Software Requirement Specification

1.1 Version

Version	Date	Change	Author
0.1	20.09.2014	Setup document	JR
0.2	28.09.2014	Add features	JR
0.3	30.09.2014	Change features, grammar, layout	JR
0.4	02.10.2014	Add overview of application/evaluation	JR
0.5	04.10.2014	Corrections evaluation	JR
1.0	05.10.2014	Grammar, layout	JR

1.2 Introduction

1.2.1 Purpose

The software requirement specification is providing 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 are well defined within this document.

1.2.2 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. After measuring the performance of each algorithm, an output file with the measured results is generated.

1.3 General description

1.3.1 Operating Environment

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

1.3.1.1 Local environment

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

1.3.1.2 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	

1.3.2 Design and Implementation Constraints

1.3.2.1 User interface

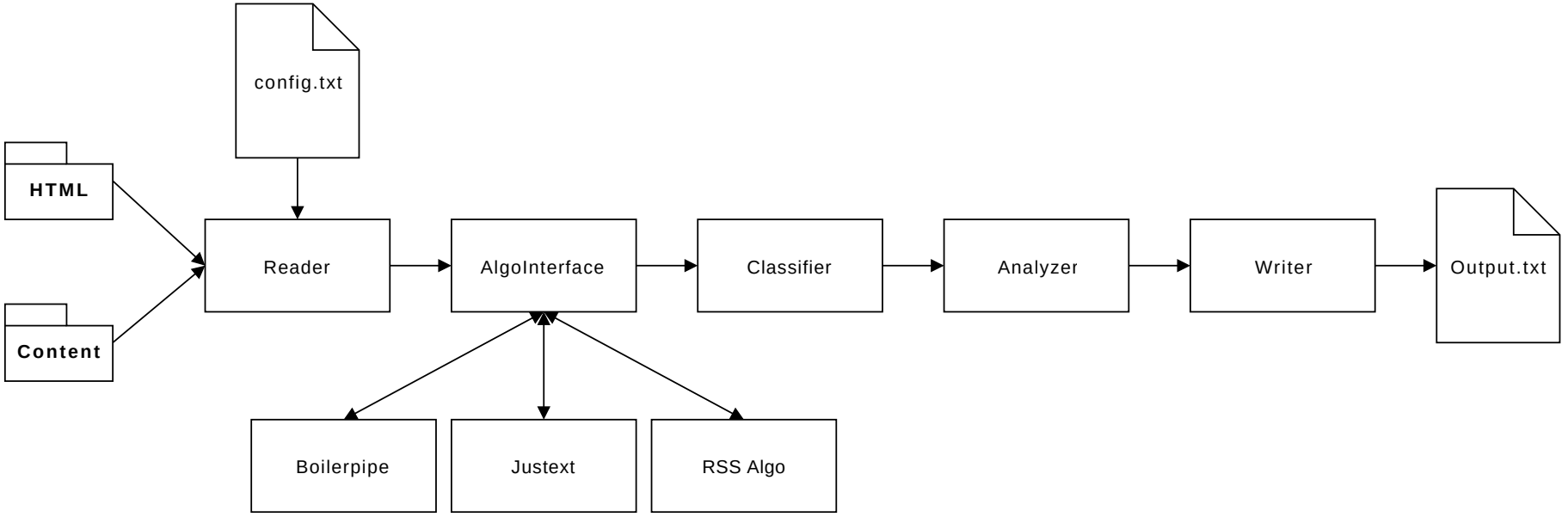
As parts of the text extraction framework may be implemented in a server environment at a later point in time and a user interface is not desired from the client, there will be no graphical user interface. The application is built, deployed and started by gradle. While the application is running, no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

1.4 System Features

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

1.4.1 Basic functionality

The following diagram and text describes the basic functionality of the application.



There are two folders defined by the configuration file (config.txt). The HTML folder contains 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 relevant content and performance data is generated. This performance data is then analyzed with statistical methods.

1.4.2 Overview

ID	Name	Chapter	Relevance
f1	Read configuration	1.4.2.1	needed
f2	Create test	1.4.2.2	needed
f3	Integration Justext algorithm	1.4.2.3	needed
f4	Integration Boilerpipe algorithm	1.4.2.4	needed
f5	Evaluation and Implementation RSS feed algorithm	1.4.2.5	nice to have
f6	Evaluation of classification text	1.4.2.6	needed
f7	Evaluation of classification blocks	1.4.2.7	nice to have
f8	Analyze data	1.4.3	needed

1.4.2.1 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

1.4.2.2 Create test

Name	Create test
Feature id	f2
Description	A test contains two input files which are an 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, the files are read and the test is started.
Relevance	needed
Related stories	tbd

1.4.2.3 Integration Justext algorithm

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

1.4.2.4 Integration Boilerpipe algorithm

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

1.4.2.5 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 an HTML document with the related RSS feed and in doing so, define the relevant content. This needs to be evaluated, implemented and integrated into the text extraction framework.
Relevance	nice to have
Related stories	tbd

1.4.2.6 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 accuracy, which is achieved by comparing the result of the algorithms with the actual content. • Check each classified block from the algorithms if its content can be found in the actual content • Categorize text as boilerplate or content • Insert results in an output text file Both the evaluation and classification are defined in more detail in section 1.4.4.
Relevance	needed
Related stories	tbd

1.4.2.7 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 but also each block of an HTML file is classified. So as to achieve the more detailed evaluation, the implementation of Justext and Boilerpiple has to be adapted so that they return classified blocks instead of the extracted text. These blocks are afterwards compared with the actual content and classified. • Check each classified block from the algorithms if its content can be found in the content file • Categorize all blocks as boilerplate or content • Insert the results in an output text file (structure output file: tbd) Both the evaluation and classification are defined in more detail in section 1.4.4.
Relevance	nice to have
Related stories	tbd

1.4.3 Analyze data

Name	Analyze data
Feature id	f7
Description	From the results of the comparison further values can be evaluated for a better understanding of the results. These values are described in more detail in section 1.4.5.
Relevance	needed
Related stories	tbd

1.4.4 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**
Actual 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 follows.

	Classified as content	**Classified as boilerplate**
Actual 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 follows.

	Classified as content	**Classified as boilerplate**
Actual 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

In conclusion, TP + FN is the correct outcome of the algorithm i.e. content classified as content and boilerplate as boilerplate. On the other hand, TN + FP is the wrong outcome of the algorithm i.e. content classified as boilerplate and boilerplate as content.

1.4.5 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 you 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 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

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

1.5 External Interface Requirements

1.5.1 Boilerpipe

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

1.5.2 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.