Embedded Human Computation for Knowledge Extraction and Evaluation (uComp)

D3.4: Distributed Knowledge Processing Framework

31 July 2014
Version: 1.0

Version History

Version	Date	Author	Comments
0.1	02/05/2014	A. Scharl	Initial Draft Document
0.2	11/05/2014	A. Scharl	Major Revision
0.3	15/05/2014	G. Wohlgenannt	Ontology Learning Section
0.4	24/07/2014	K. Bontcheva	Text Processing Section; Major Revision
1.0	31/07/2014	A. Scharl	Final Revision and Document Layout

Dissemination Level: PU – Public

uComp receives the funding support of EPSRC EP/K017896/1, FWF 1097-N23, and ANR-12-CHRI-0003-03, in the framework of the CHIST-ERA ERA-NET program line.

Table of Contents

Overview

The uComp project builds scalable, data--driven knowledge discovery and analytics applications that rely on flexible, fault-tolerant and iterative system architectures. A typical approach for such “big data” applications is to integrate the MapReduce implementation of the Hadoop Framework, and using a cloud computing layer to distribute computationally expensive processes. Despite the applicability of the batch-oriented MapReduce model for a wide range of problems, however, it should not be considered the silver bullet in every situation. After reviewing the initial plan of using Hadoop on top of a private cloud computing infrastructure in light of the specific requirements of the project, the consortium decided to adopt a more flexible approach to achieve the required level of scalability in the uComp processing pipeline.

A modular, container-based architecture that uses a combination of relational and NoSQL databases in conjunction with sharded indexing is ideally suited to process data streams (in contrast to batch-oriented Hadoop implementations), and to orchestrate a portfolio of services for the uComp analytic and knowledge processing tools - including text mining, knowledge extraction, pattern discovery, and advanced search.

The structure of this deliverable follows the resource-intensive elements of the uComp processing pipeline - including multilingual data acquisition, analytical and knowledge processing tools including ontology learning, and information retrieval.

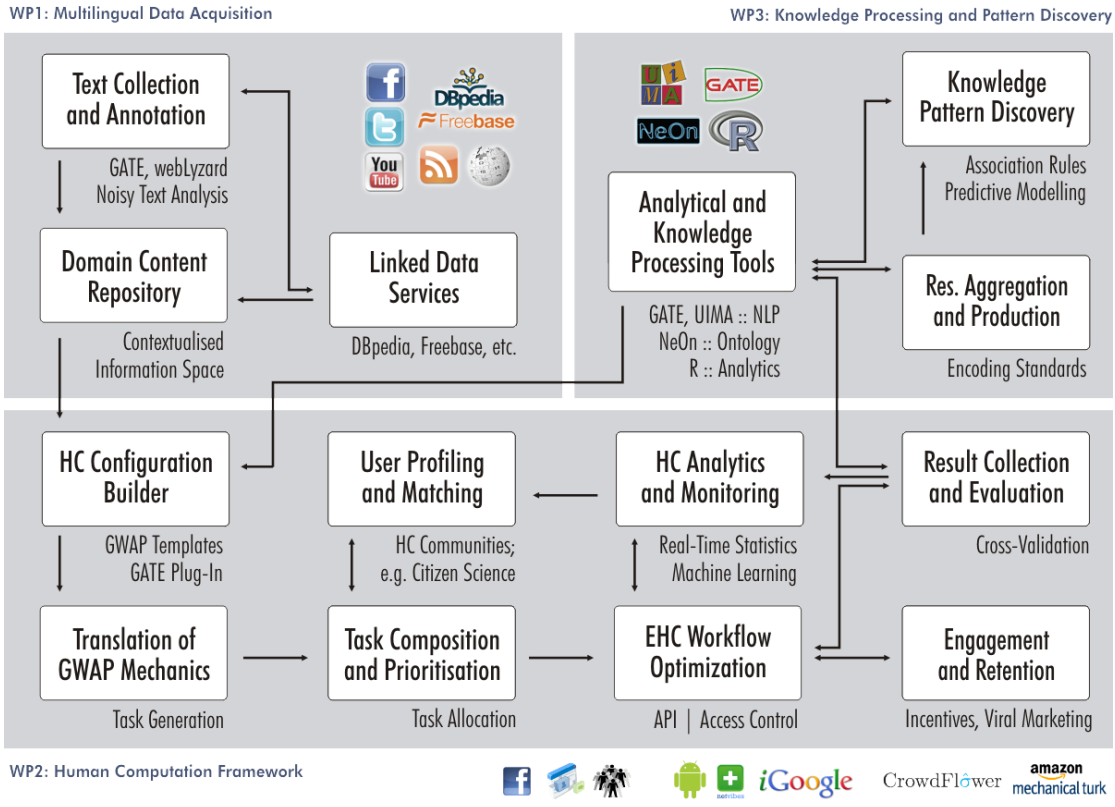

Figure 1. Overview of the uComp System Architecture

Multilingual Data Acquisition

Unstructured Data

A more dynamic crawling architecture deployed within WP1 based on scrapy[1] has replaced the previously used HTTrack.[2] The new architecture offers improved meta data handling (e.g., when gathering social media content via OpenGraph), superior caching control, and is better suited for deployment in distributed scenarios.

A new configuration interface streamlines the setup of all sources, and the workflow on how to process them. This includes the configuration of the input filter, the specification of different mirroring intervals for crawled resources and RSS feeds, and the access parameters for social media APIs - the incoming data stream is first buffered using the MongoDB NoSQL database management system,[3] and later stored in a relational PostgreSQL database.[4]

Building on work done in T1.2 and providing the content feeds for T1.5, the resulting architecture shown in Figure 2 has been designed to be scalable and straightforward to extend. Processing resources in the form of virtual machines can be added on the fly. Tasks are being distributed by Celery[5] and added to RabbitMQ[6], from which the workers get pending tasks and process them.

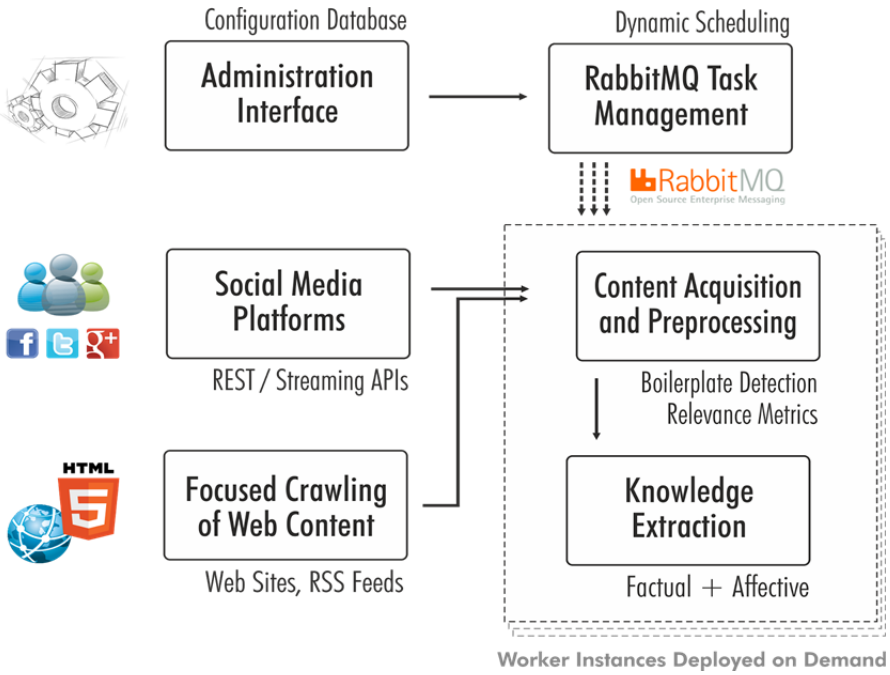

Figure 2. RabbitMQ Task Management

[1] www.scrapy.org
[2] www.httrack.com
[3] www.mongodb.org
[4] www.postgresql.org
[5] www.celeryproject.org
[6] www.rabbitmq.com

Structured Data

The WP2 repository includes data collected and pre-processed from various structured sources including 170 million triples from DBpedia.org, 155 million triples from Geonames.org, and 337 million triples from Freebase.com (see Deliverable 1.3). Additional indicators were acquired from WorldBank[7] and Eurostat.[8] The harvested data contains named entities (people, organisations, locations, dates, events, works - especially from DBpedia, Geonames, Freebase), and statistical indicators (especially from the World Bank and Eurostat). We compiled Geonames using a Python script. For statistical data, we have collected only the indicators that were needed for the climate change use case. We have used Python scripts to perform ontology alignment and index the datasets – see section "Search and Indexing Strategy" below, which ensures that statistical data is available for visualizations as well.

For the named entity recognition system, we created special repositories dedicated to a single type of entity - people from DBpedia or locations from Geonames, for example. RDFSlice (Marx et al. 2013)[9] was used for slicing large datasets per entity type, because it yields all the triples related to a certain object regardless of its position as subject or object in a statement (inverse functional dependency). RDFSlice uses SPARQL queries for matching the triples. The repositories were created for three different languages: English, German, and French. Each dataset is stored in a dedicated repository. The triples stores that host these repositories depend on their size - for repositories smaller than 200 million triples, we use Sesame,[10] otherwise BigData[11] or Virtuoso.[12]

For generating links between various datasets, Silk (Isele et al. 2013) has been adopted with the recommended settings (equality/inequality, Levenshtein distance, Jaccard distance, wgs84), in conjunction with SPARQL queries and federation. The system usually uses rdfs:labels, abstracts or descriptions for the interlinking process, if they are available.

Search and Indexing Strategy

In the second quarter of 2014, the knowledge repository of the *Media Watch on Climate Change* has been migrated to Elasticsearch,[13] a distributed search and analytics engine available under an Apache 2 open source license. It is built on Lucene and provides a RESTful API using JSON over HTTP. Built for the cloud, Elasticsearch further increases the scalability of uComp core components through multitenancy and sharded indexing.

[7] data.worldbank.org
[8] ec.europa.eu/eurostat
[9] www.aksw.org/Projects/RDFSlice.html
[10] www.rdf4j.org
[11] www.bigdata.com
[12] virtuoso.openlinksw.com
[13] www.elasticsearch.org

It radically speeds up accessing the domain-specific content repository (D1.3), thereby also increasing the throughput of data analysis and annotation methods (D1.2), the HC-Based Text Mining Tools (D3.2), and the HC-Based Pattern Discovery Algorithms (D3.3).

Text Processing

Scaling up of the text processing components delivered in WP1 and WP3 of uComp is supported via GATECloud,[14] an open cloud-based platform which enables researchers to deploy, share, and use language processing components and resources, following the data-as-a-service and software-as-a-service paradigms. The focus is on multilingual text analysis resources and services, based on the GATE open-source infrastructure and compliant with relevant NLP standards.

GATECloud is effectively an NLP Platform-as-a-Service (PaaS), a type of cloud computing service which insulates developers from the low-level issues of utilising cloud infrastructures effectively, while providing facilities for efficient development, testing, and deployment of software over the Internet, following the SaaS model (Dikaiakos et al. 2009). In the context of traditional NLP research and development, and pre-dating cloud computing, similar needs were addressed through NLP infrastructures, such as GATE (Cunningham et al. 2013) and UIMA (Ferrucci and Lally 2004). These infrastructures accelerated significantly the pace of NLP research, through reusable algorithms (e.g. rule-based pattern matching engines, machine learning algorithms), free tools for low-level NLP tasks, and support for multiple input and output document formats (e.g. XML, PDF, DOC, RDF, JSON).

The GATECloud platform can be used to develop NLP applications with little or no programming, to index the results for enhanced browsing and search, and to evaluate performance.

Utilising GATECloud is straightforward, since cloud infrastructural issues are dealt with by the platform, completely transparently to the user: load balancing, efficient data upload and storage, deployment on the virtual machines, security, and fault tolerance.

The development of text analysis algorithms and pipelines typically follows a certain methodological pattern, or life cycle. A central problem is to define the NLP task, such that human annotators can perform it with a high level of agreement and to create high quality training and evaluation datasets. It is common to use double or triple annotation, where several people perform the annotation task independently and we then measure their level of agreement (Inter-Annotator Agreement, or IAA) to quantify and control the quality of this data (Hovy 2010).

[14] cloud.gate.ac.uk

The GATECloud platform was therefore designed to offer full methodological support for all stages of the text analysis development life cycle:

1. Create an initial prototype of the NLP pipeline, testing on a small document collection, using the desktop-based GATE user interface (Cunningham et al. 2013);
2. If required, collect a gold-standard corpus for evaluation and/or training, using the GATE Crowdsourcing plugin developed in uComp;
3. Evaluate the performance of the automatic pipeline on the gold standard (either locally in the GATE development environment or on the cloud). Return to step 1 for further development and evaluation cycles, as needed.
4. Upload large datasets and deploy NLP pipeline on the GATECloud PaaS;
5. Run large-scale NLP experiment and download the results as XML or a standard linguistic annotation format (Ide and Romary 2004). GATECloud also offers scalable indexing and search over linguistic annotations and document content.
6. Analyse any errors, and if required, iterate again over the earlier steps.

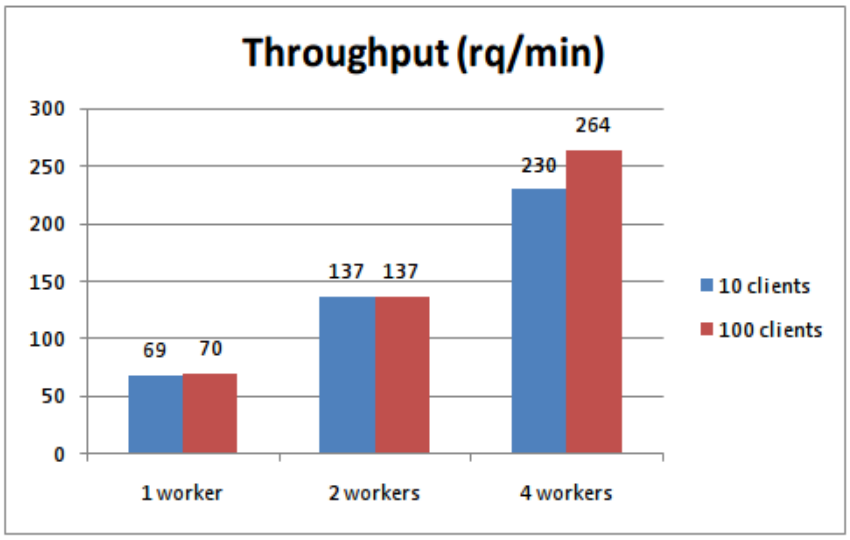

Figure 3. Scalability assessment; ten consecutive requests per client

Scalability and throughput were evaluated as part of the AnnoMarket project.[15] The results in Figure 3 (reproduced from AnnoMarket Deliverable 4.5) show that the throughput of the system scales linearly with the size of the worker swarm (1 to 4 workers). The blue columns represent 10 clients, each sending 10 requests, whereas the red ones are for 100 clients, each sending 10 requests.

These results demonstrate that scalability of the NLP pipelines in uComp can be achieved successfully through GATECloud deployment, as new worker nodes can be added transparently by the platform, if the need arises.

[15] www.annomarket.eu

Ontology Learning

We addressed the issue of performance improvements with regard to the ontology learning framework in a number of ways. First of all, the deployment of caching strategies to avoid redundant computations and API calls, secondly the implementation of an approximation algorithm for the spreading activation method, and finally switching to a new database layout. In the remainder of this paragraph, we will discuss these three aspects in more detail.

Caching

The ontology extension module caches previously computed data to reduce the required computation time for gathering the input data. The caching mechanism of the eWRT[16] module is used for the *corpus*, *social* and keyword modules. However, eWRT is not designed to deal with data that may change over time. A request to the cache is done by supplying the eWRT disk-caching module. The caching module generates a key from its input and tries to find the corresponding data in the disk-cache. If a cache miss happens the given function is called with its parameters returning the result to the initial caller. The cache stores the result for future calls. We had to adapt the eWRT caching modules with additional parameters, which are the month and year for which the ontology will be computed. This supports cache updates in monthly intervals and allows to recompute ontologies using only the data that was collected in a specific month. Caching is applied to computationally complex processes including the computation of keywords via co-occurrence statistics, calls to third-party APIs in the evidence collection phase, and calls to the DBpedia SPARQL endpoint.

Spectral Association

Spectral association is an approximation technique for spreading activation networks (Havasi et al. 2010). Its capability to search associative, neural and semantic networks is used heavily in the uComp ontology learning system, as it is the central algorithm to select and position new concepts. Spreading activation is an iterative process, which cannot be parallelized. Therefore, it is not possible to speed up spreading activation by applying MapReduce strategies. But as spreading activation is very slow for large networks exceeding 20.000 nodes and connections, we had to find a way to improve performance.

Spectral association starts with the transformation of the spreading activation network into a square symmetric matrix C of concepts. The rows and columns of C are labeled with the concepts from the network, and the values in the matrix represent the relation strength between the concepts. Using matrix operations such es calculating eigenvalues and eigenvectors, it is possible to generate a matrix operator e^C which simulates any number of spreading activation rounds. Extensive evaluation shows that spectral association can lead to very significant performance gains, in our settings up to factor 40 compared to classic spreading activation.

[16] www.weblyzard.com/ewrt

Database Redesign

To meet uComp's requirements of regular ontology learning runs on multiple domains and in multiple language, a redesign of the database to store ontology learning results and intermediary data became necessary.

Summary

This deliverable summarized the system architecture and distributed knowledge processing strategy of the uComp project with a focus on the most resource-intensive elements of the its processing pipeline - including data acquisition and analytical services to collect, extract and process factual and affective knowledge from multiple heterogeneous sources and in multiple languages.

Instead of following a MapReduce-based approach that is common for many big data applications, the project adopted a more flexible, container-based architecture that uses a combination of relational and NoSQL databases in conjunction with sharded indexing. Such a modular approach is ideally suited to process data streams (in contrast to batch-oriented Hadoop implementations), and to orchestrate the uComp portfolio of analytic and knowledge processing services including text mining, knowledge extraction, pattern discovery, and semantic search.

References

Cunningham, H., Tablan, V., Roberts, A. and Bontcheva, K. 2013. Getting more out of biomedical documents with gate's full lifecycle open source text analytics. PLoS Computational Biology, 9(2):e1002854, 02.

Dikaiakos, M.D., Katsaros, D., Mehra, P., Pallis, G. Vakali, A.. 2009. Cloud computing: Distributed Internet computing for IT and scientific research. IEEE Internet Computing, 13(5):10–13.

Ferrucci, D. and Lally, A. 2004. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. Natural Language Engineering, 10(3-4):327–348.

Havasi, C., Speer, R., Holmgren, J.: Automated color selection using semantic knowledge. In: AAAI Fall Symposium Series. Arlington, Texas (2010)

Hovy, E. 2010. Annotation. In Tutorial Abstracts of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-2010).

Ide, N. and Romary, L.. 2004. Standards for language resources. Natural Language Engineering, 10:211–225.

Isele, R., Jentzsch, A., Bizer, C. 2010. Silk Server - Adding missing Links while consuming Linked Data. COLD 2010.

Marx, E., Shekarpour, S., Auer, S. 2013. Large-Scale RDF Dataset Slicing. ICSC 2013: 228-235.