RENDER
FP7-ICT-2009-5
Contract no.: 257790
www.render-project.eu

RENDER

Deliverable D1.3.3

Final Web-based Language Corpora Collection (World-wide Collection)

Editor:	Mitja Trampuš, JSI
Author(s):	Mitja Trampuš, JSI; Blaž Novak, JSI; Blaž Fortuna, JSI
Deliverable Nature:	Report (R)
Dissemination Level: (Confidentiality)	Public (PU)
Contractual Delivery Date:	30 September 2013
Actual Delivery Date:	30 September 2013
Suggested Readers:	Researchers and practitioners in NLP
Version:	1.1
Keywords:	

Full Project Title:	RENDER – Reflecting Knowledge Diversity
Short Project Title:	RENDER
Number and Title of Work package:	WP1 – Data Collection and Management
Document Title:	D1.3.3 - Final Web-based Language Corpora Collection (World-wide Collection)
Editor (Name, Affiliation)	Mitja Trampuš, JSI
Work package Leader (Name, affiliation)	Maurice Grinberg, Ontotext

Executive Summary

This document provides the final description of the RENDER data infrastructure which has been the main source of data to be used by the different components of the project and towards accomplishing with the defined use cases.

The document is mostly based on the implemented work from month eighteen until now and is based on the deliverable D1.3.2 “Early prototype of data infrastructure”. Since then, enhancements and updates have been done including the addition of new data sources and some improvements on the infrastructure performance and accessibility.

This deliverable describes data infrastructure covering the needs of project partners as represented through their respective case-studies and the showcase study.

This document does not duplicate or include any information which was already in the deliverable D1.3.1 “Early prototype of data infrastructure” and therefore it may be needed to consult it for a better understanding.

Table of Contents

List of Figures

List of Tables

Abbreviations

NLP	Natural Language Processing
API	Application Programming Interface
POS	Part of Speech
I/O	Input/Output
SNMP	Simple Network Management Protocol
SSD	Solid State Drive
RSS	Really Simple Syndication
XML	eXtended Markup Language

1 Newsfeed

This section provides an updated list of the available data sources within the project and the enhancements done in the news stream infrastructure towards providing a robust and a real time newsfeed as a common data source for the project.

The system performance is measured by the number of articles that the system is able to process and by the capability for processing in real time the actual sources of information. Due to initial hardware constraints the performance of the system was not as good as expected and some improvements were needed. The inclusion of Twitter as a social media source also generates a larger volume of data to be processed compared to the volume generated by press agencies or news publishers.

These two aspects, a) the inclusion of a larger volume of data) and b) the adaptation of the infrastructure to use new hardware/software, are treated in the document.

In the following sections, the infrastructure's enhancements are explained in section 1.1. Section 1.2 provides an update of the sources used towards providing complete news accessibility to the project. Section 1.3 describes the multilingual extensions to the preprocessing pipeline (developed in collaboration with the XLike project) and section 1.4 describes the improved monitoring toolchain. Finally, we give updated statistics on the data available through newsfeed.

1.1 System Performance

This section shows some performance improvements to the newsfeed data collection pipeline which have been made since M18 when it was delivered as D1.3.2 "Early prototype of data infrastructure" [1]. By then, the first version of the Newsfeed service had a median time of about four hours from being published to being discovered by the system and this delay in time has been decreased.

The system has a configuration parameter describing the desired latency for each RSS source feed but it had no effect since the system where the newsfeed was running was too slow to process the feeds at the desired rate, and therefore leading to allocate all the feeds in the ready queue. A side effect of this drawback is that the prioritization capabilities based on latency configuration are lost.

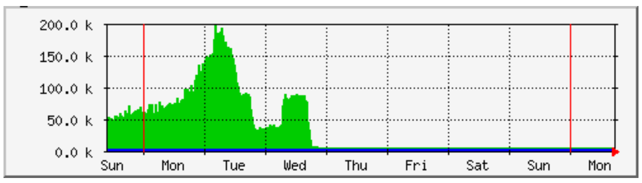
An analysis of the database access patterns has shown that the bottleneck was due to the part of system that matches newly discovered URLs with the ones which are already in the database, including redirect aliases. Revisiting an RSS feed contained, on average, only 4.2% of new URLs whereas the rest of them were either duplicates from previous visits of the feed or were received from other feeds with overlapping coverage areas. For example, this happen whenever a single news site provides multiple feeds posting the same link in more than one, or also sometimes due to news aggregation sites. Also, a significant problem occurred with some news sites that publish multiple years' worth of their news articles in a single RSS feed, with some reaching over 50.000 URLs per RSS download.

Instead of increasing the revisit times for most of the feeds, which would improve the situation with regard to the revisit times of high priority feeds, but wouldn't have changed median time discovery process, we extended the database tablespace with a pair of SSD drives and created a new set of hash value based indexes on the URL list, partitioned by published date month value. Then, the first check only tests for matching articles in the last two months, and only if that fails, it checks the entire database. This improvement has decreased the database I/O load to acceptable levels, and decreased the median article discovery time to approximately two hours.

Anther bottleneck was the RSS downloading and XML parsing process. The first version of the RSS feed monitoring stage of the pipeline was implemented as a single-thread process using greenlet routine system which provides parallelization and simplifies the database locking and consistency management. It was found out that the interplay between delays caused by CPU-bound parsing of large RSS XMLs and database query submission latencies caused by row locking turned out into large pipeline stalls.

To overcome this problem the database was refactored so that all of the shared data structures could be used by multiple RSS feed monitoring processes at the same time, and split the download and parsing of feeds between multiple processes based on the hash value of the internal ID of the feed. This modification allows the use of database sharding in the future whenever the I/O load exceeds the capabilities of a single database server.

Currently eight processes are being used to parallelize the feed processing, however only about three CPUs worth of processing power are used at any time. After processing the backlog, median latency of feed revisit decreased to within 10% of the set point. Figures 1 and 2 show the obtained results at the time of applying the improvements implemented. Figure 1 shows the decrease of feed revisit latency in seconds and Figure 2 the relative percentage to the desired value at the time of implementing the improvements. Figure 3 and Figure 4 also show the typical values throughout one day at the time of writing, in the same units, for a median latency of 1230 seconds and 114% relative to setpoint -- we expect that we will be needed about 11 minutes to revisit a feed that is set to be monitored every 10 minutes. The blue line shows the median latency, and the green fill shows the average latency, which is much larger due to inclusion of feeds that are only revisited once every day or less.



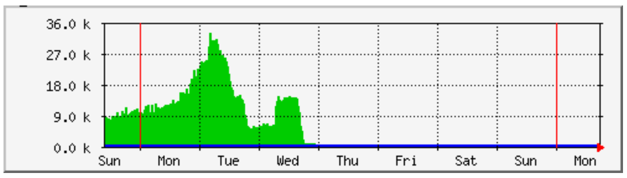
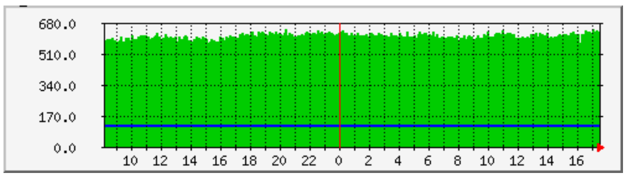


Figure 1 Average Time Latency Decreases Due to Infrastructure Improvements.

Figure 2 Average Relative to the Desired Target Time Latency Decreases Due to Infrastructure Improvements (Expressed in Percentage)



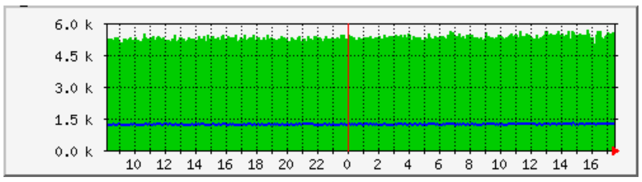


Figure 3 Median and Average Time Latency (Seconds) after Applying the Improvements.

Figure 4 Median and Average Time Latency (Seconds) Relative to Desired Target (Expressed in Percentage).

The rest of the processing pipeline does not have any bottleneck, so the system can easily be scaled to approximately an order of magnitude more of sources by using the same hardware and software. Figure 5 shows the number of articles found and processed during the time span of one year. The large spike corresponds to an one-off inclusion of a large number of additional feeds and showcases the maximum throughput of the pipeline. The long-term average rate is about 10000 articles per hour, and the current rate, using more data sources, is about twice that.

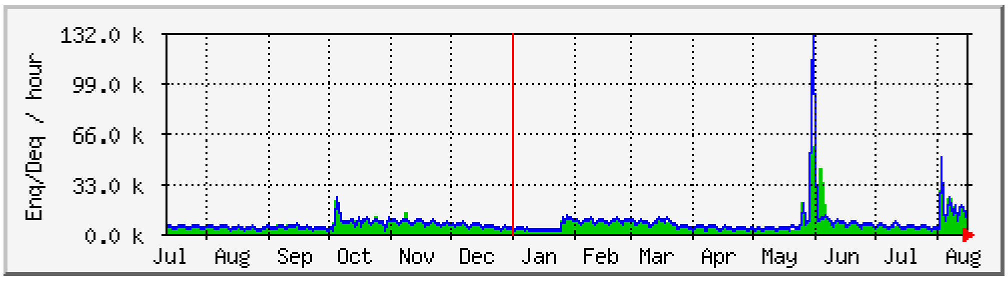


Figure 5 Number of Crawl and Processed News Articles by the JSI Newsfeed System from July 2012 to August 2013.

1.2 Additional Sources

This section describes the new added sources to the newsfeed[1] component for the RENDER project. Since the initial prototype, the Newsfeed has been extended with the following data sources: a set of blogs, a subset of Twitter which is currently the best-known microblogging service, and multiple proprietary real-time data streams from Bloomberg and STA (Slovenian Press Agency).

In the next we describe the characteristics of the mentioned newly incorporated data sources and the mapping between the metadata schemas involved.

1.2.1 Twitter

Twitter[2] is a microblogging service and its users are from most of the countries of the world publishing content in all major languages. Twitter offers a free uniform 1% sample of the posted messages as a real-time stream amounting roughly to four million messages per day. We have incorporated that stream into our RENDER pipeline and present it to the end users together with the other data sources following a unified data schema.

Regarding the accessibility we have used Twitter's REST API (version 1.1)[3] based on OAuth authentication for accessing to the twitter stream.

Twitter's schema consists of a number of different types of events. As most of them relate to data about the evolution of the social network rather than to content itself, we have ignored many of them as: 'friends', 'event', 'for_user', 'control' and 'limit', therefore this type of data does not go into the processing stage. On the other hand, for events of type 'text', e.g. the ones which actually contain new tweets we have performed the following schema mapping:

- ID: we form our internal ID simply by adding "tw-" at the beginning of the Twitter's tweet ID. This will ensure the traceability of data.
- URL: we reconstruct the HTTP URL of the tweet from its metadata. This metadata encodes, in an interpretable way, both the author's username and the tweet ID. This allows us to keep track of the user which creates that tweet without changing the schema towards Twitter specifics.
- Source coordinates: The API object contains the coordinates (longitude and latitude) of the place where the tweet was published in the property *tweet.coordinates.coordinates* (this is not always known).

[1] http://newsfeed.ijs.si
[2] https://twitter.com/
[3] https://dev.twitter.com/docs/api

- Story coordinates: The API object may also contain annotations denoting places that are being talked about in the tweet. These annotations are stored in the API object's *tweet.place* and each element representing a place is defined by the *tweet.place.bounding_box.coordinates* property. From a practical point of view this is a list of latitude-longitude pairs that delimit the area being discussed in the tweet. As our current schema only models points we converted the bounding box by averaging out all of its border's coordinates.

The full schema for describing a tweet within twitter API is described and can be consulted on [4].

All Twitter content is already included in the final aggregated stream of web content. It is distinguishable, among other above mentioned ways, by its hostname tag ("twitter.com") and is treated differently due to its relatively high data volume. At now, it currently bypasses the deep linguistic analysis and it is plugged directly into the content caching and distribution service (see Figure 1 in D1.3.2 [1]).

1.2.2 Blogs

To expand the systems coverage to blogs, we first collected various public lists of blog addresses and descriptions, such as the popular technorati.com website. In the first stage of collection, we acquired approximately 1.5 million addresses. After checking each address for the age of latest entries, we discovered that most of them either do not exist anymore (about 1 million), or haven't been updated in over half a year (another 200.000). Removing duplicates and technically problematic addresses left us with about 120.000 working blogs; 30.000 of those were since disabled for causing download problems. Figure 6 shows -- on a logarithmic scale -- the distribution of the number of articles found on currently active blogs in the last 3 months.

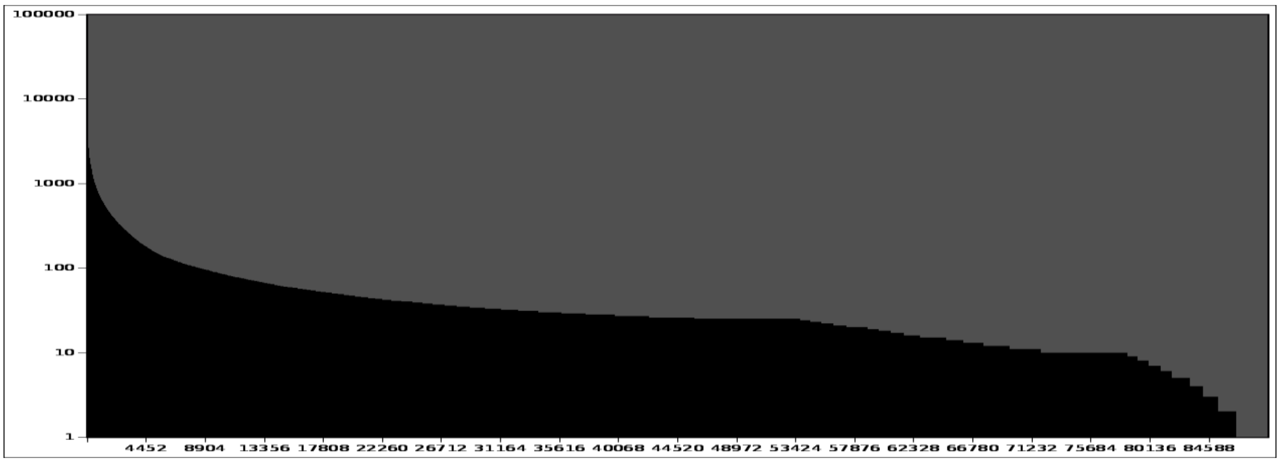


Figure 6. Number of New Blog Entries Found on Blog Sites Between June and August 2013. Y Axis is Logarithmic.

Inclusion of these blogs gives us an additional 50.000 – 100.000 new posts per day. Table 1 shows the distribution of languages used in our sample of the blogosphere.

Table 1 Distribution of Most Represented Languages in our Blog Dataset

Language	%	Language	%	Language	%
English	79.17	Dutch	0.46	Swedish	0.11
Spanish	3.50	Chinese	0.39	Slovenian	0.10

[4] https://dev.twitter.com/docs/platform-objects/tweets

German	2.31	Greek	0.38	Korean	0.09
Indonesian	2.00	Turkish	0.37	Sundanese	0.08
Japanese	1.93	Polish	0.30	Danish	0.08
Italian	1.88	Czech	0.18	Hungarian	0.08
Portuguese	1.46	Hindi	0.17	Hebrew	0.08
French	1.42	Thai	0.15	Bengali	0.07
Russian	0.92	Vietnamese	0.15	Slovak	0.07
Malay	0.70	Bulgarian	0.14	Catalan	0.06
Romanian	0.61	Tamil	0.12	Arabic	0.05

The English language is overrepresented in the dataset because of the way we collected the list of blogs. Due to the small amount of useful blog sites and the language disparity, we have investigated the possibility of using so called “blogrolls” -- lists of blogs that a given blog author recommends to his readers – as an additional input to the system. We have been able to extract some new addresses, however a better heuristic to distinguish blogrolls from other links will have to be developed before this approach can be automated.

Three new potential bottlenecks were identified after we started downloading blog posts. More than 10.000 blogs use an external service – feedburner.com – to provide an RSS feed for them, instead of providing one on their site. This limits the rate with which we can look for new posts on those blogs, as we try to avoid overloading RSS servers. There does not seem to be a way around this limitation for now except for artificially maintaining a low scan and download rate for these blogs.

Due to the increased number of links discovered in RSS feeds we had to implement a fast filtering solution to determine if a link has already been seen, in order to prevent the load on the database server to rise unnecessarily. We used a Bloom filter, which is a data structure for quickly determining the membership of an object in a set. Since the filter uses a probabilistic approach, some errors are possible: there is a small probability that the link gets rejected even if the system hasn't seen it before. With the current settings, this probability is about 1 in 10 million, which we consider acceptable.

The third potential bottleneck is parsing of the XML structure of RSS feeds, as we need to process 250 large XML files per second in order to achieve the desired scan rate. Some parallelization is still possible as not all of the CPUs have been used so far; however we are replacing the parsing library with an alternative solution. So far, all of the high-level parsing has been done in python, while the replacement is partially written in C. Tests show that we get an order of magnitude of a speed-up, however the replacement library does not support all of the functionality the current one does, so it needs to be adapted first.

1.2.3 Proprietary Feeds

In collaboration with the XLike FP7 project [2], multiple additional feeds with proprietary data formats have been added to newsfeed. As the data in those feeds is subject to strict copyright we were not able to release it to the RENDER consortium or even wider. We therefore only mention this extension in the passing, as a testimony to newsfeed’s modularity and extensibility.
Streams from the following news agencies have been added: Bloomberg, DPA, ANSA, AFP, AFP (English version), AP, APA, TANJUG, HINA, DPA (English version), ITAR-TASS, MTI, TAS.

1.3 Multilingual Linguistic Analysis and Annotation

Another newsfeed service extension pertinent to RENDER is the inclusion of new languages in the linguistic preprocessing and annotation pipeline. While this extension was supported primarily in the scope of the XLike FP7 project [2] and happened too late for its results to be currently used in downstream RENDER applications, it is directly relevant as it shows the possibility of extending RENDER or RENDER-like tools to a larger number of languages.

As of D1.3.2 [1], the newsfeed provided preprocessing of English articles using the Enrycher[5] platform. Since then this process of enrichment has been extended to several other languages (es, en, sl, zh, ca, de), supporting the following analyses: tokenization, lemmatization, POS analysis, entity-extraction and disambiguation.

The services are provided in part by JSI and in part by other XLike partners.

1.4 Story-level Article Clustering

The metadata provided by the newsfeed has also been extended by a service that performs story link detection.

The service maintains the centroids (in the high-dimensional bag-of-words space) of several thousand clusters using a dynamic proximity search data structure. Each new article is assigned to the cluster with the nearest centroid. If the resulting cluster is large enough, it is periodically considered for splitting into two subclusters using bisecting k-means; the decision on whether to accept the split or not is based on a Bayesian information criterion).

Individual articles' weight/contribution to the centroid is attenuated exponentially to prevent old stories from lingering in the system for too long. Overly old articles are discarded. Periodically, we examine pairs of similar clusters and consider merging them. A combination of cosine distance and Lughofer's ellipsoid-overlap criterion is used to determine whether to perform the merge.

The service has a push API to keep subscribers updated about cluster membership changes. A static snapshot of cluster membership information is also included in the newsfeed.

1.5 Improved Monitoring

Because of the increasing number of software components involved in the newsfeed, we have identified the need to take a more principled approach to monitoring the operation of the system. We have several monitoring services in place.

- MRTG[6], a SNMP-based graphing solution. We used this before; however, it does not easily scale across multiple computers. We continue to use MRTG for monitoring primarily the database server and have since extended it with several new graphs.
- In parallel, we have created a small network-based event aggregation service. It aggregates events (e.g. "article processed") from various newsfeed components, aggregates them over time and pushes the time-aggregated values to Leftronic[7]. See Figure 7 for a sample output.

[5] http://enrycher.ijs.si/
[6] http://oss.oetiker.ch/mrtg/
[7] http://www.leftronic.com

- Some of the services (e.g. the Google News crawler and the showcase data indexer) implement their own, internal monitoring mechanisms and trigger e-mail warnings to notify administrators of would-be problems in the pipeline.

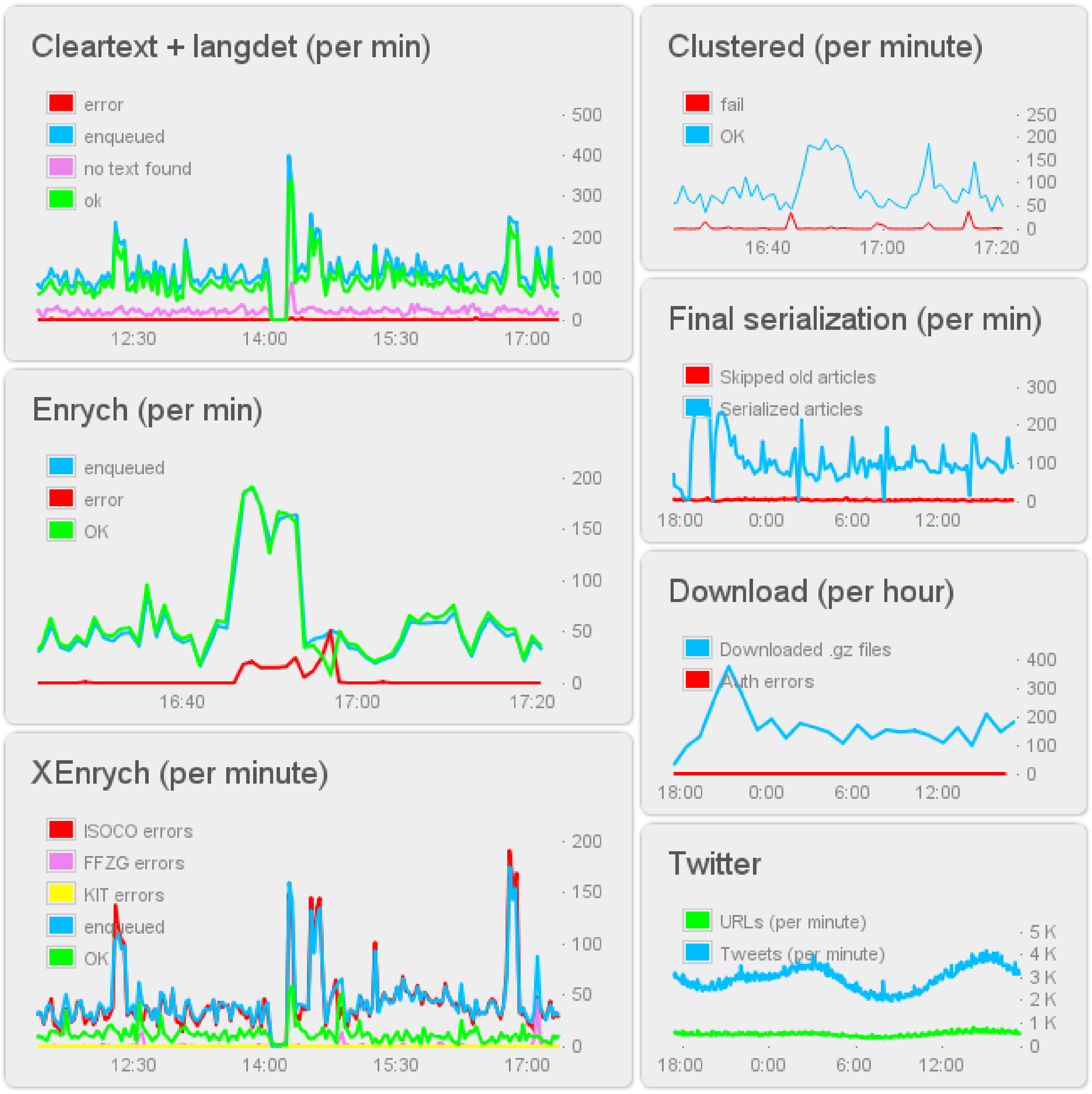


Figure 7 A Sample View of the Leftronic Monitoring Tool.

1.6 Statistics

This section provides updates on statistics already provided in D1.3.2 [1].

Sources

We have increased the number of known RSS feeds fivefold, to roughly one million. Of those, roughly 100000 are considered high enough quality and are being actively monitored.

Besides those, Twitter and blogs have been added as described in Section 1.2.

Language distribution

At the time of D1.3.2 [1], automatic language detection was only partially supported. We are now able to give more accurate estimates of data volume.

There are roughly 50 languages with at least 10 daily news articles. The top languages are:

Language ISO code	**Frequency in a random sample of 100000 articles**
eng	62570
deu	7842
spa	6537
zho	3967
fra	3494
rus	2656
nld	1635
por	1184
ita	1176
ell	1004
ind	710
hrv	556
slv	523
ces	509
tur	476
swe	469
jpn	460
ron	456
ara	384
pol	362
kor	305
msa	250
dan	245
fin	233
ukr	209
cat	196
slk	195
nob	183
hun	174
heb	167
lav	159
srp	120

Data volume

The crawler downloads 0.1-0.2 million articles per day from across 120000 different websites. The current archive contains about 100 million articles and begins in May 2008.

Responsiveness

We poll the RSS feeds and blogs at varying time intervals from 5 minutes to 12 hours depending on the feed's past activity. Google News is crawled every two hours. News agencies' data is either actively pushed to us or crawled every 30 minutes.

Based on articles where the RSS feed provides the original time of publication, we estimate 70% of articles are fully processed by our pipeline within 3 hours of being published, and 90% are processed within 12 hours.

References

[1] RENDER deliverable D1.3.2 “Early prototype of data infrastructure”.

[2] XLike project homepage: http://www.xlike.org/