xLiMe
FP7-ICT-2013-10
ICT-2013.4.1
Contract no.: 611346
www.xlime.eu

Deliverable D7.1.1

Early Benchmarking Report

Editor:	Ronald Denaux, ISOCO
Author(s):	Nuria García, ISOCO; Ronald Denaux, ISOCO; Andreas Thalhammer, KIT; Aditya Mogadala, KIT; Eduardo Torres Schumann, VICO; Joerg Schindler, Zattoo; Blaz Novak, JSI; Dubravko Culibrk, University of Trento.
Deliverable Nature:	Report (R)
Dissemination Level: (Confidentiality)	Confidential (CO)
Contractual Delivery Date:	M18 – 30 April 2015
Actual Delivery Date:	M18 – 30 April 2015
Suggested Readers:	All partners of the xLiMe project consortium and end-users
Version:	1.0
Keywords:	Benchmarking, evaluation, statistics, performance, scalability, accuracy, data processing, data annotators, data providers

Disclaimer

Full Project Title:	xLiMe – crossLingual crossMedia knowledge extraction
Short Project Title:	xLiMe
Number and Title of Work Package:	WP7 Use Cases and Evaluation
Document Title:	D7.1.1 – Early Benchmarking Report
Editor:	Ronald Denaux, ISOCO
Work Package Leader:	Ronald Denaux, ISOCO

Executive Summary

This document gives a report on the evaluation of different processing methods and technologies developed during the Y1 of the project regarding its scale and performance. The platform developed this year belongs to WPs that work with use cases and early prototype specifications (WP1, T1.4.1 – *Requirements for Early Prototype* and WP6, T6.2.1 – *Early Prototype*), and data processing technologies deployed (WP1, T1.2 – *Prototype of Data Processing Infrastructure* and WP6, T6.1 – *Toolkit Architecture Specifications*).

In this deliverable we provide results of benchmark tests for (1) use cases data providers, (2) methods described in T6.2.1, which are developed in WP2 and WP3 regarding cross-lingual converters and annotators tools (T2.1.1, T2.2.1, T2.3.1, T3.1.1, T3.2.1, T3.3.1), and (3) performance of data processing technologies employed (Apache Kafka, MongoDB, Virtuoso Server).

This document is the first of two (T7.1.1 Y2 and T7.1.2 Y3 – *Final Benchmarking Report*) that are associated with benchmarking the methods and tools developed within xLiMe project. Here we show the evaluation of these early prototypes and uses cases approaches, analysing the weak aspects in order to improve the performance and accuracy.

Table of Contents

List of Figures

List of Tables

Abbreviations

D	Deliverable
xLiMe	Cross-lingual Cross-media Knowledge Extraction
WP	Work Package
DT	Deutsche Telekom
OCR	Optical Character Recognition
HTTP	HyperText Transfer Protocol
HDS	HTTP Dynamic Streaming
HLS	HTTP Live Streaming
RSS	Really Simple Syndication
HTML	HyperText Markup Language
XML	eXtensible Markup Language
CPU	Central Processing Unit
HD	Hard Disk
GPU	Graphics Processing Unit
WER	Word Error Rate
POS	Part of Speech
ASR	Automatic Speech Recognition

1 Introduction

In this document we will present the xLiMe Early Benchmarking Report. This report provides the evaluations of the cross-lingual cross-media components developed within the first year of the project. Also the report contains the evaluation of the data processing and storage tools employed in order to manage the information. This document is the first of two (T7.1.1 Y2 and T7.1.2 Y3) that are associated with benchmarking the methods and tools developed within xLiMe project. This deliverable covers an initial benchmarking of the methods and tools chosen, exposing its efficiency and viability and highlighting the features to improve in order to get higher performance. For different methods different evaluation outcomes are foreseen. We will show statistics, numbers, charts, tables, comparatives, etc. to prove that the system is capable of supporting the near real-time management, storage, filtering, querying and streaming data efficiently.

1.1 Relation to Other Work Packages and Deliverables

The xLiMe Early Benchmarking Report described in this deliverable represents the initial evaluation of the services presented in D6.2.1 Early Prototype. The early benchmarking is strongly linked with this document (D6.2.1[2]) and its associated deliverables as the xLiMe toolkit architecture described in D6.1[1] and the prototype of the data processing presented in D1.2[5]. All of them describe the specifications and implementations of the main platform used to manage the generated data. Specifically, the early prototype includes:

- Overview of the implemented xLiMe system at the end of the first year
- Description in detail of data providers, use cases and multimedia annotation services
- Description of the data processing tools as Apache Kafka, MongoDB and Virtuoso
- First approach to evaluation with useful values as channels or items processed, data latency, etc.

The Early Benchmarking Report described in this document covers the evaluation of the methods shown in the early prototype, including text extraction (WP2), annotation (WP3), semantic integration (WP4), analytics services (WP5) and use cases prototyping (WP7). All these components are presented in D6.2.1 – Early Prototype, and link with the following deliverables:

- D2.1.1 Early Speech to Text Prototype
- D2.2.1 Early Text from Video Prototype
- D2.3.1 Early Text from Social Media Prototype
- D3.1.1 Early Prototype for Audio Annotation
- D3.2.1 Early Prototype for Video Annotation
- D3.3.1 Early Prototype for Text Annotation
- D4.1 Statistical Content Linking Prototype
- D5.1.1 Early Semantic Search Prototype
- D7.2.1 Early Prototype and Validation Report SEARCH
- D7.3.1 Early Prototype and Validation Report MONITOR

1.2 Methodology to Generate and Catch Data

The process of collecting data benchmarking in order to describe the information into this deliverable has been made by every partner in charge of the different components, algorithms, technologies, etc. Next the data collected is grouped and ordered to expose the results and conclusions in the document.

To achieve the benchmarking report, the partners have monitored their components and tools during a period of time (last month before writing the deliverable) extracting and calculating different relevant information and measure indicators. These indicators have been detected previously and sent to the partners in order to they know which data save. Next step is sharing the benchmarking results returned to filter and include them in D7.1.1.

The methodology process of generate and catch benchmarking data is outlined in the next points (Figure 1):

1. Detecting measure indicators of the components to benchmarking
2. Sending the indicators to the partners
3. Partners monitor the evaluation results returned by component
4. Partners share the information of the evaluation
5. Management and visualization of the data into D7.1.1

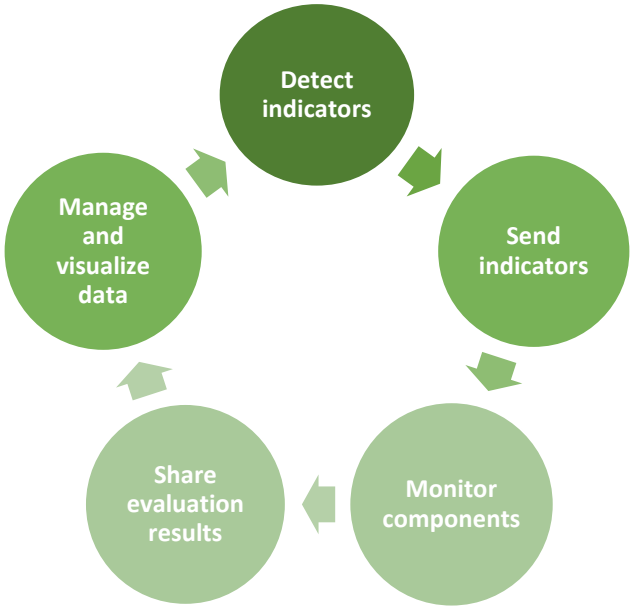


Figure 1: Methodology to generate and catch benchmarking data

1.3 Overview of Document

In this deliverable we present the evaluation of methods and tools developed within the first year of the xLiMe project giving results of benchmark tests for:

1. Specifications and statistics for data providers (VICO, Zattoo and JSI Newsfeed).
2. Methods developed in WP2, WP3, WP4 and WP5 regarding text extraction, data annotators, semantic integration and analytics services.
3. Performance of data processing technologies (Apache Kafka, MongoDB, Virtuoso).
4. Performance of xLiMe showcase and use cases prototypes.

2 Evaluation of Data Providers

2.1 Evaluation of VICO: Social Media Microposts

VICO[1] harvests large amounts of data from social media. The sources include large social networks like Twitter, Facebook, Google+, and YouTube, but also a broad spectrum of forums, blogs, review sites, and Q&A portals. The system already covers over 40 languages and provides a near real-time stream of public social-media microposts.

For the Early Prototype, we only use a subset of VICO's source data. In order to define this subset, VICO has implemented a subscription service which allows us to submit a list of keywords for different languages. This information is described in detail in D6.2.1.[2]

Below, we display data of historical values in March 2015 regarding number of microposts per day and its size in MB per day. (Table 1: VICO Microposts per day (March 2015)Table 1 and Figure 2)

Submission Day	Size (MB)	Number Microposts
08/03/2015	27.57	95196
09/03/2015	30.83	89119
10/03/2015	36.41	125358
11/03/2015	83.58	122252
12/03/2015	47.07	110469
13/03/2015	51.45	137488
14/03/2015	161.62	198638
15/03/2015	90.12	188783
16/03/2015	66.50	184278
17/03/2015	79.15	190217
18/03/2015	77.54	179948
19/03/2015	77.16	169309
20/03/2015	67.10	167262

Table 1: VICO Microposts per day (March 2015)

[1] http://www.vico-research.com/

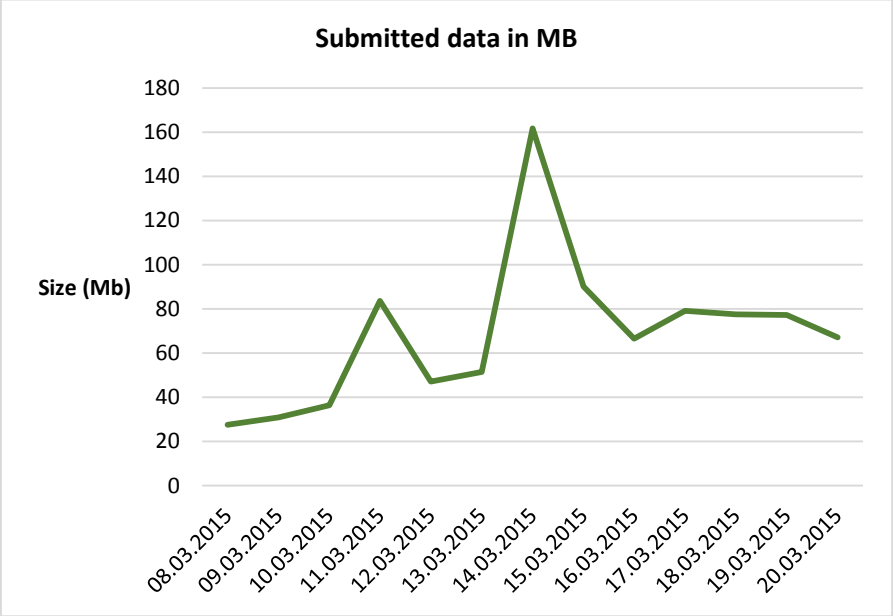


Figure 2: VICO - Evolution of submitted data during March 2015

There is a high peak the fourteenth of March as shown the Figure 2. The data submitted this day experimented a strong increase in number of microposts and specially regarding the size of the items. The rest of the days the variation was not as sharp.

The next data (Table 2) depicts the distribution per source for the VICO's microposts showing its quantities, average and deviation. Figure 3 and Figure 4 show this source distribution per number and per content size.

Source Type	Number Microposts	Size (MB)	AVG Size (Bytes)	STDDEV
Blog	18571	70,152,296	3777.52	6618.96
Forum	25452	102,108,082	4011.79	13270
Microblog	1620082	183,197,932	113.08	30.22
Social Network	246665	484,009,393	1962.21	3888.83
Video	47547	56,633,881	1191.11	1367.43

Table 2: VICO - Distribution per Source

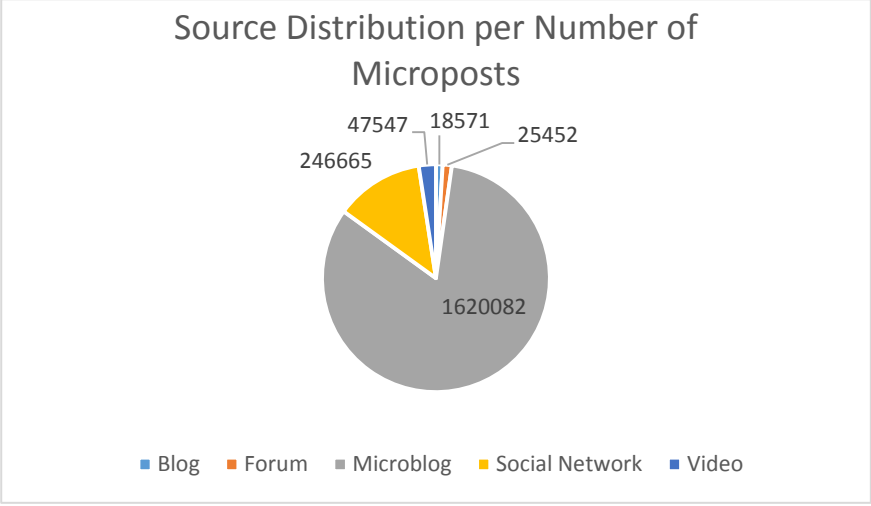


Figure 3: VICO - Source Distribution per Number of Microposts

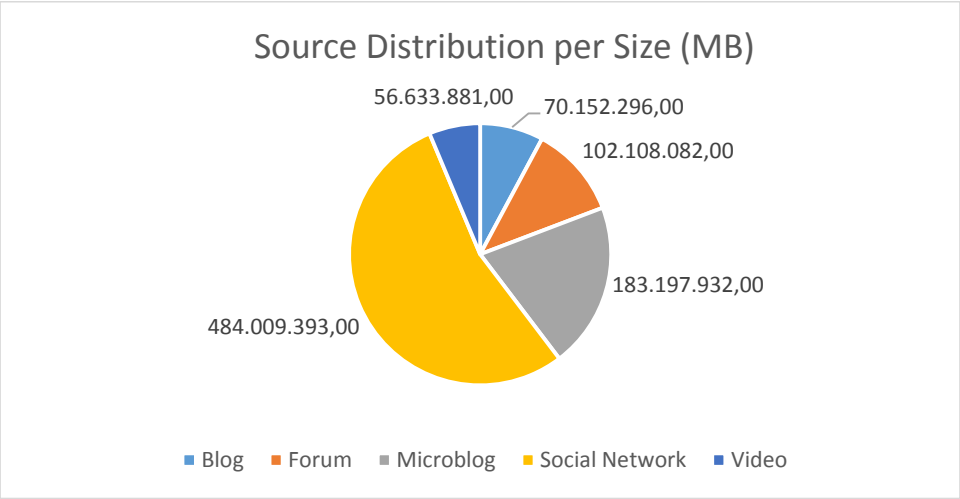


Figure 4: VICO - Source Distribution per Size

We can see that Microblog is the source with the largest number of microposts. However, Social Network source contains the highest occupancy regarding the content size. Forum source suffers the largest deviation in relation with number of microposts and its size in disk.

Below we show a chart with the historical data during March 2015 by source.

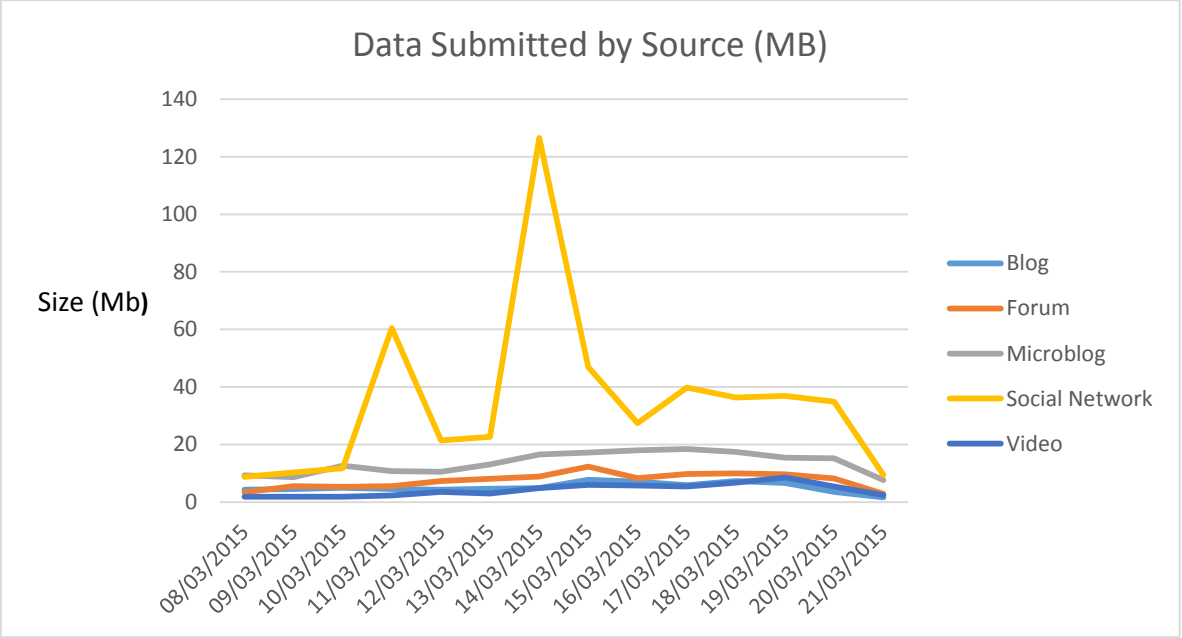


Figure 5: VICO - Data Submitted by Source

In this chart we can observe that the highest peak belongs to the Social Network source. This source has the biggest variations while the other sources have more stability.

Finally, we present data about the distribution of the VICO's microposts by language (Table 3) and a comparison between different languages by source (Figure 6).

Language	Number Microposts	Size (MB)
Catalan (ca)	539	0.11
German (de)	101616	79.16
English (en)	1637773	710.60
Spanish (es)	217632	106.08
Slovenian (sl)	757	0.13

Table 3: VICO - Distribution of Microposts by Language

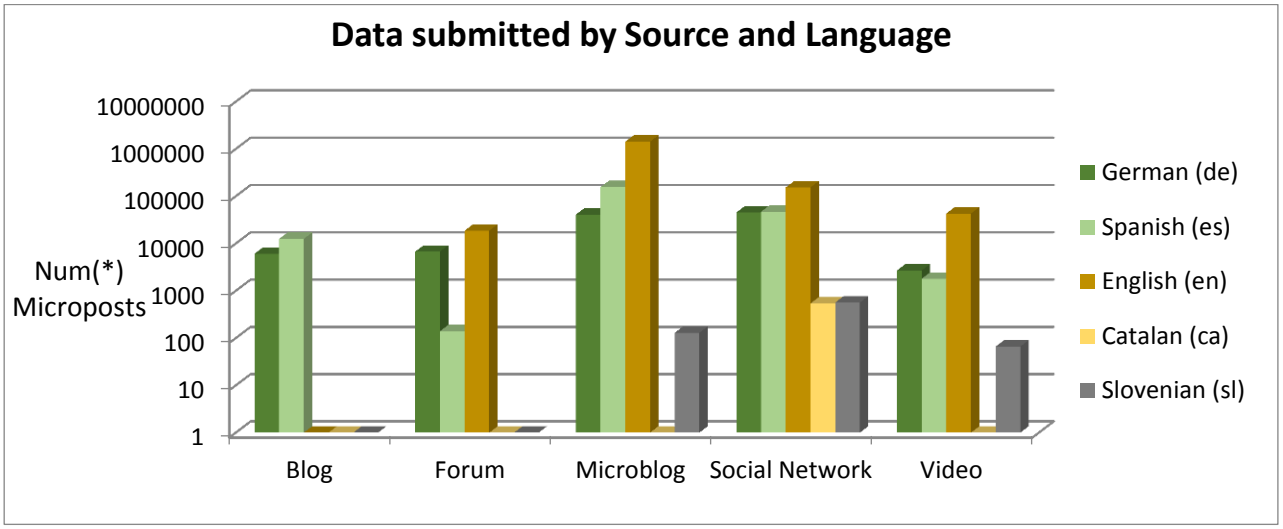


Figure 6: VICO - Distribution of microposts by Source and Language

(*) Data shown follow a logarithmic scale

VICO prototype in XLiMe project only manages five languages; German, Spanish, English, Catalan and Slovenian. The highest number of microposts is in English. However, German and Spanish languages are the only that appear in all the sources as Figure 6 shows. English is not present into Blog source of the data submitted and Slovenian and Catalan have minimum occurrences.

2.2 Evaluation of Zattoo: TV Streams and TV Programme Metadata

Zattoo[2] provides over 200 TV channels, streamed to consumers via an IP connection. The video streams are encoded in H.264 format. In order to gracefully handle differing bandwidths, dynamic bitrate streaming technology is used to ensure uninterrupted playback also when the available connection bandwidth is temporarily decreased. Depending on the target platform and application, dynamic bitrate streaming is implemented using HTTP Live Streaming (HLS), HTTP Dynamic Streaming (HDS) or Smooth Streaming technology. In the Table 4 we find the maximum qualities available to xLiMe. As we use adaptive streaming technology (HLS, HDS), also lower resolutions and bitrates are available.

Total bitrate	On-screen size (16:9)	Framerate	Video	Audio	Comment	No. of channels available to xLiMe	Channels used in Y1 prototype
1500 kbps	768x432	25	H.264 MP 3.0	AAC LC, stereo, 128 kbps	Standard Definition	107	13 (Al Jazeera, BBC World, Bloomberg, Canal 24 Horas, CNBC, CNN, Deutsche Welle, EuroNews, France24, N24, RAI News, SFR Info, Sky News)*
3000 kbps	1280x720	25	H.264 MP 3.1	AAC LC, stereo, 128 kbps	High Definition (3Mbps)	41	2 (tagesschau24, SRF Info)*

Table 4: Features Zattoo Data Provider

(*) **German**: N24, Tagesschau24, SRF Info, Deutsche Welle; **English**: BBC World, CNN, Bloomberg, CNBC, Sky News; **Spanish**: Canal 24 Horas; **French**: France24; **Italian:** RAI News

Delay

Zattoo Data Provider and its Use-case Prototype are closely related with the components *Speech-to-text* (D2.1.1)[3] and *Text-from-video* (D2.2.1)[4]. The delay in this showcase will came directly from the performance and data latency registered in these components. In xLiMe, the speech-to-text system is fed with 40 seconds-chunks of 128kbps audio, but currently many chunks are skipped in order to reduce the load on the speech-to-text system. Typical delay of the streams as xLiMe gets them from Zattoo against the original DVB-S signal is around 25..30s. However, to deal with the delay of the speech-to-text system, in the Y1 prototype, video on zattoo.xlime.eu is delayed artificially by some minutes ("close-to-realtime").

Information in detail about the performance in Speech-to-Text component can be found in its associated section 3.4.

[2] http://zattoo.com/es/

2.3 Evaluation of JSI Newsfeed: Web-Based News Articles

JSI Newsfeed[3] is a clean, continuous, real-time aggregated stream of semantically enriched news articles from RSS-enabled sites across the world.

Newsfeed pipeline performs the following main steps:

1. Periodically crawl a list of RSS feeds and obtain links to news articles
2. Download the articles, taking care not to overload any of the hosting servers
3. Parse each article to obtain
    a. Potential new RSS sources mentioned in the HTML, to be used in step (1)
    b. Cleartext version of the article body
4. Process articles with Enrycher[4] (English and Slovene only)
5. Expose two streams of news articles (cleartext and Enrycher-processed) to end users.

This news streaming tool is able to manage **470,000 articles per day**, 380,000k on weekends and holidays. The service has a rough language distribution, depends on stream selection (public news, blogs, PR, etc.). The statistics on this feature are as follows (Figure 7):

- English: 62% +- 10%
- Spanish: 10% +- 3%
- German: 4% +- 1%
- Portuguese: 2.8% (standard deviation too big to be useful)
- Russian: 2.4%
- French, Arabic, Chinese, Greek, Italian: between 2% and 1%

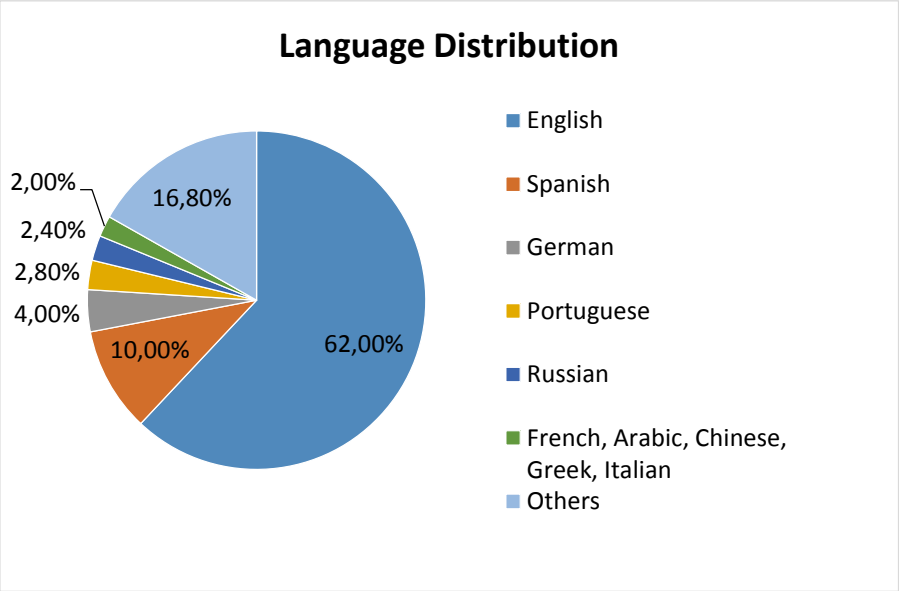


Figure 7: Newsfeed Language Distribution

In the Table 5 we show the mean and standard deviation of the size in kb per item.

[3] http://newsfeed.ijs.si/
[4] http://enrycher.ijs.si/

	Avg	Stddev
Extracted plaintext	2140 characters	2260 characters
Compact internal representation after annotation	140KB	120KB
XML serialization incurs additional 10s of KB		

Table 5: Average and standard deviation of size in kb per item

The channels processed are **1.2M various feeds** (almost all of them RSS, some links are extracted from Twitter), **110k** 'mainstream news' content and **40k** bring most of the 'mainstream news' content.

Regarding data latency statistics, the pipeline latency (from discovery to output ready for consumption) is approximately about 3-8 seconds. The article discovery latency is configurable per feed, depending on feed activity. The minimum set at 5 minutes for 4000 mainstream news, average 1h across all feeds. Next goal is reducing most of relevant feeds to below 3 minutes. The average data downloaded per feed check is 16kb, standard deviation 40kb, and with 25-75 mbps bandwidth used for new URL discovery.

Newsfeed service is running under the follows machine specifications (Table 6):

Hardware	Feature	Value
CPU 1 (feed monitoring and parsing)	Speed	2.66 GHz
	Vendor	Xeon X7460
	Total	16 cores
	Architecture	X86_64
CPU 2 (download)	Speed, Vendor, Architecture	Same values CPU1
	Total	2 cores
Memory	Capacity	25GB
Database drive array	Load	Constant 440 tps
	Read	10MB/s (mostly random access)
	Write	½ MB/s

Table 6: Newsfeed Machine Specifications

3 Evaluation of Data Annotators

In this section we evaluate the behavior of Data Annotators components. For this, we describe the next relevant features if these data are available:

- Machine Specifications: CPU, Memory, cores, speed, HD, etc.
- Resources used by component: per unit of data (e.g. millions of tweets)
- Throughput: unit of data per second (e.g. kb/s)
- Configuration options: e.g. fuzzy or full matching of entity names, indexing of full text in Virtuoso, etc.
- Slack: how much more could be handled? Or bottleneck (options to alleviate this, cost estimate to ramp up capacity, etc.)
- Precision and recall statistics

3.1 Named Entity Annotation for Social Media Microposts Evaluation

Named Entity Annotation component provides a tool for the annotations of unstructured multi-lingual texts. The input is the social media microposts extracted from VICO. The algorithm used is the KIT Entity Annotator method, described in detail in D3.3.1.

The specifications and performance data for this component are as follows:

Machine Specifications
Below, Table 7 shows the machine specifications.

Hardware	Feature	Value
CPU	Speed	2.27 GHz
	Vendor	Intel(R) Xeon(R)
	Total	4 cores
	Architecture	X86_64
Memory	Capacity	16GB
Hard Disk Storage	Capacity	160GB

Table 7: Machine Specifications - Named Entity Annotation Microposts

Resources
The process of annotating and submitting the posts runs constantly and uses 220% of the CPU and 27% of main memory in average.

Throughput/Slack
The component is submitting 150,000 microposts per day, but only a portion of them is being annotated at the moment. The reason is that a document is delivered without annotation when the processing time exceeds more than 1 minute or the content length is more 1400 bytes. The length constraint affects an average of 9,000 microposts per day.

Configuration Options

- Code Configurations
  Cross-lingual semantic annotation supports both service-oriented and user-oriented interfaces for annotating text documents. This infrastructure is implemented using a client-server architecture with communication over HTTP using a XML schema defined in XLike[5] project. The server is a RESTful web service and the client user interface is implemented as Web Application. The system can easily be extended or adapted to switch out the server or client. In this way, it supports both service-oriented and user-oriented interfaces for annotating multilingual text documents and web pages across the boundaries of languages.

- Languages support
  12 languages (English, Spanish, German, Chinese, Slovene, Catalan, Russian, Basque, French, Italian, Portuguese, Serbian) using resources from Wikipedia and Linked Open Data (LOD).

- Dependencies
  Stanford Named Entity Recognizer (NER)[6] and graph-based disambiguation engine JUNG Framework[7] has been used.

3.2 Visual Object Type Recognition Evaluation

The requirements of the video annotation component are mainly derived from the VICO use case (cross-media brand and topic monitoring), where the component will be used to detect the appearance of brands in the video stream. Visual Object Type Recognition tool recognizes a variety of different objects in videos and images. The input is a video stream provided by Zattoo via the Zattoo API, and the algorithm used is described in D3.2.1. The annotations produced are locations in the video stream linked to synsets from Wordnet.

The specifications and performance data for this component are as follows:

Machine Specifications

Early Prototype for Video Annotation basically uses GPUs for processing, so the only requirement is that the machine is able to support an NVIDIA Tesla K40c GPU, which has the following specs (Table 8):

Hardware NVIDIA Tesla K40c GPU	Value
Stream Cores	2880
Max Memory Size	12GB
Memory Clock Speed	16GHz
Memory Interface	384-bit
Max Memory Bandwidth	288 (GB/sec)
Peak Double Precision floating point performance (GFLOP)	1.43 Tflops
Peak Single Precision floating point performance (GFLOP)	4.29 Tflops

Table 8: GPU Specifications - Video Annotation

[5] http://www.xlike.org/
[6] http://nlp.stanford.edu/software/CRF-NER.shtml
[7] http://jung.sourceforge.net/

Resources
The component uses 6GB of the GPU memory for our Deep Neural Nets. As we said, the only requirement is that the machine is able to support an NVIDIA Tesla K40c GPU.

Throughput
For the classification task, that we use the Deep Neural Nets for currently, the nets are able to process up to 250 256x256 (8 Kb) pixel images per second. (2000 kb/s)

Precision and Recall
Two different nets are used, one for the general object labelling and the other for DT (Deutsche Telekom) logo detection. As stated in D3.2.1 the general purpose classifier achieves 15.5% top 5 error rate (meaning that the right class is in the five top ranked labels 84.7% of time). For the DT logo, an own dataset has been created. The testing was done on a separate test set, containing 1646 images, 520 positives and 1125 negatives. The accuracy achieved was 99.5%.

3.3 Text from Video Evaluation

The requirements of the text from video component are mainly derived from the VICO use case (cross-media brand and topic monitoring), where the component will be used to detect the appearance of brand mentions in textual form, in the video stream. Text from Video is designed to detect and extract as much textual data as possible from the frame. The input is a video stream provided by Zattoo via the Zattoo API, and it is used an OCR (Optical Character Recognition) algorithm to convert text appearing in a video. This is described in detail in D2.1.1. The annotations produced are locations in the video stream linked to recognised words.

The specifications and performance data for this component are as follows:

Machine Specifications
Below, Table 9 shows the machine specifications.

Hardware	Feature	Value
CPU	Speed	2.40 GHz
	Vendor	Intel(R) Xeon(R) E5620
	Architecture	X86_64
Memory	Capacity	24GB

Table 9: Machine Specifications - Text from Video

Resources
The component is able to process 6 frames per second on this system. The system has 24 GB of memory, but the memory consumption is negligible and hard to evaluate. We can ensure that more than 1 GB of memory is not needed to be able to do this processing and annotating task.

Throughput/Slack
The performance is 6 frames (64Kb) per second (384 kb/s).
There is no really slack for this component, since the complexity of the algorithms that we run is limited by the resources. To achieve greater efficiency and slack the better possibility is increase the number of GPUs and/or the cluster of machines to do the OCR. Highlighting that a Tesla K40c GPU is priced at about $3,000, so this implies that the cost of ramping up can be excessive. Another strategy is basically to reduce the number of frames processed per channel and the complexity of our approaches. In the second year the

idea is to move on to a logo localization instead of the basic detection (classification) approach, which, in its current application, takes about 13 seconds per frame on a single Tesla GPU.

Precision and Recall
Two different testing methodologies are used to evaluate the text detection performance. The actual OCR performance is not evaluated since we use a well-known Tesseract OCR for final text extraction. When evaluated using a ground truth generator obtained from the relevant literature, the accuracy of text detection is 59%. We also evaluated on a dataset of frames extracted from new video streams for the purposes of xLiMe, and the accuracy was 30%, the recall 69%, while the precision was 30%.

3.4 Speech to Text Evaluation

Speech to Text component is a software system developed at JSI that wraps external speech recognition engines and provides a common infrastructure for speech transcription to the xLiMe project. Since the only data sources for the speech recognition service within the project are the Zattoo TV channels, we have implemented an entire pipeline that acts as a client to the Zattoo TV distribution cloud, from where it downloads audio and video content based on a configurable set of channels. This service uses two external APIs for converting speech to text, one from VecSys[8] and another one from PerVoice[9]. The positions in the audio streams are linked to transcriptions of the spoken audio. More details of this component can be found in D2.1.1.

Highlighting that Text from Speech and JSI Newsfeed use the same pipeline components (same instances), so performance information is shared and combined into these stats. Load caused by speech-to-text data is negligible compared to article processing.

The specifications and performance data for this component are as follows:

Machine Specifications
VecSys cloud servers use 4 cores of latest Intel CPUs to process a single audio stream in real time.

Throughput
A single 40s audio chunk is processed in 2min 20s, with standard deviation 62s on the VecSys cloud instance. Multiple instances can be used, but only during off-peak hours. Since processing of short chunks not can be parallelized, estimate of "4 cores for real-time" the numbers would be: 140s / 40s = 3.5. The estimation for every item is approximately 40Kb of size, so throughput would be 140Kb/s.

Slack
CPU is a hard bottleneck. To solve this, more number of CPUs would be needed. KIT provides virtual machines with AMD CPUs, but they are apparently much slower, about 5-11 minutes per single 40s chunk.
The cost is directly proportional to number of audio channels processed. The use cases of the project are intended to have 26 channels of audio, it means that around 96 Intel Xeon cores would be needed.

Precision and Recall
Recall varies widely between different content types, speakers, etc. According to manual evaluation on English, 20-75% word error rate on random sample of 130 40s chunks.

[8] http://www.bertin-it.com/vecsys-expert-en-technologies-vocales/
[9] http://www.pervoice.com/en/

3.5 Named Entity Annotation for Text Evaluation

Named entity recognition and annotation component provides a service to parse, analyse and extract different entities from texts. The tool creates both a vector-space-model representation of the text and annotations, namely named entities on normalized social media text (algorithm described in D2.3.1).

This component has the next pipeline:

1. Input data: Independent streams of text data, documents, from different sources.
2. Pre-processing: We apply a series of transformations to each document to prepare it for the output generation.
3. Output: We extract the desired information from each pre-processed document in order to generate the output.

The annotations produced are DMOZ[10] topic for the articles and named entities based on multilingual Wikipedia pages.

The specifications and performance data for this component are as follows:

Machine Specifications

Below, Table 10 shows the machine specifications.

System (Load split component-wise between)	Hardware	Feature	Value
System 1	CPU	Vendor Total	Xeon X7460 24 cores
	Memory	Capacity	192GB
	Hard Disk Storage	Capacity	3 x 6TB + 16 x 1TB + 2x 2TB + 3 x 136GB
System 2	CPU	Vendor Total	AMD Opteron 6200-series 64 cores
	Memory	Capacity	512GB
	Hard Disk Storage	Capacity	External SAS enclosure with 20+TB of space on 2TB SATA drives

Table 10: Machine Specifications Named Entity Annotation Texts

Resources

- XLike stage 1 -- tokenization, POS tagging for {english, spanish, catalan, german, chinese, slovene}
  - 48% of one core, 3.7GB memory used
- XLike stage 2 -- KIT wikifier
  - 10 - 40 cores used. Per article/text snippet processing time can range from 1s to 1minute
- XLike CCA stage
  - 2 cores used

[10] http://www.dmoz.org/

Throughput
The component has around 3-10 (average 5.6) text items per second (= news or ASR output text). Load varies by ~50% through day. We estimate every item is approximately 20Kb size, so throughput would be 112Kb/s.

Slack
The component has 30% slack for XLike wikifier to full load of the machine. The component is being reimplemented in C++ for performance. The early tests show over order of magnitude improvement in speed. It requires 22GB of RAM for languages in the "XLike stage 1" list.

3.6 Syntactic Annotations for News Articles Evaluation

This component extends the annotation pipeline used for named entity recognition described in section 3.5 by adding sentiment analysis and n-gram vectorization step to describe the original text. Details of this algorithm are described in D2.3.1.

The input of the service is a raw text from news articles and the annotations produced are sentiment analysis consisting of a polarity of the article and a confidence value, positions of the recognised named entities and vectorised n-grams of the article.

Machine Specifications

Hardware	Feature	Value
CPU	Speed Vendor Total	3.33 GHz Xeon W3680 12 threads
Memory	Capacity	24GB
Hard Disk Storage	Capacity	3x 3TB RAID

Table 11: Machine Specifications Syntactic Annotation

Resources
< 10% of a single CPU thread for all tasks, 5.5GB ram.

Throughput
The component has 5.6 newsfeed articles + <1 ASR text chunk + ~1 tweet/FB post/etc. per second all together, limited by supply side. We estimate every item is approximately 20Kb of size, so throughput would be 112Kb/s.

Slack
Additional CPU power for higher rate would be needed.

3.7 Cross-Media Recommendations Based on Statistical Linking Evaluation

Cross-Media recommendations are generated on the data stored in MongoDB repositories using retrieval based approach. Content similarity and recommendation algorithms are described in detail in D4.1.
Below, we list out the system configuration and other interesting components for storing and providing recommendations.

Machine Specifications
Table 12 depicts the machine specifications where the component is deployed.

Hardware	Feature	Value
CPU	Speed	2 GHz
	Vendor	AMD
	Total	8
	Virtualization	AMD-V
	L1d Cache	64k
	L1i Cache	64k
	L2 Cache	512k
	Byte Order	Little Endian
	Architecture	X86_64
Memory	Capacity	80GB
Hard Disk Storage	Vda1	1M
	Vda2	3T

Table 12: Machine Specifications - Cross Media Recommendations

Resources

The resources used by this hardware are as follows.

Storage:

- Social Media Content – 1.01 million + posts in past 7 days
  - Tweets
  - Facebook Posts
  - Forums
- TV Programs Information – 350 + news programs in past 3 days
  - Metadata
- News Articles – 40k + articles in past 3 days

Throughput

The **throughput** of the cross-media recommendations component is **232Kb/sec**. It is handled for processing incoming data from Apache Kafka to store in MongoDB.

Configuration Options

In the Table 13 we show the configuration options of the component.

Storage Configurations	MongoDB	Additional Security using Authentication mechanism
		Sharding support - default
		Running MongoDB as daemon
		External Log configuration to store MongoDB log
		Enabled Text Indexing for fast retrieval used for recommendations
Code Configurations	Python configuration	Libraries required for MongoDB access using python are installed
		Support for JSON object encoding is installed
		Python 2.7+ version is configured.
	Java Support	Conversion of TRiG to JSONLD is supported by Apache Jena libraries
Recommendation Configurations	No Fuzzy Matching	
	Key-Phrase are extracted from Large Texts	
	Retrieval of relevant information in the Text indexes of MongoDB is done by querying	
	Queries match relevant information in MongoDB by collection aggregation functions	

Table 13: Configuration Options - Cross Media Recommendations

Slack

Storage

- MongoDB
  - Highly scalable and is supported by sharding
  - Do not acquire any external cost
- Code
  - Efficiency of processing information can be improved by spawning multiple threads

Precision and Recall

It is necessary to create gold standard dataset to calculate precision and recall statistics for this component.

3.8 Comparison of Data Annotators

Regarding Data Annotators performance we can summarize throughput information in the Table 14. Here we can see the variability in performance for different components. Figure 8 depicts this data to show the comparison of the tools.

Data Annotator Component	Throughput
Named Entity Microposts	9000 microposts per day < 1400 bytes Low throughput Even estimating files of 20Kb, throughput ~ 2.1kb/s
Video Annotation	250 images, 256x256 (8 Kb) per second. (2000 kb/s)
Text from Video	6 frames (64Kb) per second (384 kb/s)
Speech to Text	3.5 items (~40kb) – 140Kb/s
Named Entity Newsfeed and Sentiment Annotation	5.6 items (~20Kb) – 112Kb/s
Cross-media Recommendations	232Kb/s

Table 14: Data Annotators Throughput

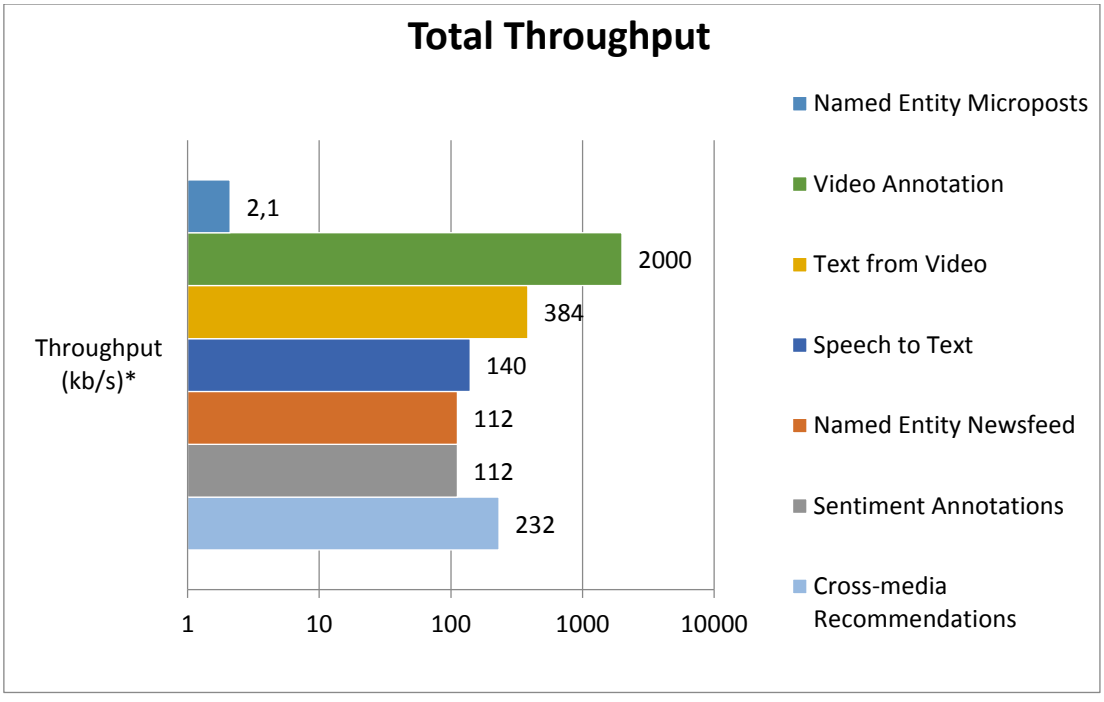


Figure 8: Total Throughput Data Annotators

(*) Data shown follow a logarithmic scale

Data collected show a high performance of the Video Annotation component. This tool leads in more than 1000 to the rest. The large variations are due to different machine specifications, resources available for each component and the overload of the different pipelines depending of the process of the component. Another main reason is because Video Annotation and Text from Video components have an input bigger than Text Retrieval components (video contains more information than text but, by the moment, we extract less data).

Named Entity Microposts component has very low throughput in comparison with the other tools. We should improve the performance of this component to match others in the future given that it is doing something similar to the Named Entity Annotation for Text service.

4 Evaluation of Data Processing Tools

4.1 Evaluation of Apache Kafka

Apache Kafka is a distributed publish/subscribe messaging system. It is fast (a single broker can manage hundreds of megabytes of reads and writes per second from thousands clients), scalable, durable, distributed by design, partitioned and replicated commit log. Kafka has messages marked by the offset in the log versus the ids in traditional broker systems, so this improves the performance, avoiding overhead.
In order to manage Kafka capacities of storage we have collected the data depicted in Table 15 and Table 16. Here we can check values in two different dates for the consumer *isoco-console-consumer-20140919*. The features shown are:

- Topic: Name of the topic
- Pid: Topic id
- Offset: Messages consumed
- LogSize: Total messages
- Lag: Difference between the end of the log and the consumer offset

Our Kafka structure has the following topics and partitions available: (**jsi-annotations**: 1 partition; **jsi-newsfeed**: 2 partitions; **socialmedia**: 2 partitions; **tv-metadata**: 4 partitions; **tv-ocr**: 2 partitions; **tv-visual**: 1 partition; **zattoo-asr**: 2 partitions; **zattoo-epg**: 2 partitions)

24th September 2014

Topic	Pid	Offset	LogSize	Lag
jsi-annotations	0	5052	5055	3
jsi-newsfeed	0	504698	516878	12180
jsi-newsfeed	1	504650	516830	12180
socialmedia	0	143185	151024	7839
socialmedia	1	142334	150174	7840
tv-metadata	0	18	24	6
tv-metadata	1	18	18	0
tv-metadata	2	24	37	13
tv-metadata	3	36	42	6
tv-ocr	0	452296	456662	4366
tv-ocr	1	452277	456643	4366
tv-visual	0	39225	47957	8732
zattoo-asr	0	15636	24161	8525
zattoo-asr	1	15621	24146	8525
zattoo-epg	0	7751	9207	1456
zattoo-epg	1	7728	9184	1456

Table 15: Storage Kafka 24th September 2014

10th March 2015

Topic	Pid	Offset	LogSize	Lag
jsi-annotations	0	5398	5398	0
jsi-newsfeed	0	504698	4074680	3569982
jsi-newsfeed	1	504650	4074633	3569983
socialmedia	0	143185	560170	416985
socialmedia	1	142334	558403	416069
tv-metadata	0	18	108	90
tv-metadata	1	18	138	120
tv-metadata	2	24	117	93
tv-metadata	3	36	132	96
tv-ocr	0	452296	460966	8670
tv-ocr	1	452277	460945	8668
tv-visual	0	39225	56563	17338
zattoo-asr	0	15636	289950	274314
zattoo-asr	1	15621	289918	274297
zattoo-epg	0	7751	75864	68113
zattoo-epg	1	7728	75822	68094

Table 16: Storage Kafka 10th March 2015

These data collected allow us to check the variations regarding the documents pushed by topic. We can see the differences in number of messages and information consumed between the two dates. Figure 9 depicts the increase of the logsize in this scenario.

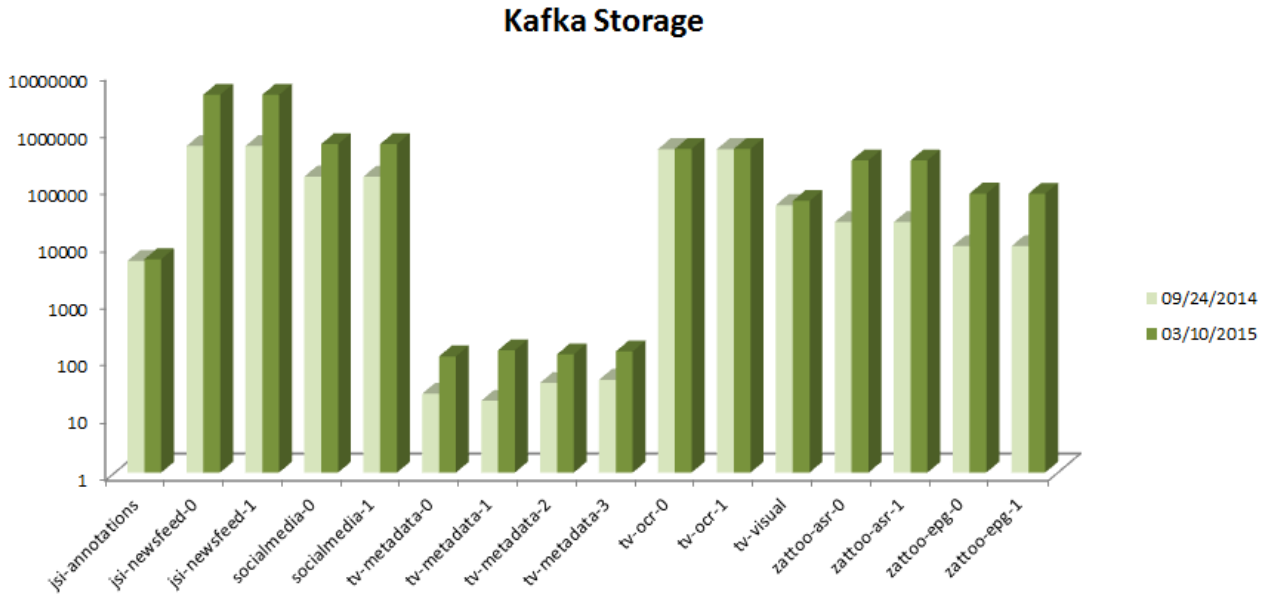


Figure 9: Increase of number of messages in Kafka

The storage in Kafka during the five months elapsed increases notably in several topics. For instance, *jsi-newsfeed* topic presents an increase of 3,557,800 of documents. The performance in Apache Kafka has not been reduced despite the huge amount of data added to the server and growing every day. So Kafka is working successfully under our demands of scalability and reliability.

Below, we provide some input on the Kafka benchmark. Both tests were executed with the configurations from the commands (e.g. 1000.000 messages) with the current Kafka client and server version 0.8.2.1.

Producer

- Command: *kafka-producer-perf-test.sh --topic test --broker-list xlime.kafka.host:9092 --messages 1000000 --message-size 1024 --show-detailed-stats*
- Results:

Start	End	Compression	Message Size	Batch Size	Data Sent (MB)	MB sec	Data Sent (nMsg*)	nMsg.sec
2015-03-25 14:27:10:610	2015-03-25 14:29:08:309	0	1024	200	976.56	8.2971	1000000	8496.2489

Table 17: Apache Kafka - Stats Producer

*nMsg: number of messages

Consumer

- Command: *kafka-consumer-perf-test.sh --topic test –zookeeper xlime.kafka.host:2181 --messages 1000000 --message-size 1024*
- Results:

Start	End	Fetch Size	Data Consumed (MB)	MB sec	Data Consumed (nMsg*)	nMseg.sec
2015-03-25 14:31:43:403	2015-03-25 14:33:18:443	1048576	976.5606	10.8459	1000000	11106.1750

Table 18: Apache Kafka - Stats Consumer

*nMsg: number of messages

4.2 Evaluation of CumulusRDF/Virtuoso Server

OpenLink Virtuoso is a triple store and database engine that combines the functionality of RDBMS, virtual database, RDF, XML, etc. in a single system. It's a universal server that implements multiple protocols. Our Virtuoso (version 7.1.0) is deployed on a Virtual Machine with 8 Cores at 2.0Ghz and 80GB of Main Memory.

There are several benchmarking of Virtuoso with multiple tests and configurations. One of the most recognized is the Berlin SPARQL Benchmark[11]. This source has a complete description of the performance of Virtuoso and compares its results with other triplestores such as BigData, BigOwlim or TDB. The webpage provide tests with different datasets and machine specifications. The results cover aspects regarding querying performance and loading times for each triplestore and comparing them. In the sections 4.4, 5.4 and 6.2 we can see benchmarking results for Virtuoso. Section 7 makes a comparison study between all triplestores analyzed.

4.3 Evaluation of MongoDB

MongoDB is a NoSQL database which supports document storage. It is schema-less and allows JSON documents having key and values. The advantage of MongoDB is that it is document-oriented. It also supports ad-hoc queries and can be used for indexing. Since it can be replicated, it provides high availability. The horizontal scaling of MongoDB is also possible using sharding. For accessing the data stored in the MongoDB, we can use many front-end programming languages.

MongoDB evaluation is as follows:

MongoDB Storage

- MongoDB Total Text Index Size – 18GB
- Size of Each Distributed Index – 2GB

Querying Performance

Documents Retrieved

- News Articles – 40k+ docs/ 3m 36 sec
- TV Metadata – 350+ docs / 0.145 sec
- Social Media – 10000+ docs / 4.12 sec

[11] http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/results/V7/

5 VICO: Use Case Monitor Evaluation[12]

VICO's XLiMe Use Case is a cross-lingual, cross-media brand monitoring and behaviour analysis. This means that the use case manages the monitoring of brands and public opinion, across languages, content types and content generation types to enable seamless trend prediction and issue management for organizations beyond local markets.

In this section, we provide a comparison of monitoring results of different media types (Social Media, TV, News), performed by analysts at VICO. We show a content evaluation and quantitative analysis for English documents. The datasets evaluated are:

- Social Media: Twitter post, forum post or comment, YouTube comment, etc.
- TV: speech-to-text transcription of a maximum of 40s speech snippet.
- News: news article

For the content evaluation VICO analysed the results of the monitoring of one topic and one brand, in this case the brand *Mercedes*. Random documents, distributed over all media types from within a defined time period was selected for the study. The system identifies salient categories and assigns them to each document, along with positive or negative sentiment if applicable. The categories are *buying*, *news/novelties*, *F1*, *tests*, *repair/spare parts*, *Black Friday/Cyber Monday* and the time period is *Nov 22 2014 – Nov 25 2014*.
Figure 10 depicts the results of this categorization for all media sources. We can see that the major data comes from the 'F1' group. Figure 11 show the percentages regarding different sources (TV and Social Media).

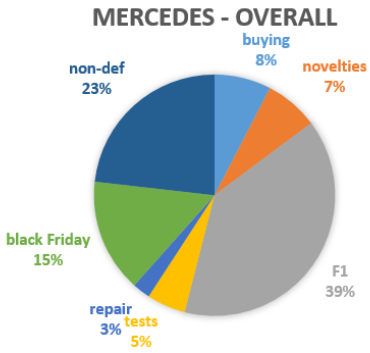


Figure 10: Overall categorization brand 'Mercedes'

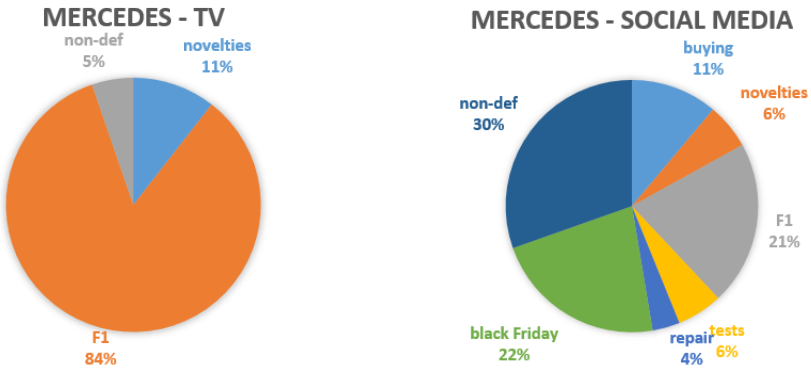
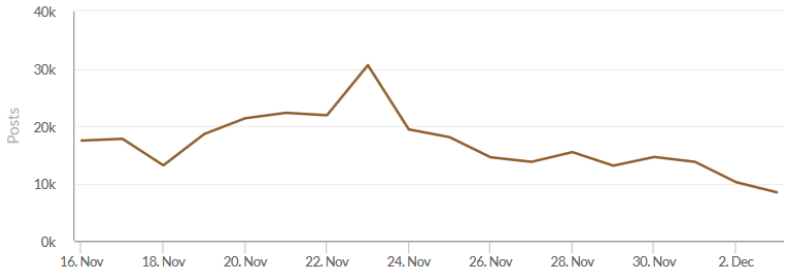


Figure 11: Categorization 'Mercedes' in TV and Social Media

[12] We only include VICO Use Case Evaluation. Zattoo Use Case Evaluation is not available yet due to the incomplete state of the prototype for collecting feedback. It will be described in next benchmarking deliverable 7.1.2

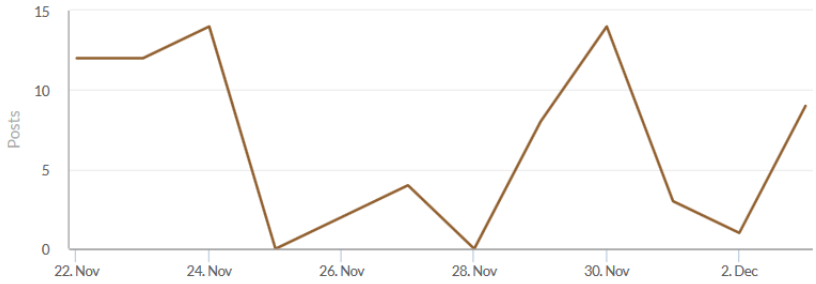
For quantitative analysis we provide some charts reflecting the size of posts about 'Mercedes' brand. Figure 12 represents the posts in social media type in the second fortnight of November. Here we can check a high peak during the day of the race.



Formula 1 Abu Dhabi Grand Prix Race: 23.Nov

Figure 12: 'Mercedes' in Social Media

Below, Figure 13 shows the same background for TV media source. This chart contains bigger variations around the nearest days of the race.



Formula 1 Abu Dhabi Grand Prix Race: 23.Nov

Figure 13: 'Mercedes' in TV

Finally, we show a comparison of the different brands in F1 and its occurrence.

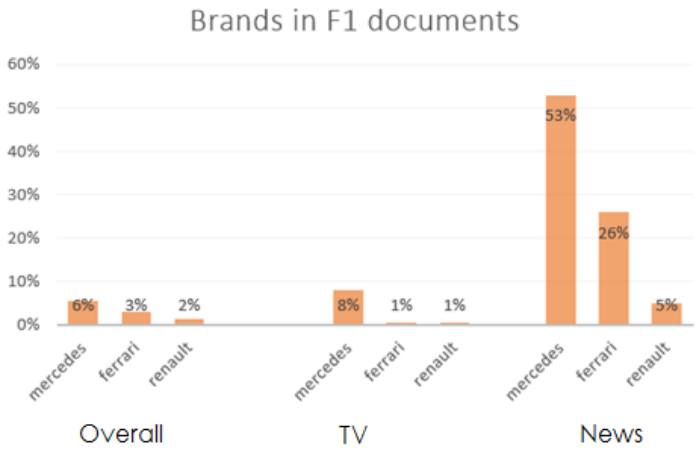


Figure 14: Comparison brands in F1

6 Conclusions

In this deliverable we have described the evaluation performed on methods and tools developed and described in D2.1.1, D2.2.1, D2.3.1, D3.1.1, D3.2.1, D3.3.1, D4.1, D7.2.1 and D7.3.1. Also, we show evaluations for Data Providers (VICO, Zattoo and JSI) and Data Processing Tools (Apache Kafka, Virtuoso and MongoDB).

In this comparison we identify the specifications and resources used and analyse the throughput and options to slack and improve the performance for the different components. We include some tables and figures which depict the data collected and reported, making easier to the reader understanding the results of the evaluation.

In the second year of the xLiMe project, we will add more data providers (ECONDA), the existing data providers will increase the data they provide (e.g. Zattoo will add subtitles to the video streams), and more annotation services will be included. In general we expect to improve the performance on the current annotation services and data processing tools. Next benchmarking report (D7.1.2 Y3 – *Final Benchmarking Report*) will describe the performance of these improvements and new features.

In the next benchmarking deliverable D7.1.2 we will also focus on analyse the scalability of the different components. It would be possible duplicate the throughput adding another server for each component? We should study the next possible options to act regarding this issue:

- In theory we could add a new server but we would need to make a dispatcher in order to can duplicate the throughput
- We have a component ready to duplication, we would only need to buy and configure a new server
- It is not possible parallelize the process because the algorithm needs to access to all input, so we not can distribute it in various servers.

As well, we should analyse the estimate cost of add new servers and the overhead of each copy. For instance, we comment into Text from Video Evaluation section concerns regarding the addition of new GPUs because of its high cost, an issue that would mean a limited margin of improvement through this solution.

Theoretically all components are reproducible but only Apache Kafka and MongoDB have this functionality ready. For the rest of components, we should add that dispatcher function to achieve performance improvements.

References

[1] xLiMe deliverable “D6.1 – Toolkit Architecture Specifications”

[2] xLiMe deliverable “D6.2.1 – Early Prototype”

[3] xLiMe deliverable “D2.1.1 – Early Speech to Text Prototype”

[4] xLiMe deliverable “D2.2.1 – Early Text from Video Prototype”

[5] xLiMe deliverable “D1.2 – Prototype of Data Processing Infrastructure”

