage.setItem("itemScopeIdList", JSON.stringify(analyzer.
              itemScopeIdList));
795         localStorage.setItem("itemScopesToContainedItemScopes", JSON.stringify(
              analyzer.itemScopesToContainedItemScopes));
796         localStorage.setItem("possibleItemScopeRelationHash", JSON.stringify(analyzer
              .possibleItemScopeRelationHash));
797         localStorage.setItem("possibleItemScopeRelationHashWithT", JSON.stringify(
              analyzer.possibleItemScopeRelationHashWithT));
798         localStorage.setItem("directCrossConnectionToItemScopeHash", JSON.stringify(
              analyzer.directCrossConnectionToItemScopeHash));
799         localStorage.setItem("directItemScopeToCrossConnectionHash", JSON.stringify(
              analyzer.directItemScopeToCrossConnectionHash));
800         localStorage.setItem("directCrossConnectionToItemScopeHashWithT", JSON.
              stringify(analyzer.directCrossConnectionToItemScopeHashWithT));
801         localStorage.setItem("directItemScopeToCrossConnectionHashWithT", JSON.
              stringify(analyzer.directItemScopeToCrossConnectionHashWithT));
802     }
803 };
804 var helper = {
805     isPlural: function (cName, sName) {
806         var isPlural = false;
807         if (cName + "s" === sName || cName === sName + "s") {
808             isPlural = true;
809         }
810         if (cName === "actors" || sName === "actors") {
811             isPlural = true;
812         }
813         return isPlural;
814     },
815     percents: function (firstItem, secondItem) {
816         var countedPropertyCount = 0;
817         for (var i = 0; i < firstItem.children.length; i++) {
818             var propertyObject1 = firstItem.children[i];
819             var propertyObjectNodeName1 = propertyObject1['nodeName'];
820             for (var j = 0; j < secondItem.children.length; j++) {
821                 var propertyObject2 = secondItem.children[j];
822                 var propertyObjectNodeName2 = propertyObject2['nodeName'];
823                 if (propertyObjectNodeName1 !== "thumbnailUrl" &&
                        propertyObjectNodeName1 !== "url" && propertyObjectNodeName1 !== "
                        image" && propertyObjectNodeName1 !== "provider") {
824                     if (propertyObjectNodeName2 !== "thumbnailUrl" &&
                            propertyObjectNodeName2 !== "image" && propertyObjectNodeName2
                             !== "provider") {
825                         if (propertyObjectNodeName1 === propertyObjectNodeName2) {
826                             if (propertyObject1['nodeValue'] === propertyObject2['
                                    nodeValue']) {
827                                 countedPropertyCount++;
828                                 break;
829                             }
830                         }
831                     }
```

```
832                 }
833             }
834         }
835         var perc1 = ((100 / firstItem.children.length ) * countedPropertyCount).
                toFixed(2);
836         var perc2 = ((100 / secondItem.children.length ) * countedPropertyCount).
                toFixed(2);
837         return [perc1, perc2];
838     },
839     toLowerCaseTrimmedSpaces: function (string) {
840         return string.toLowerCase().replace(/\s+/g, '␣').replace(/^\s+|\s+$/, '');
841     },
842     cleanUrl: function (string) {
843         var ignoredUrls = ["www.imdb.com/plugins", "www.imdb.com/offsite",
844             "www.imdb.com/video", "www.imdb.com/language",
845             "www.imdb.com/country", "www.imdb.com/search", "pro.imdb.com/signup"];
846         for (var i = 0; i < ignoredUrls.length; i++) {
847             if (string.indexOf(ignoredUrls[i]) > -1) {
848                 return "";
849             }
850         }
851         if (string.indexOf("//www.imdb.com/") > -1) {
852             //remove unnecessary reference from link to improve mapping of person
                    instances
853             var splitUrl = string.split("?");
854             string = splitUrl[0];
855         }
856         return string;
857     },
858     objectDoesNotExist: function (currentItem) {
859         var doesNotExists = true;
860         var currentItemKeys = Object.keys(analyzer.jsonHash);
861         for (var x = 0; x < currentItemKeys.length; x++) {
862             var currentItemKey = currentItemKeys[x];
863             var storageItem = analyzer.jsonHash[currentItemKey];
864             //when same source -> check fore more similarities
865             if (currentItem !== undefined && storageItem !== undefined) {
866                 if (currentItem['nodeSource'] === storageItem['nodeSource']) {
867                     //when same type -> compare properties
868                     if (currentItem['nodeType'] === storageItem['nodeType']) {
869                         //when same name -> check fore more similarities
870                         if (currentItem['nodeName'] === storageItem['nodeName']) {
871                             //when same value -> check fore more similarities
872                             var cleanCurrentItemNodeValue = helper.
                                    toLowerCaseTrimmedSpaces(currentItem['nodeValue']);
873                             var cleanStorageItemNodeValue = helper.
                                    toLowerCaseTrimmedSpaces(storageItem['nodeValue']);
874                             if (cleanCurrentItemNodeValue ===
                                    cleanStorageItemNodeValue) {
875                                 //if children (properties) exist and are same length
                                        compare
876                                 var currentChildren = currentItem.children;
877                                 var storageChildren = storageItem.children;
878                                 if ((currentChildren !== undefined) && (
                                        storageChildren !== undefined)) {
```

```
879                                 var cLength = currentChildren.length;
880                                 var sLength = storageChildren.length;
881                                 if (cLength > 0 && sLength > 0) {
882                                     if (cLength === sLength) {
883                                         if (cLength === 1) {//nodeName === url
                                                and no name propertie exists
884                                             if (currentChildren[0]['nodeType']
                                                === storageChildren[0]['nodeType'
                                                ]) {
885                                                 if (currentChildren[0]['nodeName'
                                                    ] === storageChildren[0]['
                                                    nodeName']) {
886                                                     var
                                                        cleanCurrentItemChildNodeValue
                                                         = helper.
                                                        toLowerCaseTrimmedSpaces(
                                                        currentChildren[0]['
                                                        nodeValue']);
887                                                     var
                                                        cleanStorageItemChildNodeValue
                                                         = helper.
                                                        toLowerCaseTrimmedSpaces(
                                                        storageChildren[0]['
                                                        nodeValue']);
888                                                     if (
                                                        cleanCurrentItemChildNodeValue
                                                         ===
                                                        cleanStorageItemChildNodeValue
                                                        ) {
889                                                         doesNotExists = false;
890                                                         return doesNotExists;
891                                                     }
892                                                 }
893                                             }
894                                         } else {
895                                             //compare name properties for final
                                                comparison
896                                             var duplicatePropertyCount = 0;
897                                             for (var j = 0; j < cLength; j++) {
898                                                 var currentChildNodeName =
                                                    currentChildren[j]['nodeName'
                                                    ];
899                                                 if (currentChildNodeName === "
                                                    thumbnailUrl" ||
                                                    currentChildNodeName === "
                                                    image" || currentChildNodeName
                                                     === "provider") {
900                                                     duplicatePropertyCount += 1;
901                                                 } else {
902                                                     for (var k = 0; k < sLength;
                                                        k++) {
903                                                         if (currentChildNodeName
                                                            === storageChildren[k
                                                            ]['nodeName']) {
```

```
904                                                                 var cleanCurrentItemChildNodeValue = helper.toLowerCaseTrimmedSpaces(currentChildren[j]['nodeValue']);
905                                                                 var cleanStorageItemChildNodeValue = helper.toLowerCaseTrimmedSpaces(storageChildren[k]['nodeValue']);
906                                                                 if (cleanCurrentItemChildNodeValue === cleanStorageItemChildNodeValue) {
907                                                                     duplicatePropertyCount += 1;
908                                                                     break;
909                                                                 }
910                                                             }
911                                                         }
912                                                     }
913                                                     //if all properties are the same -> ignore item
914                                                 }
915                                                 if (duplicatePropertyCount === sLength) {
916                                                     doesNotExists = false;
917                                                     return doesNotExists;
918                                                 }
919                                             }
920                                         }
921                                     } else {
922                                         //both the same with no children
923                                         doesNotExists = false;
924                                         return doesNotExists;
925                                     }
926                                 } else {//both the same with undefined children
927                                     doesNotExists = false;
928                                     return doesNotExists;
929                                 }
930                             }
931                         }
932                     }
933                 }
934             }
935         }
936         return doesNotExists;
937     },
938     sortAscending: function (unsortedList) {
939         return unsortedList.sort(function (a, b) {
940             return a - b
941         });
```

```
942     },
943     sortByLengthDesc: function (unsortedList) {
944         return unsortedList.sort(function (a, b) {
945             if (a.length > b.length)
946                 return -1;
947             if (a.length < b.length)
948                 return 1;
949             return 0;
950         });
951     },
952     createTab: function (name, color) {
953         var tab = document.createElement("div");
954         tab.id = name + "Tab";
955         tab.className = "SemanticTab";
956         tab.style.width = "100%";
957         tab.style.height = "779px";
958         tab.style.overflowY = "auto";
959         tab.style.borderTop = "1px solid black";
960         tab.style.background = "white";
961         tab.style.top = "20px";
962         tab.style.right = "0";
963         tab.style.backgroundColor = color;
964         tab.style.display = 'none';
965         var box = document.createElement("div");
966         box.style.width = "100%";
967         box.style.height = "100%";
968         box.id = name + "Box";
969         tab.appendChild(box);
970         return tab;
971     },
972     createButton: function (name, color) {
973         var button = document.createElement("div");
974         button.style.padding = "1px 10px";
975         button.id = name + "Button";
976         button.style.float = "left";
977         button.style.backgroundColor = color;
978         button.style.height = "20px";
979         button.style.display = "none";
980         if (typeof button.textContent !== "undefined") {
981             button.textContent = name;
982         } else {
983             button.innerText = name;
984         }
985         button.onclick = function () {
986             helper.toggleTab(name);
987         };
988         return button;
989     },
990     toggleTab: function (name) {
991         var tabs = document.getElementsByClassName("SemanticTab");
992         for (var i = 0; i < tabs.length; i++) {
993             var tab = tabs[i];
994             if (tab.id == name + "Tab") {
995                 tab.style.display = 'block';
996             } else {
```

```
997                 tab.style.display = 'none';
998             }
999         }
1000     }
1001 };
1002 var run = {
1003     run: function (auto, analyze, map, render) {
1004         console.time('run');
1005         if (analyze) {
1006             run.localStorageToJavaScript();
1007             run.analysis();
1008             run.javaScripToLocalStorage();
1009         }
1010         if (map) {
1011             run.localStorageToJavaScript();
1012             run.mapping();
1013             run.javaScripToLocalStorage();
1014         }
1015         if (render) {
1016             run.localStorageToJavaScript();
1017             run.rendering();
1018         }
1019         if (auto) {
1020             automatize.extractMovieFromIMDB();
1021         }
1022         console.timeEnd('run');
1023         return false;
1024     },
1025     analysis: function () {
1026         console.log("START analysis");
1027         console.log("--------------");
1028         console.time('analysis');
1029         console.log("analyzing : extracting data from webpage");
1030         analyzeRunStart = new Date().getTime().toString();
1031         analyzer.extract.JSONRootItemScopesFromWebpage(analyzer.extract.helper.
             HTMLRootItemScopesFromWebpage());
1032         analyzeRunEnd = new Date().getTime().toString();
1033         analyzer.timesTaken += "analyze" + (analyzeRunEnd - analyzeRunStart).toFixed
             (2) + "ms ";
1034         console.timeEnd('analysis');
1035         console.log("------------");
1036     },
1037     mapping: function () {
1038         console.log("START mapping");
1039         console.log("-------------");
1040         console.log("mappping : find connections between references");
1041         var mapRunStart = new Date().getTime().toString();
1042         analyzer.mapAllForSemanticReference();
1043         var mapRunEnd = new Date().getTime().toString();
1044         analyzer.timesTaken += "map" + (mapRunEnd - mapRunStart).toFixed(2) + "ms ";
1045         console.log("-----------");
1046     },
1047     rendering: function () {
1048         console.log("START rendering");
1049         console.log("---------------");
```

```
1050            console.time('rendering');
1051            var renderRunStart = new Date().getTime().toString();
1052            console.time('currentWebPageItems');
1053            visual.render.currentWebPageItems();
1054            console.timeEnd('currentWebPageItems');
1055            //console.time('directConnections');
1056            //visual.render.directConnections();
1057            //console.timeEnd('directConnections');
1058            console.time('directCrossConnection');
1059            visual.render.directCrossConnection();
1060            console.timeEnd('directCrossConnection');
1061            var renderRunEnd = new Date().getTime().toString();
1062            analyzer.timesTaken += "render" + (renderRunEnd - renderRunStart).toFixed(2)
                    + "ms ";
1063            console.timeEnd('rendering');
1064            console.log("-------------");
1065        },
1066        localStorageToJavaScript: function () {
1067            console.log("START : read from LocalStorage store in JS");
1068            var loadRunStart = new Date().getTime().toString();
1069            storage.readFromLocalStorage();
1070            var loadRunEnd = new Date().getTime().toString();
1071            analyzer.timesTaken += "load" + (loadRunEnd - loadRunStart).toFixed(2) + "ms
                    ";
1072            console.log("END : read from LocalStorage store in JS");
1073        },
1074        javaScripToLocalStorage: function () {
1075            console.log("START : update LocalStorage from code");
1076            storeRunStart = new Date().getTime().toString();
1077            storage.writeToLocalStorage();
1078            storeRunEnd = new Date().getTime().toString();
1079            analyzer.timesTaken += "store" + (storeRunEnd - storeRunStart).toFixed(2) + "
                    ms ";
1080            console.log("END : update LocalStorage from code");
1081        },
1082        deleteLocalStorage: function () {
1083            console.log("START : delete LocalStorage");
1084            console.time('deleteLocalStorage');
1085            storage.resetLocalStorage();
1086            console.timeEnd('deleteLocalStorage');
1087            console.log("END : delete LocalStorage");
1088        }
1089    };
1090
1091    var automatize = {
1092        IMDB: {},
1093        count: 0,
1094        extractMovieFromIMDB: function () {
1095            automatize.IMDB = JSON.parse(localStorage.getItem("SemanticIMDB")) || {};
1096            var url = "http://www.imdb.com/title/";
1097            if (document.location.href.indexOf(url) > -1) {
1098                var split = document.location.href.split("/tt");
1099                var currentId = parseInt(split[1]);
1100                var lastVisited = 87331;
1101                if (automatize.IMDB[url] === undefined) {
```

```
1102                automatize.IMDB[url] = {};
1103            }
1104            if (automatize.IMDB[url]["tt"] === undefined) {
1105                automatize.IMDB[url]["tt"] = {};
1106            }
1107            if (automatize.IMDB["count"] === undefined) {
1108                automatize.IMDB["count"] = 0;
1109            }
1110            if (automatize.IMDB[url]["lastVisited"] !== undefined) {
1111                lastVisited = automatize.IMDB[url]["lastVisited"];
1112            }
1113            var nextVisit = lastVisited + 1;
1114            if (currentId == nextVisit) {
1115                automatize.IMDB["count"] = 0;//reset count
1116                automatize.IMDB[url]["lastVisited"] = currentId;
1117                automatize.IMDB[url]["tt"][currentId] = analyzer.timesTaken;
1118                localStorage.setItem("SemanticIMDB", JSON.stringify(automatize.IMDB))
                    ;
1119                document.location.href = url + "tt" + (currentId + 1) + "/";
1120            } else {
1121                automatize.IMDB["count"] += 1;//reset count
1122                if (automatize.IMDB["count"] == 2) {
1123                    automatize.IMDB[url]["lastVisited"] = nextVisit;
1124                    automatize.IMDB["count"] = 0;
1125                }
1126                localStorage.setItem("SemanticIMDB", JSON.stringify(automatize.IMDB))
                    ;
1127                var finalUrl = url + "tt" + nextVisit + "/";
1128                document.location = finalUrl;
1129            }
1130        }
1131    }
1132 };
1133 //automatize,analyze,map,render
1134 //run.run(true, true, false, false);//automatic IMDB
1135 //run.run(false, true, false, false);//analyze
1136 //run.run(false, false, true, false);//map
1137 //run.run(false, false, false, true);//render
1138 run.run(false, true, true, true);//all
```