Hochschule Luzern

PAWI

Evaluation of different content extraction algorithms

Author:
Joel Rolli

Supervisor:
Patrick Huber / Patrik Lengacher

A thesis submitted in fulfilment of the requirements for the degree of some HSLU degree

in the

Research Group Name
Department or School Name

October 2014

Declaration of Authorship

I, Joel Rolli, declare that this thesis titled, 'Evaluation of different content extraction algorithms' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________

Date: ______________________________

"Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism."

Dave Barry

HSLU

Abstract

Faculty Name
Department or School Name

some HSLU degree

Evaluation of different content extraction algorithms

by Joel Rolli

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too...

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor…

Contents

List of Figures

List of Tables

Abbreviations

LAH	**L**ist **A**bbreviations **H**ere

Physical Constants

Speed of Light	c	$=$	$2.997\ 924\ 58 \times 10^8\ \mathrm{ms}^{-\mathrm{S}}$ (exact)

Symbols

a	distance	m
P	power	W (Js^{-1})
ω	angular frequency	$rads^{-1}$

For/Dedicated to/To my…

Chapter 1

Planning

1.1 Planning concept

So as to plan the project, a combination of the two well known planning frameworks scrum and RUP are used.
For a first rough planning, the assignment is split into working packages and assigned to milestones. Delivery objects are defined for each milestone.
This plan is then assigned to the given time table of about 12 weeks. The project effort is defined as 180 hours. This results in about 15 hours work load per week.

A more detailed planning is done for the incoming milestone / sprint. The predefined working packages are split into smaller packages. For the first draft, only the first milestone is split into smaller packages. The later milestones are going to be defined in more detail as soon as all needed information is available.

The effort needed for the documentation is not listed separately. All the tasks already contain additional time for updating the documentation.

The milestones dates are not finally defined, which means that the meeting dates can vary by up to some days. sadfjsalödkfjsadlk

1.2 Milestones overview

Name	**Shortcut**	**Weeks**	**Estimated hours**	**Hours total**	**Closing date**
Milestone one	m1	2.5	39	39	01.10.2014
Milestone two	m2	3	45	84	22.10.2014
Milestone three	m3	2	30	114	05.11.2014
Milestone four	m4	2	30	144	19.11.2014
Milestone five	m5	2.5	38	182	08.12.2014

1.3 Delivery objects

Milestone	**Delivery date**	**Delivery objects**
Milestone one	01.10.2014	• System specification • Sketch software architecture • Short presentation CI environment • Draft risk evaluation
Milestone two	22.10.2014	• Elaborated software architecture • Tested code of test framework (tbd: which components) • Interface definition for justext/boilerplate components • HTML test data
Milestone three	05.11.2014	• Working test environment with both justext and boilerplate components integrated
Milestone four	19.11.2014	• Evaluation environment for output data of test framework • First approach to new algorithm
Milestone five	08.12.2014	• Implementation of new algorithm • Final documentation • Final presentation

1.4 Milestone one - m1

- Closing date date: 1.10.2014
- Available time: ca. 39h

Story	**Shortcut**	**Estimated time**
Planning	s1	4h
Research HTML / Algorithms	s2	8h
System specification	s3	12h
Risk evaluation	s4	3h
Draft software architecture	s5	8h
Configuration CI environment	s6	4h
Total		39h

1.4.1 Stories m1

Title	Planning
Id	s0
Estimated time	4h
Description	As a project owner, you need to have a time schedule so that you can see when you will achieve which results. The PAWI project is split into several working packages which are then split into single stories. The working packages are assignment to milestones and for each milestone, delivery objects are defined. This can be a document, a piece of test or production code or some other kind of work.

Title	Research HTML / Algorithms
Id	s1
Estimated time	8h
Description	My knowledge of HTML and content extraction algorithms is still limited. In order to find out what challenges I will face and which aspects I will have to take into consideration for performing the first tasks, a short research on these topics is needed.

Title	System specification
Id	s2
Estimated time	12h
Description	The PAWI project is defined through a short project description. This description does not cover all necessary information to both plan and perform this project. The key features, interfaces and delivered objects have to be defined more closely. The system specification should cover all these requirements.

Title	Draft software architecture
Id	s3
Estimated time	8h
Description	A first rough software architecture should be made as soon as possible, so that any misunderstandings between tutors and student can be uncovered. Moreover, it is much easier to plan the further steps when the software is split into several parts.

Title	Risk evaluation
Id	s4
Estimated time	8h
Description	Potential risks should be uncovered with the knowledge that was gathered by defining the specification and the software architecture. What is more, further actions can be defined to minimize the above mentioned risks.

Title	Configuration CI environment
Id	s5
Estimated time	4h
Description	To deliver high quality software a continuous integration environment is required. Following tools should be evaluated and configured for further use: • Version control (git) • Project build automation tool (gradle) • continuous integration service (Travis CI)

1.5 Milestone two - m2

- Closing date date: 22.10.2014
- Available time: ca. 45h

Story	**Shortcut**	**Estimated time**
Implementation test framework	s6	20h
Prototype Integration of justext/boilerpipe	s7	17h
Collection of test data	s8	8h
Total		45h

1.5.1 Stories m2

Title	Implementation Testframework
Id	s6
Estimated time	20h
Description	Implementation of a first part of the test framework. This story will be divided into smaller stories as soon as the software architecture and the system specification is reviewed.

Title	Prototype Integration of justext/boilerpipe
Id	s7
Estimated time	4h
Description	Implementation of a small prototype which uses the existing implementation of justext and boilerpipe. A final interface for both components needs to be defined for further use. This story will be divided into smaller stories as soon as the software architecture and the system specification is reviewed.

Title	Collection of test data
Id	s8
Estimated time	8h
Description	To evaluate the functionality of the text extraction algorithms, a certain amount of test data is needed. This test data contains HTML files of several web pages. The HTML code is categorized into content and boilerplate.

1.6 Milestone three - m3

- Closing date date: 5.11.2014
- Available time: ca. 30

Story	**Shortcut**	**Estimated time**
Implementation test framework	s9	20h
Final integration of justext / boilerplate	s10	10h
Total		30h

1.6.1 Stories m3

Title	Implementation test framework
Id	s9
Estimated time	20h
Description	Final implementation of the test framework. This story will be divided into smaller stories as soon as the software architecture and the system specification is reviewed.

Title	Prototype Integration of justext/boilerpipe
Id	s10
Estimated time	4h
Description	Complete integration of the justext and boilerplate algorithms into the test framework. This story will be divided into smaller stories as soon as the software architecture and the system specification is reviewed.

1.7 Milestone four - m4

- Closing date date: 19.11.2014
- Available time: ca. 30h

Story	**Shortcut**	**Estimated time**
Evaluation environment for results	s11	20h
Research on new algorithm	s12	10h
Total		30h

1.7.1 Stories m4

Title	Evaluation environment of results
Id	s11
Estimated time	20h
Description	The test framework will produce a lot of output data, which has to be reviewed using an evaluation environment. This should process this data and present the results in a descriptive way. This story will be divided into smaller stories as soon as the software architecture and the system specification is reviewed.

Title	Research on new algorithm
Id	s12
Estimated time	20h
Description	A first research on the new algorithm should be performed. After this research it should be possible to decide if this solution is possible and if an implementation with the remaining time resources is realistic. This story will be divided into smaller stories as soon as the software architecture and the system specification is reviewed.

1.8 Milestone five - m5

- Closing date date: 8.12.2014
- Available time: ca. 38h

Story	**Shortcut**	**Estimated time**
Implementation of new algorithm	s13	19h
Complete documentation	s14	15h
Prepare final presentation	s15	4h
Total		38h

1.8.1 Stories m5

Title	Implementation of new algorithm
Id	s13
Estimated time	19h
Description	Implementation of the new algorithm and analysis of the test results with the existing evaluation environment.

Title	Complete documentation
Id	s14
Estimated time	15h
Description	Complete and review all chapters of the documentation.

Title	Prepare final presentation
Id	s15
Estimated time	4h
Description	Prepare the final presentation and the final printed / digital version of the thesis.

Chapter 2

Software requirement specification

2.1 Introduction

2.1.1 Purpose

The software requirement specification should provide all needed information to develop the context extraction framework and define all delivery objects. All interfaces to external components, input and output data, deployment considerations and quality attribute should be well defined within this document.

2.1.2 Scope

The context extraction framework will perform automated text extraction on a set of HTML test data with two to three different text extraction algorithms. The performance of each algorithm is measured and an output file with the measured results is generated.

2.2 General description

2.2.1 Operating Environment

2.2.1.1 Local environment

JDK	1.7.X
Gradle	1.1
Eclipse Keppler	2.X
git	1.9.X
python	2.7.X

2.2.1.2 Continuous Integration Environment

Open JDK	1.6.X
Open JDK	1.7.X
Oracle JDK	1.7.X
Oracle JDK	1.8.X
Gradle	1.1
Travis CI	

2.2.2 Design and Implementation Constraints

2.2.2.1 User interface

As parts of the text extraction framework may be implemented in a server environment and a user interface is not desired from the client so there will be no graphical user interface. The application is built, deployed and started by gradle. While the application is running, no interaction with the user is needed.

2.3 System Features

This section specifies all system features. Each feature is specified more close with multiple user stories but all important information such as external dependencies and output files are defined in this chapter. The related user stories are located in the planning section.

2.3.1 Overview

2.3.1.1 Read configuration

Name	Read configuration
Feature id	f1
Description	The text extraction framework is configurable with an external text file. The configuration file will contain following items: • Path to folder with HTML files • Path to folder with text files • Path to folder with output files • Configuration for algorithms • etc. The configuration file location is defined as a relative path to the source directory and structured in a key value list: `key:value;` `key:value;` `key:value;`
Relevance	needed
Related stories	tbd

2.3.1.2 Create test

Name	Create test
Feature id	f2
Description	A test contains two input files which are a HTML file and a text file. They are located in the directories defined by the configuration. As soon as the test framework finds an HTML and a text file with the same name, a new test is created, the files are read and the test is started.
Relevance	needed
Related stories	tbd

2.3.1.3 Integration Justext algorithm

Name	Integration Justext algorithm
Feature id	f3
Description	Justext is implemented in python so a service is needed to call the python script and get the extracted text or the extracted blocks.
Relevance	needed
Related stories	tbd

2.3.1.4 Integrate Boilerpipe algorithm

Name	Integration Boilerpipe algorithm
Feature id	f4
Description	Boilerplate is implemented in Java so an interface is needed to call the Boilerplate component and get the extracted text or the extracted blocks.
Relevance	needed
Related stories	tbd

2.3.1.5 Integrate RSS feed algorithm

Name	Evaluation and implementation RSS feed algorithm
Feature id	f5
Description	The basic idea of the RSS feed algorithm is to match the content of a HTML document with the related RSS feed and define the relevant content like that. This need to be evaluated, implemented and integrated into the text extraction framework
Relevance	nice to have
Related stories	tbd

2.3.1.6 Evaluation of classification text

Name	Evaluation of classification
Feature id	f6
Description	All the text extraction algorithms return an extracted document as text. This document needs to be checked for correctness. To do so the result from the algorithms is compared with the predefined content. This evaluation and classification is defined in more detail here: 2.3.2 • Check each classified block from the algorithms if it's content can be found in the content file • Categorize all blocks as true negative and false positive • Put the results into an output text file (structure output file: tbd)
Relevance	needed
Related stories	tbd

2.3.1.7 Evaluation of classification blocks

Name	Evaluation of classification blocks
Feature id	f6
Description	A more detailed evaluation of the algorithms could be done if not only the text is classified but each block of an HTML file. To do so, the implementation of Justext and Boilerpipe have to be addapted that they return classified blocks instead of the extracted text. These blocks are then compared with the predefined content and classified. This evaluation and classification is defined in more detail in following chapter: 2.3.2 • Check each classified block from the algorithms if it's content can be found in the content file • Categorize all blocks as true negative and false positive • Put the results into an output text file (structure output file: tbd)
Relevance	nice to have
Related stories	tbd

2.3.1.8 Analyze data

Name	Analyze data
Feature id	f7
Description	From the results of the comparison several further values can be calculated. Some possible values are: • $Presicion = \frac{TP}{TP+FP}$ • Recall/True positive rate (TPR): $\frac{TP}{TP+FN}$ • false positive rate (FPR: $\frac{FP}{FP+TN}$ • F-measure: $2 * \frac{presicion*recall}{presicion+recall}$ • Reciever Operation Characteristics (ROC): $TPR = f(FPR)$
Relevance	needed
Related stories	tbd

Name	Visualize data
Feature id	f8
Description	The calculated values from feature f8 are visualized in diagrams. (tbd: which tool)
Relevance	tbd
Related stories	tbd

2.3.2 Further explanation for evaluation of classification

The general meaning of the expressions true positive, true negative, false positive and false negative related to the text extraction topic is shown in following table:

	Classified as content	**Classified as boilerplate**
Acutal content	True positive (TP)	True negative (TN)
Actual boilerplate	True negative (TN)	False negative (FN)

2.3.2.1 Evaluation the results as text

When the results are compared based on the text, the expressions are interpreted as follow:

	Classified as content	**Classified as boilerplate**
Acutal content	Number of words classified as content and are content	Number of words classified as content but are boilerplate
Actual boilerplate	Number of words classified as content but are boilerplate	Number of words classified as boilerplate and are boilerplate

2.3.2.2 Evaluation the results as blocks

When the results are compared based on HTML blocks, the expressions are interpreted as follow:

When the results are compared based on the text the expressions are interpreted as follow:

	Classified as content	**Classified as boilerplate**
Acutal content	Number of blocks classified as content and are content	Number of blocks classified as content but are boilerplate
Actual boilerplate	Number of blocks classified as content but are boilerplate	Number of blocks classified as boilerplate and are boilerplate

2.4 External Interface Requirements

2.4.1 Boilerpipe

The boilerpipe algorithm is already implemented in Java so it is easy to integrate. The API can be found under following link. `https://code.google.com/p/boilerpipe/`

Other useful links:

Getting started: `http://code.google.com/p/boilerpipe/wiki/QuickStart`

javadoc extractor: `http://boilerpipe.googlecode.com/svn/trunk/boilerpipe-core/javadoc/1.0/de/l3s/boilerpipe/extractors/ExtractorBase.HTML`

2.4.2 justext

The justext algorithem is implemented in python and it is not yet defined how it will be integrated into the text extraction framework. See risk analysis for further information. The documentation can be found under following link:

`https://code.google.com/p/justext/`

jython: http://www.jython.org/

Chapter 3

Risk analysis

3.1 Introduction

3.1.1 Purpose

This document evaluates and weights all possible risks and defines actions to minimize them as good as possible.

3.2 Risk evaluation

3.2.1 Unclear requirements

Requirements are somehow vague at the beginning of each project and if they are not well defined as soon as possible, they stay vague trough out the whole project and this can lead to a disaster.

3.2.2 New technologies

The new technologies which are present in this project are:

- Gradle
- Travis CI
- Python

Each of them brings his own risk.

3.2.3 Interface Boilerpipe

The Boilerplate algorithm needs to be integrated into the text extraction framework. Every interface of an external component is a possible risk factor.

3.2.4 Interface Justext

The Justext algorithm needs to be integrated into the text extraction framework. Every interface of an external component is a possible risk factor.

3.2.5 Implementation RSS algorithm

The development and implementation of a new algorithm is predestined to generate risks.

3.3 Assessment of risks

Risk	**Impact**	**Probability of occurrence**	**Risk factor**
Unclear requirements	2	4	8
New technologies	3	3	9
Interface Boilerpipe	5	1	5
Interface Justext	5	5	20
Implementation RSS algorithm	1	5	5

3.4 Consequences

3.4.1 Unclear requirements

As I am working with the client each and every day, it is very easy prevent misunderstandings with asking the client at once. Even though misunderstandings can occur between student and expert. To prevent this, it is necessary to have a document to define the requirements as soon and as exact as possible. This well be done in the form of the system requirement specification in the first mile stone. Possible ambiguities can be clarified at the first mile stone meeting.

3.4.2 New technologies

It is important to do prototyping with new technologies in the first phase of the project to eliminate these risks as soon as possible.

Gralde and Travic CI are needed in the first mile stone to set up the programming environment. So if there is any problem it will occur in a very early stage of the project and a possible solution can be found.

The risks about python are related to the chapter 3.4.4.

3.4.3 Interface Boilerpipe

This risk is rated much lower than the Justext interface because it's implementation is in Java and it provides a Java API. Never then less a prototype should be done as soon as possible to prevent any nasty surprises with the interface.

3.4.4 Interface Justext

This point is classified as the highest risk of all. This is because the implementation is in Python and it is not clarified yet how it will be integrated into the text extraction framework. An analysis of possible solution with prototypes needs to be done as soon as possible.

Possible solution are:

- jython (`http://www.jython.org/`)
- Implementation in Java
- Java Processor Interface

3.4.5 Implementation RSS algorithm

This risk has a very high probability of occurrence because it is very likely that a development and an implementation of a new algorithm is going to cause problems. There is no real solution to that risk. But because of this requirement is nice to have, the impact on the outcome of the project is very low. Further more Patrik Lengacher, the tutor of this project, is very experienced in this subject area and will be able to help out if any problems occur.

Appendix A

Appendix Title Here

Write your Appendix content here.

Bibliography