Chapter 1

Software requirement specification

1.1 Introduction

1.1.1 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.

1.1.2 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.

1.2 General description

1.2.1 Operating Environment

The operation environment for the text extraction framework is defined in this section.

1.2.1.1 Local environment

Ubuntu JDK	12.04 1.7.X
Gradle	1.11
Eclipse Keppler	2.X
git	1.9.X
python	2.7.X

1.2.1.2 Continuous Integration Environment

Ubuntu	12.04
Open JDK	1.6.X
Open JDK	1.7.X
Oracle JDK	1.7.X
Oracle JDK	1.8.X
Gradle	2.0
Travis CI	

1.2.2 Design and Implementation Constraints

1.2.2.1 User interface

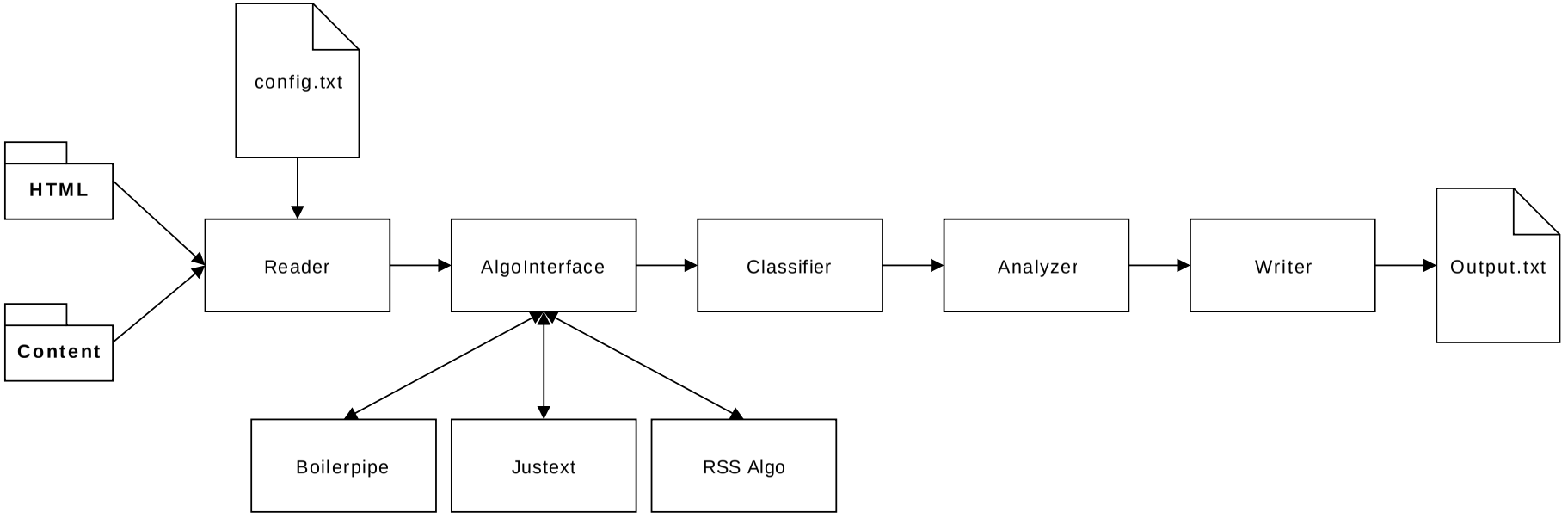
As parts of the text extraction framework may be implemented in a server environment at a later point in time and a user interface is not desired from the client so there will be no graphical user interface. The application is built, deployed and started by gradle. While the application is running, no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

1.3 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.

1.3.1 Basic functionality

Following diagram and text describes the basic functionality of the application.



There are two folders defined by a configuration flies which. The HTML folder contains plain HTML files of web pages. The content folder contains text files with the relevant content of the related HTML files. As soon as a test is started the HTML file and the text file are read and the HTML file is extracted and classified with all the available algorithms. The result of the classification is then compared to the text file with the relevant content and performance data is generated. This performance data is then analyzed with statistical methods.

1.3.2 Overview

ID	Name	Chapter	Relevance
f1	Read configuration	1.3.2.1	needed
f2	Create test	1.3.2.2	needed
f3	Integration Justext algorithm	1.3.2.3	needed
f4	Integration Boilerpipe algorithm	1.3.2.4	needed
f5	Evaluation and Implementation RSS feed algorithm	1.3.2.5	nice to have
f6	Evaluation of classification text	1.3.2.6	needed
f7	Evaluation of classification blocks	1.3.2.7	nice to have
f8	Analyze data	1.3.3	needed

1.3.2.1 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

1.3.2.2 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

1.3.2.3 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

1.3.2.4 Integration 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

1.3.2.5 Evaluation and Implementation 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

1.3.2.6 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 in section 1.3.4. • Check each classified block from the algorithms if it's content can be found in the content file • Categorize text as boilerplate or content • Put the results into an output text file (structure output file: tbd)
Relevance	needed
Related stories	tbd

1.3.2.7 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 adapted 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 section 1.3.4. • Check each classified block from the algorithms if it's content can be found in the content file • Categorize all blocks as boilerplate or content • Put the results into an output text file (structure output file: tbd)
Relevance	nice to have
Related stories	tbd

1.3.3 Analyze data

Name	Analyze data
Feature id	f7
Description	From the results of the comparison several further values can be evaluated for a better understanding of the results. These values are described in more detail in section 1.3.5.
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

1.3.4 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)	False negative(FN)
Actual boilerplate	False positive (FP)	True negative (TN)

When the results are compared based on words, the expressions are interpreted as follow.

	Classified as content	**Classified as boilerplate**
Acutal content	Word classified as content by algorithm and is content	Word classified as content by algorithm but is boilerplate
Actual boilerplate	Word classified as content by algorithm but is boilerplate	Word classified as boilerplate by algorithm and is Boilerplate

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

	Classified as content	**Classified as boilerplate**
Acutal content	Block is classified as content by algorithm and is content	Block is classified as content by algorithm but is boilerplate
Actual boilerplate	Block is classified as content by algorithm but is boilerplate	Block is classified as boilerplate by algorithm and is boilerplate

To sum this up TP + FN is the correct outcome of the algorithm i.e. content classified as content and boilerplate as boilerplate and TN + FP on the other hand is the wrong outcome of the algorithm i.e. content classified as boilerplate and boilerplate as content.

1.3.5 Analytical values

In this paragraph we use the notion of objects instead of word/block. The results of the comparison deliver basic characteristics which can be used to calculate statistical values which help to analyze the test outcome.

Sensitivity / Recall /True positive rate / TPR / Hitrate

Recall is the probability that a relevant document is retrieved in a search which in our case is

$$Recall = \frac{TP}{TP + FN} \tag{1.1}$$

correct classified content objects divided by the sum of all actual objects.

Precision / True negative rate / TNR

Precision is the probability that a retrieved document is relevant which in our case is

$$Presicion = \frac{TP}{TP + FP} \tag{1.2}$$

correct classified content objects divided by the sum of all objects classified as content.

F-measure / F1-score / F-score

F-measure is the weighted harmonic mean of precision and recall which in our case is

$$Fmeasure = 2 * \frac{presicion * recall}{presicion + recall} \tag{1.3}$$

a measure of the test's accuracy.

Fallout / False positive rate / FPR

Fallout is the proportion of non-relevant objects that are retrieved, out of all non-relevant objects available which in our case is

$$Fallout = \frac{FP}{FP + TN} \tag{1.4}$$

1.4 External Interface Requirements

1.4.1 Boilerpipe

The boilerpipe algorithm is implemented in Java and the documentation is found under https://code.google.com/p/boilerpipe/.

1.4.2 justext

The justext algorithem is implemented in python and the documentation is found under https://code.google.com/p/justext/. It is not yet defined how it will be integrated into the text extraction framework. See risk analysis for further information (2.4.4.).

Chapter 2

Risk analysis

2.1 Introduction

2.1.1 Purpose

This document evaluates and calculates all possible risks and defines actions that can minimize these risks as well as possible.

2.2 Risk evaluation

2.2.1 Unclear requirements

The start of a project is normally no easy task because its requirements are vaguely known. If they are not well defined as soon as possible, the requirements will stay vague through out the whole project, which can lead to a disaster.

2.2.2 New technologies

The new technologies which are present in this project are the following:

- Gradle
- Travis CI
- Python

Each of them brings its own risk.

2.2.3 Integration Boilerpipe

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

2.2.4 Integration Justext

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

2.2.5 Implementation RSS algorithm

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

2.3 Assessment of risks

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

2.4 Consequences

2.4.1 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.

2.4.2 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 **??**.

2.4.3 Integration 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.

2.4.4 Integration 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

2.4.5 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.