WebLicht: Bombarding Services before they Explode

Daniël de Kok, Wei Qiu, Marie Hinrichs

Introduction

- WebLicht
- Measuring User Activity
- Simulating Usage Patterns
- Changes in WebLicht
- Conclusions

WebLicht

Web application for automatic linguistic annotation, such as:

- Morphology
- Morphosyntactic analysis
- Syntactic analysis:
  - Phrase structure
  - Dependencies
- Named entity recognition
- Speech-text alignment

WebLicht is distributed

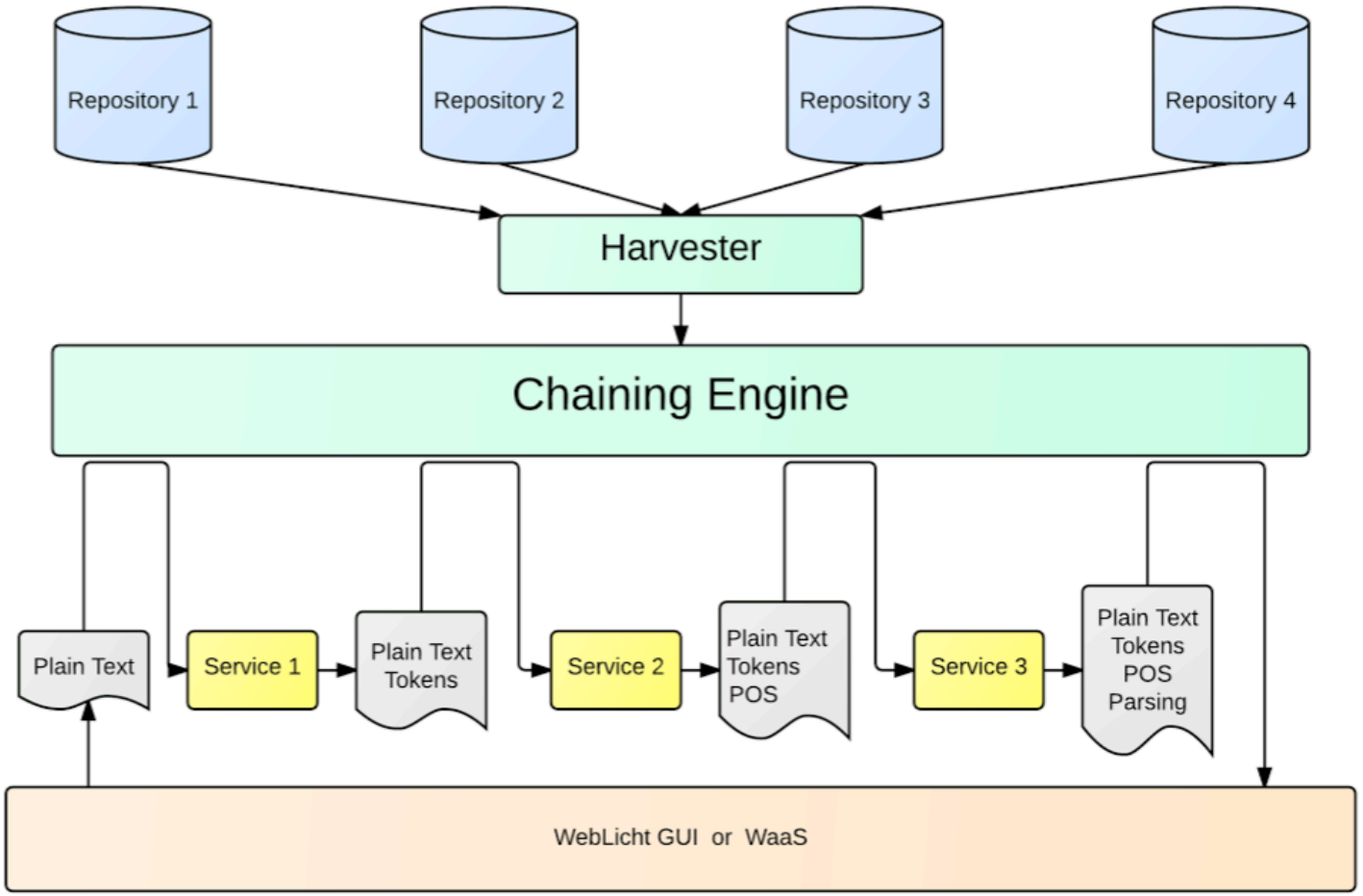
WebLicht is distributed:

- CLARIN centers can contribute annotations tools in the form of web services.
- CMDI metadata describes services:
  - Description
  - Maintainer
  - Input features
  - Output features
- CMDI metadata is harvested every two hours.

Annotation chains

- Annotation tools can be combined to form a chain.
- E.g.: tokenization -> morphosyntactic analysis -> syntactic parsing
- The WebLicht chainer:
  - Ensures that chains are valid (matching input/output)
  - Executes the annotation chain

Overview


Repository 1
Repository 2
Repository 3
Repository 4
Harvester
Chaining Engine
Plain Text
Service 1
Plain Text Tokens
Service 2
Plain Text Tokens POS
Service 3
Plain Text Tokens POS Parsing
WebLicht GUI or WaaS

Growth of WebLicht

- Growth in 2 dimensions:
  - Increasing number of users
  - Larger data sets being submitted for annotation
- Need to understand usage patterns
  - Optimize tools
  - Identify bottlenecks

- Use recent usage statistics to identify which services are growing:
  - Number of jobs
  - Job sizes
- Test services to see if they can handle the (extrapolated) future usage within a wide margin: bombarding

Tools for web analytics

Web analytics tools:

- Webserver log analyzers:
  - Webalizer
  - AWStats
- Analyzers that use Javascript:
  - Google Analytics
  - Piwik

Problems

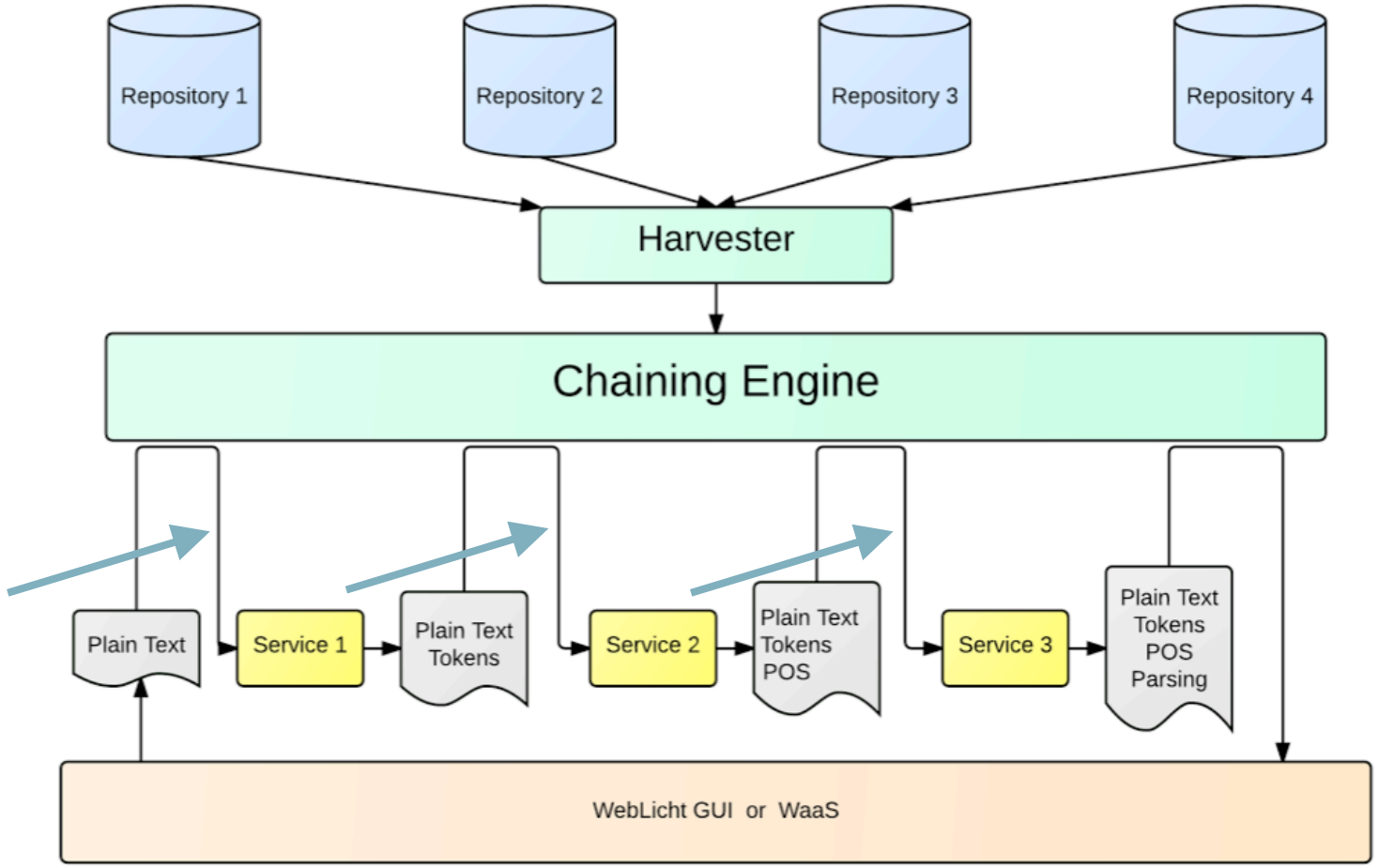
- Log-based approach:
  - WebLicht uses a distributed system.
  - If annotation tools are logged: distributed among many servers at different CLARIN centers.
- Javascript-based approach:
  - Only registers page loads.
  - WebLicht is a single-page application.
  - Annotation tools are not called by the browser.

Solution

- Tools such as Piwik and Google Analytics can be called programmatically (REST call).
- The WebLicht chainer registers each annotation service call via Piwik.

Benefits

- We instantly have statistics for all annotation tools that are available in WebLicht.
- No work for the CLARIN centers that offer annotation tools.
- When new annotation tools are added, statistics for those tools are recorded immediately without manual intervention.


Repository 1
Repository 2
Repository 3
Repository 4
Harvester
Chaining Engine
Plain Text
Service 1
Plain Text
Tokens
Service 2
Plain Text
Tokens
POS
Service 3
Plain Text
Tokens
POS
Parsing
WebLicht GUI or WaaS

Piwik: visitor profile

Visitor profile

IP [redacted]

ID [redacted]

Chrome Mac

Resolution unknown

Summary

Spent a total of **22 min 24s** on the website, and **viewed 16 pages in 1 visits.**
Converted 0 Goals.

First visit

30 Jan 2015 - 10 days ago
from: idp.uni-konstanz.de

Location

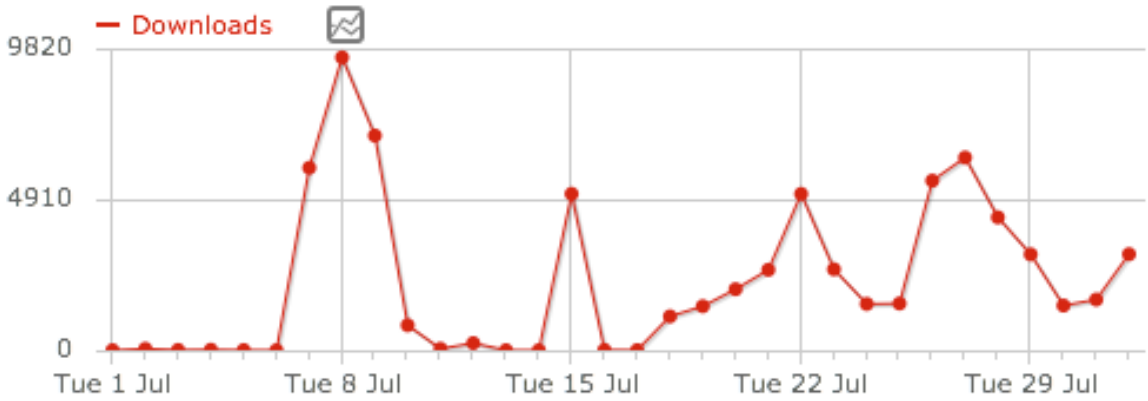
1 visit from Konstanz, Germany *(show map)*

Visited pages

Visit #1 - (22 min 24s) 30 Jan 2015 15:29:47

1. http://weblicht.sfs.uni-tuebingen.de/rws/service-converter...
2. http://clarin05.ims.uni-stuttgart.de/cgi-bin/dspin/tokeniser...
3. http://clarin05.ims.uni-stuttgart.de/treetagger2008
4. http://weblicht.sfs.uni-tuebingen.de/rws/service-converter...
5. http://clarin05.ims.uni-stuttgart.de/cgi-bin/dspin/tokeniser...
6. http://clarin05.ims.uni-stuttgart.de/RFTaggerMorph
7. http://weblicht.sfs.uni-tuebingen.de/rws/service-converter...
8. http://clarin05.ims.uni-stuttgart.de/cgi-bin/dspin/tokeniser...
9. http://clarin05.ims.uni-stuttgart.de/treetagger2008
10. http://weblicht.sfs.uni-tuebingen.de/rws/service-converter...
11. http://clarin05.ims.uni-stuttgart.de/cgi-bin/dspin/tokeniser...
12. http://weblicht.sfs.uni-tuebingen.de/rws/BerkeleyParser_0...
13. http://weblicht.sfs.uni-tuebingen.de/rws/service-converter...
14. http://clarin05.ims.uni-stuttgart.de/cgi-bin/dspin/tokeniser...
15. http://weblicht.sfs.uni-tuebingen.de/rws/service-opennlp/a...
16. http://weblicht.sfs.uni-tuebingen.de/rws/parsers/service-...

Piwik: number of annotation tool invokations



Piwik: invocations per tool

Statistics of the Stuttgart tools

- Tokenizer and RFTagger are highly used
- Such tools should be especially efficient and robust

weblicht.sfs.uni-tuebingen.de	431	53691
clarin05.ims.uni-stuttgart.de	177	18285
/cgi-bin/dspin/tokeniser4.perl	61	3997
/RFTaggerMorph	48	14135
/treetagger2008	45	126
/cgi-bin/dspin/bitpar4.perl	18	20
/cgi-bin/dspin/tei2tcf3.perl	2	3
/rftagger	2	3
/cgi-bin/dspin/smor4.perl	1	1
dspin.dwds.de:8080	26	34
ws1-clarind.esc.rzg.mpg.de	12	13
kaskade.dwds.de	6	6
chopin.ipipan.waw.pl:8083	1	1

Simulating Usage Patterns

Bombard:

- Developed at CLARIN-D center in Tübingen
- Simulates users invoking tool chains
- Flexible test-case configuration
  - annotation tool chain
  - input
  - time interval between toolchain runs
  - maximum permitted processing time

Simulating Usage Patterns

- Many test-cases can be run simultaneously during one “bombardment”
- Bombard reports statistics for each tool:
  - successes/failures
  - average processing times
- Bombard is used by WebLicht tool developers to ensure that they can handle usage patterns reported by Piwik

Example

We noticed that some chains were often used in classroom settings. Used Bombard to simulate:

- 80 simultaneous WebLicht users
- Submitting requests during a period of 2 minutes.
- 40 small texts (couple of paragraphs)
- 40 small novels (Alice in Wonderland)

Types of problems discovered with Bombard:

1. Failure on large texts, e.g. non-linear growth of processing time,
2. Failure on “noisy” texts,
3. Inability to keep up when requests are made in rapid succession (mostly time-intensive tools such as parsers)

Let’s look at our solution for #3

Changes in WebLicht

- Jesque - a distributed task queue framework
- Exploit the parallel processing capabilities of modern servers and computing clusters
  - parallel processing within requests
  - concurrent processing of requests
  - guarantees with respect to usage of resources
  - fairness (small requests should not have to wait for large ones)

Further Work

- Bombard more tools to identify bottlenecks
- Use the extended Jesque framework to improve performance of WebLicht tools
- Link Piwik and Bombard:
  - Detect growth of usage of a chain.
  - Bombard for the expected usage.
  - Report results if there is a problem.

Thank You