Diplomová práce

České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Minimální dokument

Jakub Podlaha

May 2014

/ Prohlášení

Prohlašuji, že jsem se neflákal.

Abstrakt / Abstract

Tento dokument je pouze pro potřeby testování.

This document is for testing purpose only.

/ Obsah

Kapitola 1
Introduction

During past few years the Web went through bigger or smaller revolutions.

- WEB 2.0 and tag cloud
- HTML5 and semantic tags
- Smartphones, Tablets and mobile web everywhere,
- The run out of IPv4 addresses, nonexistent boom of IPv6,
- Cloud technologies and BigData,
- Bitcoin, Tor, anonymous internet,
- WikiLeaks, NSA, Heartbleed and secourity concerns
- Google Knowledge Graph, Facebook Open Graph, ...

That's only few examples of some of the biggest recent issues on the web in general. We live in an age, where so little can mean so much. The enviroment online is dramatically changing, mostly on a wave of some new, useful or frightening technology. The Semantic Web technologies have been described, standardized and implemented for several year now (XXX Example with linked data, rdf) and their tide is near, though yet to come.

Semantic Web itself reates to several principles (and their implementation) that allow users to add meaning to their data. This meaning brings not only a standardized structure, but also posibility to query and reason on it. Once given the structure, similar data can be also joined in a form of a bigger cloud. This phenomena is called Linked Data.

In this work we'd like to bring the Semantic Web technologies closer to users. The approach is to propose a methodology for extracting structured data out of unstructured ones, designing and implementing an appropriate tool, to simplify the process of annotating (yet anonymous) data on a webpage, i.e. to bring structure and meaning into it.

1.1 Problem Statement and Motivation

Giving meaning, i.e. semantization of web pages gets more popular. The most obvious it's probably on the way google serves it's results. Showing menu fields parsed directly from HTML5, or visualizing data from their own internal ontology

XXX `https://en.wikipedia.org/wiki/Google_Knowledge_Graph`

XXX Strigil - `http://delivery.acm.org/10.1145/2540000/2539170/p453-starka.pdf`

Mhat are the options for bringing sematic into a web?

One direction to go is to annotate data on `the server side`, i.e. at the time it is being created and/or published. The person creating the data have to use the right tool and spend time giving the data the appropriate annotation. There is enough technologies for it: HTML5 adding tags for better annotation of the page structure (such as nav, article, section, aside, ...), microformats `http://microformats.org/` using html classes to bring standardized patterns for several basic use cases with fixed structure, such as vcard or event, or RDFa to annotate data on a webpage with an actual ontology (see in separate section XXX).

There are tools for extracting and testing structured data
`http://www.google.com/webmasters/tools/richsnippets`
`http://rdfa.info/play/`

To bypass the gap between anonymous data present on the web on one side and rich, linked, meaningful ontologies (XXX example) on the other, we can go the opposite direction as well. We can take the unannotated data already present on the web and retrieve them in a form, that is defined by some ontology structure.

To allow such a proccess we need to create tools that allow users to annotate the, previously meaning-free, data with elements of existing ontology. By using existing ontologies we not only give the meaning to our data, but also valuable connection to any other dataset annotated using the same ontology.

1.2 Current solution crOWLer

The suggested base-technology is being developed on our faculty XXX. Crawler called crOWLer serves the needs of extracting data from web. In current technology, both, the scenario and the ontology structure/schema are hard-coded into the crOWLer code. This requires unnecessary load of work for each separate use case, whilst in practice all the use cases share the same workflow.

1. load the ontology
2. add selectors to specific resources from the ontology
3. run the crawling process according the above

1.3 Proposed Solution and Methodology

To simplyfy the creation of guidelines, or scenarios for crOWLer, we propose a tool that allows user to select all the element directly on the web page being crawled, with all the necessary settings, pass the scenario created to the crOWLer and obtain the results in a form of a graphical feedback.

1.4 Specific goals of the project

- design the semantic data creation use-cases
- implement extension for a browser
- load and visualise ontology
- create scenario for crOWLer
- serialize scenario and ontology
- parse it by crOWLer creating it's configuration
- run crOWLer
- visualize the extracted data (feedback)

1.5 Work structure (XXX)

TBD

Kapitola 2
Existing solutions

2.1 Semantic and non semantic crawlers

By researching existing solutions, there is currently no open source or openly available solution to solve this task. Rumor goes there is proprietary tool in IBM.

Existing tools named as `Ontology-based Web Crawlers` refer mostly to crawlers that `rank` pages being crawled by guess-matching them against some ontology. In those programs user can't specify data that are being retrieved. Moreover, there is no way to get involved in the crawling process. It is solely used to automatically rank the relevance of documents. They are solving different task where input is several documents and possibly an ontology and output is the best matching document.

In case we are trying to solve the input is one or more documents and one or more ontologies and the result is data obtained from the documents and annotated with resources from the ontologies.

2.2 Advantages and pitfalls of Semantic crawler and linked data

The simplest approach is manual searching for keywords, or even simple browsing the web. That might be useful in some cases, but when there is a lot of data, it becomes exhausting.

Crawling data using simple tools like 'wget –mirror' allows us to load data and then write a program or script to retrieve a relevant information. This approach takes a lot of energy for one time only solution of a given problem.

By storing such crawled data into database we obtain persistent database, possibly automatically obtained by the script from pervious case. Such data is static, but can be queried over and over and possibly re-retrieved when becomes obsolete. It's structure is, however, based on programmers imagination an needs to be described in order to understand and handle the data properly.

When using Ontology-based solution, tailor made for crawling and annotating data from web, we obtain several benefits `for free`. The tool designed specially for this purpose makes it easy. Once the data is annotated, we can not only query on them, but also automatically reason on them and obtain more or more specific/narrow results than with general data. The atributes and relations within ontology, that allow reasoning, are usually part of the ontology deffinition and as such comes, again, `for free`.

Last for benefits: using ontology from public resource as a schema for our data can give us correct structure without need for XXX making it up or building it from scratch. Also by using some common ontology, we can join together any accessible data structured according to this ontology and simply query on resulting super set. With this approach we can utilize the power of linked data cloud (XXX reference).

Semantic crawling is not a silver bullet. The technology is only finding it's place and uses and it's being shaped by the needs of it's users. In current it's mostly used on accademic field XXX.

There is always a threat of inconsistency of an ontology when some data don't fit the rules or breaks structure of an ontology. (XXX more)

Just like with `hardcoded` crawling technique, the semantic crawling is tightly connected (XXX better) with the structure of the web being crawled and selectors (XXX explain term) used for matching data on the web. Any change on a webpage structure can lead to broken selectors or links during the crawling process (XXX and make the scenario useles, more on self-repairing of scenarios?).

A lot of web pages loads their data dynamically using AJAX queries. Some pages simply changes it's content frequently (XXX typically news pages, forums: rt.com, vimeo.com, ...) which would require almost constant crawling and growth into an massive ontology (XXX any suggestions on that? =).

Stating that, the semantic crawling is an usefull way to effectively obtain and query on (otherwise anonymous) data from the web, but it still have it's challenges to overtake.

2.3 Research - existující řešení - platforma

2.3.1 InfoCram 2000 - Jirka Mašek

- zalozeny na Aardwark [1])

2.3.2 iMacros

- `http://wiki.imacros.net/Command_Reference`
- `http://wiki.imacros.net/iMacros_for_Firefox`
- `http://wiki.imacros.net/iMacros_for_Chrome`

2.3.3 Selenium IDE

- IDE - `http://www.seleniumhq.org/projects/ide/`
- plugins - `http://www.seleniumhq.org/projects/ide/plugins.jsp`
- current commands - `http://release.seleniumhq.org/selenium-core/1.0.1/reference.html`
- documentation - `http://docs.seleniumhq.org/docs/index.jsp`
- extending selenium API (blog, tutorial) - `http://adam.goucher.ca/?s=selenium&paged=2`
  - randomString example - `http://adam.goucher.ca/?p=1348`

2.4 crOWLer

[1]) `https://addons.mozilla.org/en-US/firefox/addon/aardvark/`

2.4.1 zavislosti

- maven - apache project managing tool
  - `https://maven.apache.org`
  - `https://maven.apache.org/run-maven/index.html`
  - `https://maven.apache.org/guides/mini/guide-ide-eclipse.html`
- sesame
  - `http://www.openrdf.org/download.jsp` ??
- jena
  - `https://github.com/ansell/JenaSesame` !!
  - or `https://github.com/afs/JenaSesame` ??
  - or `http://jena.apache.org/` ???
  - or `http://sjadapter.sourceforge.net/` ????
  - or `http://sourceforge.net/projects/jenasesamemodel/`
  - might help `http://www.iandickinson.me.uk/articles/jena-eclipse-helloworld/`
  - little hint `http://spqr.cerch.kcl.ac.uk/?page_id=130`
  - another hit `http://answers.semanticweb.com/questions/20865/how-to-get-the-jena-sesame-adapter`
  - wiki `https://en.wikipedia.org/wiki/Jena_(framework)`
  - jena vs. sesame flame `http://answers.semanticweb.com/questions/1638/jena-vs-sesame-is-there-a-serious-complete-up-to-date-unbiased-well-informed-side-by-side-comparison-between-the-two`

2.4.2 Classes of CrOWLer

- ImmovableHeritageConfiguration extends MonumnetConfiguration implements ConfigurationFactory
  - implements Configuration, which is parameter for FullCrawler.run() method
- FullCrawler
  - implements the whole crawling algorithm
  -

2.4.3 Run configuration

```
  crowler cz.sio2.crowler.configurations.npu.ImmovableHeritageConfiguration file results
  crowler cz.sio2.crowler.configurations.kub1x.KbxConfiguration file results
  crowler cz.sio2.crowler.configurations.parser.SeleniumConfiguration\
           file results generated.html
```

- Class ImmovableHeritageConfiguration implements Configuration class.
- Folder jena_con will be created and all the rdf's will be stored in int with names derived from ontology uri

Kapitola 3
Knowledge base, principles and technologies

Linked Data, RDFa, ...

informativni cast, teorie

Seznamte se technologiemi pro automatickou extrakci dat z webových stránek a s jazyky sémantického webu RDF, RDFS a OWL.

3.1 automatická extrakce dat

3.2 RDF and RDFS

- https://en.wikipedia.org/wiki/Resource_Description_Framework

3.3 OWL

- http://www.w3.org/TR/owl2-primer/
- https://en.wikipedia.org/wiki/Web_Ontology_Language
- http://www.w3.org/TR/2012/REC-owl2-quick-reference-20121211/

3.4 Linked Data

- http://linkeddata.org/guides-and-tutorials
- http://linkeddatabook.com/editions/1.0/
- http://lov.okfn.org/dataset/lov/

3.5 Ontology repositories

- http://www.w3.org/wiki/Ontology_repositories

3.6 RDFa

- https://www.sio2.cz/web/psiotwo/publications
- http://rdfa.info/play/

3.7 dalsi

- https://en.wikipedia.org/wiki/SPARQL
- https://en.wikipedia.org/wiki/Turtle_(syntax)

Kapitola 4
Program design

4.1 Use Cases

- NPU
- beerborec.cz
- citybee.cz

4.2 Model

4.3 Imlementation

4.4 Issues - solved and unsolved

- error handling (non existent selector, missing data, ...)

Kapitola 5
Program Implementation

Kapitola 6
Results and Tests

6.1 Data

6.1.1 Pamatky

- http://onto.mondis.cz/resource/page/npu/
- http://monumnet.npu.cz/pamfond/list.php?hledani=1&KrOk=&HiZe=&VybUzemi=1&sNazSidOb=&Adresa=&Cdom=&Pamatka=&CiRejst=&Uz=B&PrirUbytOd=3.5.1958&PrirUbytDo=10.12.2013
- http://dominanty.cz/pamatky-cihana.php

Kapitola 7
zaver

TBD