Combining and Uniting Business Intelligence with Semantic Technologies
Acronym: CUBIST Project No: 257403
Small or Medium-scale Focused Research Project FP7-ICT-2009-5 Duration: 2010/10/01-2013/09/30

Semantic ETL from unstructured data sources

Abstract:

Type:	Report
Document ID:	CUBIST D2.1.2
Workpackage:	WP2
Leading partner:	ONTO
Author(s):	Pavel Mihaylov (ONTO) Alex Simov (ONTO) Marin Dimitrov (ONTO)
Dissemination level:	PU
Status:	final
Date:	02 October 2012
Version:	1.9

Copyright ▪ CUBIST Consortium 2010-2013

Versioning and contribution history

Version	Description	Contributors
1.1	Initial version	Pavel Mihaylov
1.2	Updated formatting in accordance with template	Pavel Mihaylov
1.3	Added IdRF	Pavel Mihaylov
1.4	Expanded IdRF	Pavel Mihaylov
1.5	Updated use case requirements	Pavel Mihaylov
1.6	Silk and LIMES sections added	Alex Simov
1.7	Added latest GATE development, reorganised sections	Pavel Mihaylov
1.8	Fixed bibliography, intro and conclusion	Pavel Mihaylov
1.9	Implemented review corrections	Pavel Mihaylov

Table of contents

1 Introduction

The goal of this deliverable is to provide an update of existing approaches towards extracting information from unstructured data sources, existing text mining and semantic annotation platforms that can be adapted and deployed as part of the Data Integration and Federation Platform of CUBIST (D2.3.2), as well as an assessment of the suitability of these platforms with respect to the particular requirements of CUBIST use cases for analysing unstructured data. The deliverable builds upon D2.1.1.

The three use cases of CUBIST vary significantly with respect to their requirements for extracting information from unstructured data sources. The HWU use case seemingly contained no unstructured data sources (D7.1.1), but during the course of the project new unstructured data sources were identified in the context of the use case. The SAS use case provides only some unstructured data sources and work is focused on identifying entities and relations from texts and interlink them with the corresponding documents. The INN use case is the one where the text mining task is of highest importance, not only because most of the data sources are unstructured, but also due to specific requirements of the use case for automated qualification extraction, sentiment mining and identity resolution, which go beyond the traditional information extraction and semantic annotation.

This deliverable is organised as follows:

- Section 2 provides a brief summary of previous work in the domain of text mining and semantic annotation from unstructured data sources.
- Section 3 provides an overview of recent developments regarding GATE and introduces IdRF, a framework for identity resolution that can benefit data integration, as well as Silk and LIMES, two other identity resolution solutions.
- Finally, section 4 provides an update of current work in extracting information from unstructured data sources, with respect to the specific requirements of the CUBIST use cases.

2 Previous work

This section outlines previous work on methods, techniques and platforms for ETL from unstructured data sources, which were described in detail in D2.1.1.

2.1 Methods and techniques

This section provides a summary of the methods and techniques for ETL from unstructured data sources identified in D2.1.1.

2.1.1 Text mining

Text mining is the process of deriving high-quality structured information from unstructured (textual) data in natural language. This involves structuring the input text, deriving patterns from the structured data, plus evaluation and interpretation of the output. Some of the typical text mining tasks are text categorisation, concept/entity extraction, taxonomy or ontology generation, sentiment/opinion analysis, document summarisation, and relationship extraction.

The textual data sources—for example: user manuals, instructions, reports, etc.—contain information which has to be processed and transformed into structured metadata in order to be aligned with existing entities in the knowledge base (KB). The generation of metadata involves information extraction and assigning information based on a given ontology in the form of annotations. Metadata is acquired with natural language processing (NLP) based techniques that identify relevant pieces of information. The transformation of textual information into metadata is the process of identifying chunks of text, sequences of words, as describing particular metadata.

2.1.2 Semantic annotation

Semantic annotation is the process of identifying knowledge elements in text and mapping them to instances and entities in a given knowledge base. It is the process of automatic generation of named-entity[1] annotations with class and instance references to a semantic repository. Semantic annotation is applicable for web, non-web documents, and text fields in databases.

The semantic annotation process can be seen as a classical named entity recognition and annotation process. The named entity type is specified by reference to an ontology, and the semantic annotation requires identification of the entity. The approach of semantic annotation comprises two processes: *information extraction* and *identity resolution*.

2.1.2.1 Information extraction

Information extraction refers to the lightweight process of detecting pieces of relevant informational units in texts and representing them in the form of attribute/value templates [1] [2]. For example, the information extraction engine will detect the expression “event in London” and place the word “London” in the respective template as the value for the attribute “location”.

[1] *Named-entity* stands for a phrase that clearly identifies one item from a set of other items that have similar attributes (person names, company names, geographic locations, etc.)

Some natural language processing techniques explore particular parts of the texts to identify *named entities* such as organisation names, person names, geographical locations, monetary units and time expressions.

Named Entity Recognition (NER) is a subtask of the information extraction process, which identifies and classifies language expressions into predefined categories, and annotates the input text with the recognised categories.

There are different approaches to information extraction: *statistical*, *rule-based* and *hybrid* approach.

The information extraction process identifies the text chunk—the knowledge element—and proceeds to identity resolution by matching the result to the instance information for each known named entity in the text. It adds the new entities with their semantic descriptions and relations to the knowledge base. Thus, as a result each named entity is linked to its type and to its individual semantic description.

2.1.2.2 Identity resolution

The process of *identity resolution* (also called *entity reconciliation*) has the goal to determine whether two or more data representations refer to the same object and thus should be resolved into one representation. For example, identity resolution looks at addresses, names, social security numbers, dates, and customer history to make interconnections between different identity records. The process of identity resolution is closely related to the ontology development and use. The named entity references in the text are linked to an entity individual in the knowledge base. The entity instances all bear unique identifiers that allow the annotations to be linked to the exact individual in the knowledge base and thus each recognised named entity is linked to an individual in the knowledge base and the associated semantic description.

Identity resolution is necessary in the process of data acquisition from multiple data sources or knowledge bases in order to ensure that all the information related to an entity instance is aggregated together and there are no duplicate objects in the knowledge base referring to the same instance.

2.1.3 Sentiment analysis

Sentiment analysis or *opinion mining* aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. The attitude may be his or her judgment or evaluation, affective state (the emotional state of the author when writing), or the intended emotional communication (the emotional effect the author wishes to have on the reader). A basic task in sentiment analysis is classifying the polarity of a given text at the document, sentence, or feature/aspect level—whether the expressed opinion in a document, a sentence or an entity feature/aspect is positive, negative or neutral.

Automated sentiment analysis of digital texts utilises elements from machine learning such as latent semantic analysis, support vector machines and "bag of words"[2]. More sophisticated methods try to detect the holder of a sentiment (i.e. the person who maintains that affective state) and the target (i.e. the named entity or target whose affective state one is interested in). To mine the opinion in context and get the feature, which has been opinionated, the grammatical relationships of words are used. Grammatical dependency relations are obtained by deep parsing of the text [3].

[2] http://en.wikipedia.org/wiki/Bag_of_words

2.2 Text mining platforms

This section provides a brief summary of the GATE and KIM platforms for text mining and semantic annotation described in D2.1.1.

2.2.1 GATE (General Architecture for Text Engineering)

GATE[3] is a platform for developing and deploying text mining software components. GATE is open source software (distributed under a LGPL license[4]); users can obtain free support from the user and developer community or on a commercial basis from the ecosystem of industrial partners. GATE comes in several editions suited to different tasks: GATE Developer, GATE Teamware and GATE Embedded.

GATE includes components for diverse language processing tasks, e.g. parsers, morphology analysers, part-of-speech taggers, information retrieval tools, information extraction components for various languages, etc. GATE Developer and GATE Embedded are supplied with a standard information extraction pipeline (ANNIE).

The GATE architecture is based on components: reusable chunks of software with well-defined interfaces that may be deployed in a variety of contexts. The design of GATE is based on an analysis of previous work on infrastructure for language engineering.

GATE comes with various built-in components on board that can be readily used to develop applications or as basis for more advanced components.

Another powerful GATE component is JAPE (Java Annotation Patterns Engine). It provides finite state transduction over annotations based on regular expressions. JAPE makes it possible to recognise complex regular expressions in annotations on documents, and custom annotation manipulation routines written in Java.

2.2.2 KIM

KIM[5] is a software platform for automated semantic annotation, indexing, and retrieval of unstructured and semi-structured content. The most popular use cases for KIM are:

- Generation of meta-data for the Semantic Web, which allows hyper-linking and advanced visualisation and navigation.
- Semantic search over unstructured and semi-structured content.

KIM analyses textual content and identifies references to entities (persons, organisations, locations, dates, etc.) or the relations that exist between entities (such as job positions). Then it tries to match the discovered entities to already known entities in the knowledge base. Finally, the original documents are enriched with metadata about the entities and relations that they contain. The whole process is referred to as *semantic annotation*.

---

[3] http://gate.ac.uk/

[4] http://www.gnu.org/licenses/lgpl.html

[5] http://www.ontotext.com/KIM

In order to allow the easy bootstrapping of semantic annotation applications, KIM is equipped with a small upper-level ontology called PROTON[6]. Furthermore, KIM provides a knowledge base pre-populated with about 200,000 entity descriptions for the most popular entities. Its role is to provide background knowledge that improves the semantic annotation process.

KIM incorporates several popular open-source platforms: GATE, Sesame[7] and Lucene[8].

KIM is a highly adaptable and modular platform for linking and navigating data, content, and knowledge. It can be configured to use all or some of its components to suit different needs. Computationally intense components such as concepts extraction and semantic database can be clustered to reach the performance you need.

[6] http://proton.semanticweb.org/

[7] An RDF repository and framework: http://www.aduna-software.com/technology/sesame

[8] An open-source information retrieval engine: http://lucene.apache.org/

3 Platforms for ETL from unstructured data sources

This section provides an update of the latest developments in software for ETL on unstructured data sources. It builds upon work described in detail in D2.1.1 and summarised in section 2 of this deliverable.

3.1 New GATE components

As GATE is a component-based platform, it is easy to extend with additional functionality. This section describes new GATE extensions that benefit information extraction from unstructured data sources.

3.1.1 Advanced HTML processing

The HTML parser and document model that come with the standard GATE distribution have no special support for HTML tags. When new HTML documents are created, the tags are stripped and converted to simple annotations. This approach has several disadvantages:

- The HTML tree is lost and tag nesting is not directly accessible.
- Text content appears as plain text and there is no easy way to process text based on its HTML context.

These issues make it difficult to rely on HTML structures when performing information extraction on the unstructured text, or in other words they make the unstructured data source even more unstructured. To remedy the situation, Ontotext developed two new GATE components that deal with HTML in a more structured way:

- HTML DOM parser: the HTML is parsed to a W3C DOM tree and that information is kept as a document feature.
- XPath processing resource: uses the DOM tree to execute XPath queries, which allows very fine-grained access to text and annotations in respect to HTML context.

The two features can be combined to focus information extraction only in certain portions of the HTML document, e.g. job titles on vacancy pages are typically found in major structuring tags such as H1, H2, etc. This allows for better precision by avoiding false positives contained within less important tags.

The components are based on Ontotext's Web Mining Framework (WMF)[9].

3.1.2 Location and date parser

Many information extraction applications need to process locations and dates found in unstructured data sources. Typically, GATE applications solve these tasks by using the standard gazetteers and rules from ANNIE. They will recognise if something *is* a location (= found in the location gazetteer) or a date (= syntactically correct for a date), but they will not provide any information on *which* location exactly (e.g. London may refer to several other places besides the capital of the UK) or point in the time continuum it is (e.g. 27th September 2012 refers to a predefined exact day).

[9] http://www.ontotext.com/wmf

Ontotext developed two new GATE processing resources (PRs) based on advanced location and date parsing provided by Ontotext's WMF. The PRs are capable of:

- Location parsing and disambiguation, e.g. London, UK and London, Canada will be parsed and recognised as two distinct entities, respectively referring to London in the United Kingdom (http://www.geonames.org/2643743/london.html) and London in the Canadian province of Ontario (http://www.geonames.org/6058560/london.html).
- Parsing dates of various formats, e.g. 27/09/2012, 27.09.2012, 2012-09-27, 27-Sep-2012 and 27th September 2012 will all be parsed to a single unique value[10].

3.2 IdRF (Identity Resolution Framework)

IdRF [4] is a Java framework that implements identity resolution for generic objects. IdRF uses *entity descriptions* to represent both input and output objects. Every entity description describes exactly one instance object using RDF. Entity descriptions are thus very simple triple stores and the framework can work directly with RDF data. The identity resolution rules are specified in a custom human-friendly language based on first order predicate logic.

3.2.1 IdRF Architecture

There are three main data processing components. Processing (see Figure 1) starts from a new entity that is passed to the system. Then the *pre-filtering* component will retrieve all entity candidates from the target dataset and for each of them will build a candidate pair. All the pairs will then be passed to the *evidence collection* component, which will calculate the similarity between the two objects in the pair. The results will be finally passed to the *decision maker* component that fuses the data and activates the entity storing procedure. The last step is optional.

The backbone of the framework is the Semantic Description Compatibility Engine (SDCE), which is responsible for the mediation between the data processing component and the repository. It translates the pre-filtering restrictions into semantic queries and the similarity rules into executable methods. It plays the most significant role in the implementation and contains domain specific rules for identification. According to the knowledge representation, each entity belongs to a particular class in the ontology and it is also associated with a set of predefined properties and relations. Therefore, the set of entity description criteria can be retrieved based only on the entity description. The criteria are coded as predicates and form rules/formulas that describe the whole class of entities. Then the interpretation of these rules is performed by the SDCE. The predicates may provide a general purpose calculation or a specific measure and may be used only for a certain class model. For example, specific handling for alias similarity for the class Person deals with abbreviations of the first name, while the alias similarity for Organisation includes analysis of company suffixes. Thus the respective formulas will use different predicates for alias comparison.

[10] The 734773rd day of the Common Era in the proleptic Gregorian calendar.

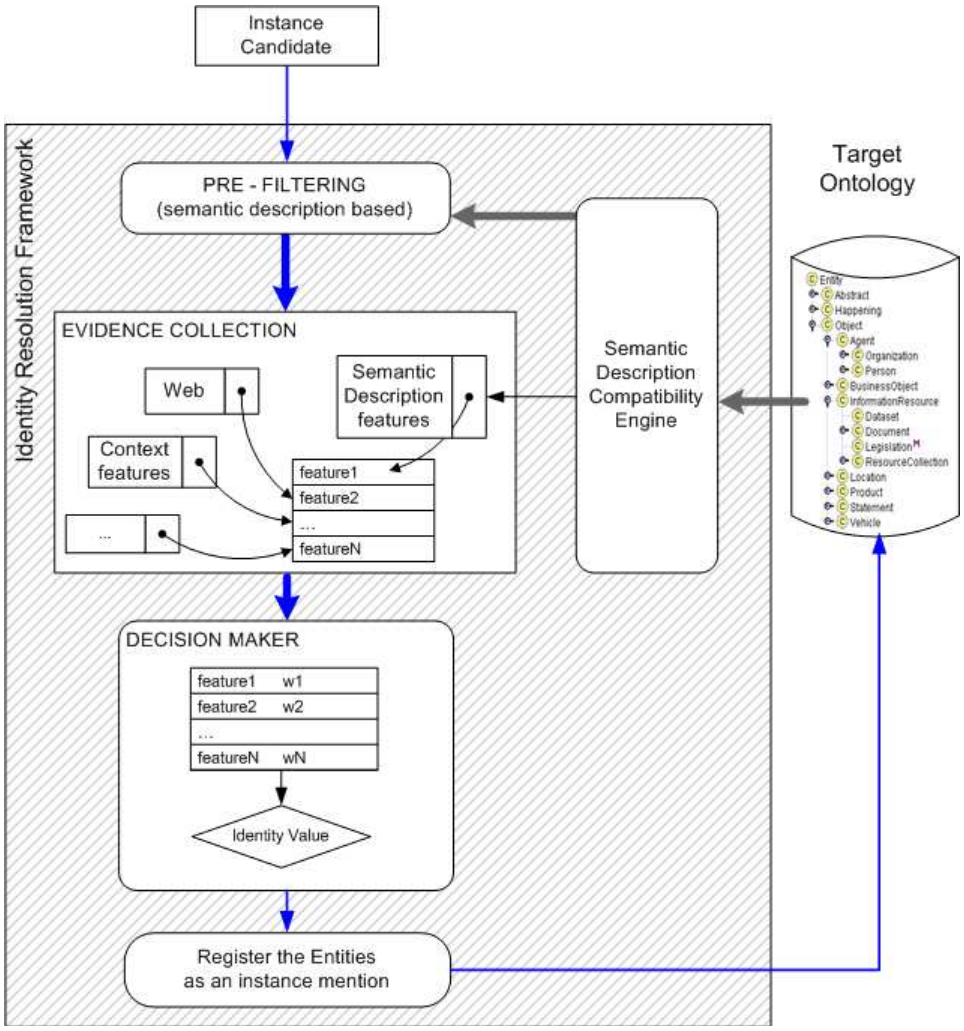

Figure 1. Identity Resolution Framework

The *Pre-filtering* component is responsible for candidate selection. It filters out the irrelevant part of the ontology and forms a smaller set of instances similar to the source object. Since the source repository can contain a huge amount of data (millions or even billions of objects), comparing all existing records might not be feasible. Pre-filtering is intended to restrict the ontology instances to a reasonable number, to which the source object will be compared. It can be regarded as pre-selection of ontology objects that are eligible for identification. The selected instances are potential target instances that might be identical to the source object; they already appear in the knowledge base and are somehow similar to the source object.

The *Evidence Collection* component is responsible for the similarity measure. It aims to collect as much as possible evidence about the similarity between the source object and each of the identification targets in the ontology. The evidence comes from various identity criteria. They express different aspects of the identity of the source object with each of the instances selected during the pre-filtering stage. Simple weighting of evidences can often be the cheapest solution (both in terms of setting up and calculation), especially when dealing with well defined objects described in simple structures. The default identity criterion is based on the semantic descriptions of the objects and it is calculated according to class definitions and the ontology data. Other custom criteria could be lexical distance of names of the objects, web appearance, context similarity, etc.

Once all the evidence for different identity possibilities are collected, the *Decision Maker* has to select the best identity match. The decision is based on the strength of the presented evidence and whether it demonstrates identity between the source object and the target instance it is compared with. The final decision about the identity of a given fact is actually a choice between all possible candidates. It is based on the collected evidence about each of the candidates and decides which of them are identical to the currently processed object. Obviously, this decision is not trivial and it strongly depends on the entity type and the application domain. Once all identity evidence is collected, they need to be ranked and combined in such a way that their relevance to the entity type and domain is preserved. For example, birth date can be a very strong indication of People identity, but published date is a questionable evidence for equivalence of job offers. Furthermore, this component fuses the new object with the identity candidate resolving all possible conflicts between the attribute values of the two entity descriptions.

After the decision is made, the incoming object can be optionally registered to the ontology as a final stage in the IdRF. The source object can be either a new one or successfully identified with an existing instance. If the system is not able to find a reliable match, the incoming object is inserted as a new instance in the KB. In case it is associated with an existing instance, then the object description is added to the description of the identified KB instance. The integration of the newly processed data enriches the ontology adding either entirely new objects or only new attributes sources to the existing ones. In this way the ontology aggregates the description of all the incoming objects and provides a single view to the processed data. As an effect of the constantly enlarging KB, the identity criteria are dynamically refined improving the identity resolution by both refining the evidence calculation and introducing new entities serving as identity goals. Details about the two effects are given below:

- The evidence calculation is refined when a new value, attribute, property or relation is added to an existing instance description. Then, the identity criteria for this instance are changed in order to reflect the newly available data adding new comparison restrictions. For example, if the person age is added to his/her description, the age restriction will be added as a new identity criterion.
- New identity goals are all the new entities that are added to the knowledge base. They are created by insertion of entirely new objects to the KB. Then the entities that are processed in a later stage have to be compared not only to the previously available entities but also to the newly added instances.

Each of the above stages is supported by a default implementation that is easily configurable for a new domain.

3.2.2 Predicates

IdRF predicates are individual interchangeable units that play a central role in the identity resolution process and are directly referenced in rules. They are used for both pre-filtering and evidence collection as defined previously. When used for pre-filtering, each predicate will select a set of candidates (potential matches) that need to be compared to the input object. When used for evidence collection, each predicate will compare the input object to a candidate and return a *match score*. The match score indicates how similar the compared objects are. It is a real number between 0.0 and 1.0, where 0.0 indicates no similarity at all and 1.0 indicates the two objects are fully identical.

Predicates typically have one or more arguments that define the attribute(s) or relation(s) to be compared. The framework distinguishes three types of predicates:

- *Built-in* Java predicates; a set of a predicates that provide implementations for common string similarity algorithms, number comparison, relation identity etc. For example, the predicate SameRelation(x) will check if the given relation x is the same in both the input and the candidate object.
- *Third-party* Java predicates; these are Java classes that implement the Predicate interface. They are used to implement custom comparison algorithms, e.g. matching company names with respect to non-significant abbreviations like Ltd or GmbH.
- *User-defined* predicates; these are predicate chains consisting of predicates and operators assigned to a new identifier much like defining functions in programming languages.

3.2.3 Rules language

The IdRF rules language is based on first order predicate logic and thus it allows for easy and intuitive expression of identity rules. Rules are expressed as logical formulas, which can be evaluated in order to produce a match score. The building blocks of IdRF rules are:

- IdRF predicates.
- Common logical connectives:
  - Conjunction: $a \wedge b$. The score is equal to ab.
  - Disjunction: $a \vee b$. The score is equal to $a + b - ab$.
  - Negation: $\neg a$. The score is equal to $1 - a$.
  - Material implication: $a \rightarrow b$. Equivalent to $\neg a \vee b$, so the score is equal to 1 – a + ab.

The above can be freely combined to write formulas where a and b are IdRF predicates.

3.3 Silk Link Discovery Framework

The Silk Link Discovery Framework[11] is a tool for discovering relationships between data items within different Linked Data sources. Using the declarative *Silk Link Specification Language*[12], users can specify which types of RDF links should be discovered between data sources as well as which conditions data items must fulfil in order to be interlinked. These linkage rules may combine various similarity metrics and can take the graph around a data item into account, which is addressed using an RDF path language. Silk accesses the data sources that should be interlinked via the SPARQL protocol and can be used against local as well as remote SPARQL endpoints.

Silk is provided in three different variants, which address different use cases:

- *Silk Single Machine* is used to generate RDF links on a single machine. The datasets that should be interlinked can either reside on the same machine or on remote machines that are accessed via the SPARQL protocol. Silk Single Machine provides multithreading and caching. In addition, the performance can be further enhanced using an optional blocking feature.

[11] http://www4.wiwiss.fu-berlin.de/bizer/silk/

[12] https://www.assembla.com/wiki/show/silk/Link_Specification_Language

- *Silk MapReduce* is used to generate RDF links between data sets using a cluster of multiple machines. Silk MapReduce is based on Hadoop[13] and can for instance be run on Amazon Elastic MapReduce[14]. Silk MapReduce enables Silk to scale out to very big datasets by distributing the link generation to multiple machines.
- *Silk Server* can be used as an identity resolution component within applications that consume Linked Data from the Web. Silk Server provides an HTTP API for matching instances from an incoming stream of RDF data while keeping track of known entities. It can be used together with a Linked Data crawler to populate a local duplicate-free cache with data from the Web.

All variants are based on the *Silk Link Discovery Engine,* which offers the following features:

- Flexible, declarative language for specifying linkage rules.
- Support of RDF link generation (*owl:sameAs* links as well as other types).
- Employment in distributed environments (by accessing local and remote SPARQL endpoints).
- Usable in situations where terms from different vocabularies are mixed and where no consistent RDFS or OWL schemata exist.
- Scalability and high performance through efficient data:
    - Reduction of network load by caching and reusing of SPARQL result sets.
    - Multi-threaded computation of the data item comparisons.
    - Optional blocking of data items.

3.3.1 Architecture

Consider Figure 2. The *DataSource* generates a stream of data items. The optional *Blocking* phase partitions the incoming data items in clusters. The *Link Generation* phase reads the incoming data items and computes a similarity value for each pair. The incoming data items, which might be allocated to a cluster by the preceding blocking phase, are written to an internal cache. From the cache, pairs of data items are generated. If blocking is disabled, this will generate the complete Cartesian product of the two datasets. If blocking is enabled, only data items from the same cluster are compared. For each pair of data items, the link condition is evaluated, which computes a similarity value between 0.0 and 1.0. Each pair generates a preliminary link with a confidence according to the similarity of the source and target data item.

Figure 2. Silk workflow diagram

The *Filtering* phase filters the incoming links in two stages:

1. In the first stage, all links with a lower confidence than the user-defined threshold are removed.

[13] http://hadoop.apache.org/

[14] http://aws.amazon.com/elasticmapreduce/

2. In the second stage, all links that originate from the same subject are grouped together. If a link limit is defined, only the links with the highest confidence are forwarded to the output. The number of links which are forwarded per source item, is specified by the link limit

Finally, the *Output* phase is responsible for writing the generated and filtered links to a user-defined destination.

3.3.1.1 Blocking

As the data to be processed is usually huge there is an increasing need for link discovery tools, which scale to very large datasets. A number of methods have been proposed to improve the efficiency of link discovery by dismissing definitive non-matches prior to comparison. The most well-known method to achieve this is known as blocking. Unfortunately, traditional blocking methods need a separate configuration and in general lead to a decrease of recall due to false dismissals.

Silk employs a novel blocking method that maps entities to a multidimensional index (see Figure 3). The basic idea of the mapping function is that it preserves the distances of the entities, i.e. similar entities will be located near to each other in the index space. Blocking works on arbitrary link specifications, which aggregate multiple different similarity measures such as string, geographic or date similarity. No separate configuration is required as the indexing is directly based on the link specification and all parameters are configured automatically. Additionally, it guarantees that no false dismissals and thus no loss of recall can occur.

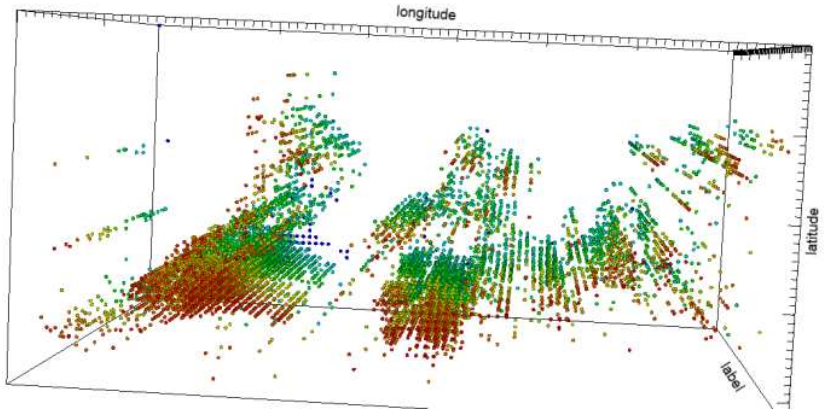

Figure 3. Silk multidimensional index

3.3.2 Linkage rules learning

The link discovery process described so far is based on manually written linkage rules, which specify the condition, which must be fulfilled for two entities in order to be interlinked. Additionally the Silk Framework offers a machine learning implementation, which automatically generates linkage rules from a set of reference links. The learning algorithm is based on genetic programming and it is capable of generating complex linkage rules which compare multiple properties of the entities and employ data transformations in order to normalise their values.

The implementation of the algorithm treats the expression of each possible linkage rule as a tree. A fitness function measures the quality of the rules with respect to the training data. An iterative process refines a set of candidate rules trying to improve fitness measures by applying transformations on the tree representations. On each iteration, the rules with highest fitness from the previous iteration are

preserved and new rules are produced until certain set size is reached. This process continues until a certain level of fitness or certain number of iterations is reached.

A web-based UI frontend (see Figure 4) supports the user in the process of initiating the learning process, customising it and reviewing the results.

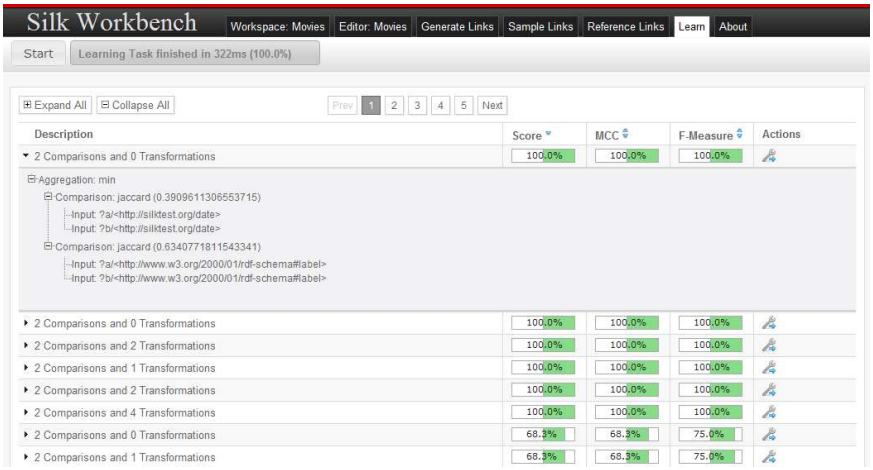

Figure 4. Rules Learning in Silk Workbench

3.3.3 Benchmarking

A series of experiments have been performed at Ontotext with different setups and data extracts. Here we present the results from a single experiment involving data from two popular linked data sets: *DBpedia*[15] and *GeoNames*[16].

The experiment evaluates the ability of the system to learn linkage rules on the base of small portion (approx 10%) of predefined reference mappings. We took the reference data from *DBpedia*[17] website and extracted a subset relevant for the experiment, i.e. linking cities from DBpedia to GeoNames. The input entities coming from the two dataset have the following characteristics:

DBpedia cities (*rdf:type dbpedia-ont:City*):

- rich set of various properties, values in different languages, aliases.
- noisy data – language tags mismatch, incorrect decimal values and geo coordinates.

GeoNames cities (*geonames:featureCode geonames:P.PPL*)

- small set of properties – city name + geo coordinates.
- cleaner data – precise coordinate values.

The results of the experiment measure the coverage of the suggested links on the reference links (*recall*) and the incorrectly suggested links (*precision)*.

The diagram (Figure 5) below shows the result linking with different confidence levels (percentage).

[15] http://dbpedia.org/About

[16] http://www.geonames.org/

[17] http://wiki.dbpedia.org/Interlinking

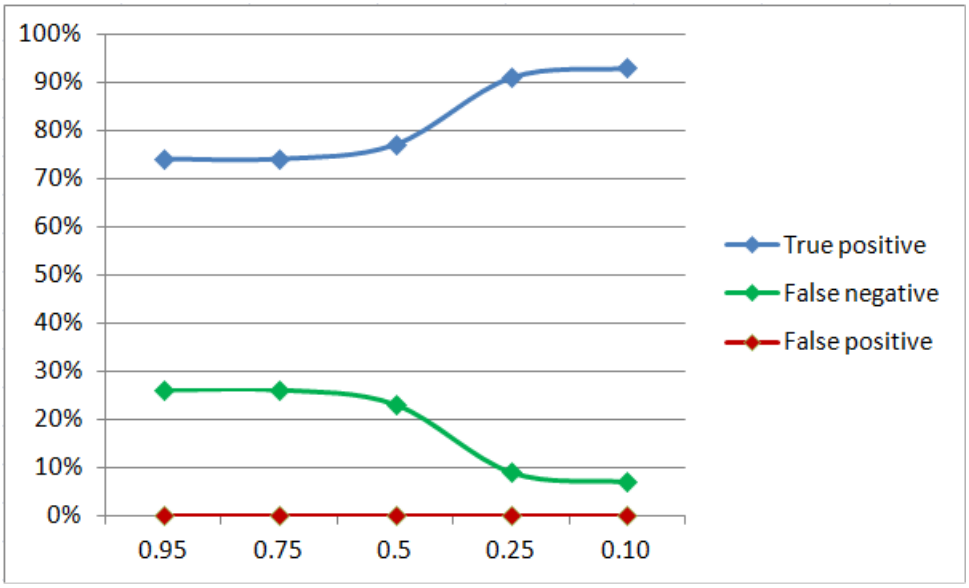

Figure 5. Silk benchmark results

The results show that the system covers most of the positive links but with low levels of confidence and in the same time it disallows incorrect linking (the red line). A closer look at the undiscovered links reveals that for some of them there are incorrect property values in the corresponding entities.

This experiment proves Silk is a reliable platform for data linkage and although it does not always find the correct link, it will not introduce noise by incorrect linking.

3.4 LIMES – Link Discovery Framework for Metric Spaces

LIMES[18] implements time-efficient approaches for large-scale link discovery based on the characteristics of metric spaces. The approaches utilise the mathematical characteristics of metric spaces to compute estimates of the similarity between instances. These estimates are then used to filter out a large amount of those instance pairs that do not suffice the mapping conditions. By these means, LIMES can reduce the number of comparisons needed during the mapping process by several orders of magnitude.

[18] http://aksw.org/Projects/limes

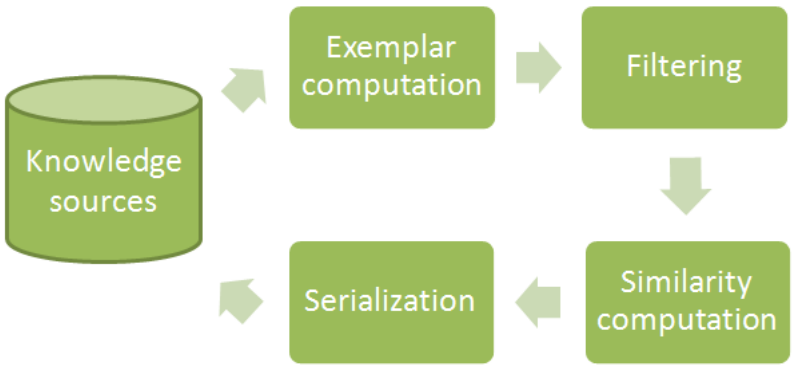

Figure 6. LIMES workflow diagram

The general workflow (see Figure 6) implemented by the LIMES framework comprises four steps: Given a source, a target and a threshold, LIMES first computes a set of exemplars for the target data source (step 1). Matching each target instance to the closest exemplar concludes this process. In steps 2 and 3, the matching is carried out. In the filtering step, the distance between all source instances and target instances is approximated via the exemplars computed previously (step 3). Obvious non-matches are then filtered out. Subsequently, the real distance between the remaining source and target instances are computed (step 3). Finally, the matching instances are are serialised, i.e., written in a user-defined output stream according to a user-specified format, e.g. NTriples (step 4).

4 ETL from unstructured data sources in CUBIST

Based on the analysis in D2.1.1 and the updates from section 2.2, this section will provide an updated outline for the work in Task 2.1, with respect to the specific requirements of the three CUBIST use cases.

CUBIST D3.1.1 already provides a brief description of the various data sources that the three CUBIST use cases provide but over the course of the project, new requirements were identified. The following sections summarise D3.1.1 and introduce new requirements and general updates on progress.

4.1 WP7 HWU

The HWU use case appeared not to contain any unstructured data, since all the EMAP/EMAGE data was stored in (structured) databases. Analysing textual content from scientific journal publications related to EMAGE was considered as optional requirement.

There is, however, newly identified unstructured data. The data is spatial-temporal information that is contained implicitly within EMAGE images. The information cannot be directly accessed by a computer; further processing and reasoning is required. The goal is to create a representation that provides the most useful information already processed and interpreted (to some degree, i.e., further work may be required depending on the exact use case). Ultimately, the representation will be a summary of the original image, but will not be an exact replication of the information stored within the source image.

As the data is not text-based, none of the text mining solutions is applicable. Work on this requirement is ongoing and will be discussed in more detail in later deliverables.

4.2 WP8 SAS

The SAS use case provides a limited set of unstructured data sources, usually in the form of human generated notes, messages or communication transcripts. The SAS systems currently in use do not analyse the unstructured content. The requirement for the SAS use case within CUBIST is to identify entities and relations in the text content and to interlink it with the data from the structured data sources in order to provide more expressive search capabilities. Valuable features of the Semantic ETL system would be support for co-reference resolution and fuzzy matching.

The first version of the CUBIST prototype (D2.3.1) describes information extraction from unstructured space logs using GATE. The main idea was to extract and annotate abbreviations and various numeric values, including those specific to flight dynamics.

4.3 WP9 INN

The INN use case also provides a mixture of structured and unstructured data sources (with predominantly *unstructured* ones). The unstructured data consists of vacancy description, crawled company web pages, public domain news sources, company forums, social streams such as twitter. This is also the use case that requires the most advanced types of text mining, going beyond the traditional entity and relation extraction, e.g. automated job categorisation, automated skill and qualification extraction. A newly identified requirement is identity resolution.

GATE plays a central role the ETL process for the INN use case. The complete process is described in detail in D2.3.1.

4.3.1 WP9 qualification extraction

The European Commission is developing a European Skills, Competences and Occupations taxonomy (ESCO)[19], which will describe the most relevant skills, competences and qualifications needed for several thousand occupations. At European level, ESCO will contribute to improving the services provided by EURES, the European Job Mobility Portal and make it possible to develop new services such as Match & Map, which aims to improve the matching of jobseekers to available jobs. ESCO is part of New Skills for New Jobs, a joint policy initiative carried out in cooperation between the European Commission and the EU Member States to foster skills development and employability.

As ESCO is an ontology that contains skills, qualifications and occupations, it is the perfect choice for modelling qualifications extracted from job vacancies as required by the use case.

4.3.2 WP9 sentiment mining

The initial goal was to implement a system for polarity mining that would be able to detect the positive or negative sentiment expressed towards the activities of an organisation. However, at present there is no collected data to process.

4.3.3 WP9 identity resolution with IdRF

The INN use case collects data from multiple data sources. It is important to ensure that all the information related to an entity instance is aggregated together and there are no duplicate objects in the knowledge base referring to the same instance. For example:

- One and the same *job vacancy* is typically posted on multiple job boards (or rarely multiple times on the same job board), or a single job vacancy may appear multiple times under different URLs with a varying amount of detail on websites that use dynamic publishing.
- *Advertising organisations* appear under slightly different names on different job boards with a varying amount of details (address, contact information, website) and all new advertisers have to be linked to an existing company database in order to produce accurate market reports.

Both problems cannot be solved by simple comparison of the extracted structured content. This is where IdRF can help by providing state-of-the-art identity resolution for job vacancies and organisations.

[19] About ESCO (2011): http://ec.europa.eu/social/BlobServlet?docId=5653&langId=en

5 Conclusion

This deliverable provides the second version of the analysis of CUBIST use case requirements for extracting information from unstructured data sources, an update of existing solutions to support these requirements, as well as an update and general progress on identified requirements and goals within the three use cases.

D2.1.1 focused mostly on *Extract*, while this deliverable logically continues to *Transform*. The central tools in the process are several new GATE components and platforms for identity resolution. The first concrete results were provided within the first integrated prototype (D2.3.1), while further work on extracting information from unstructured data sources within CUBIST will continue with the second integrated prototype of the CUBIST data integration and federation platform (D2.3.2).

6 Bibliography

1. **Sarawagi, S.** *Information Extraction.* Foundations and Trends® in Databases : p.261-377, 2008.

2. **Grishman, R.** *Information Extraction: Techniques and Challenges.* International Summer School on Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology, Springer-Verlag : p.10-27, 1997.

3. **Lipika Dey, S K Mirajul Haque.** *Opinion Mining from Noisy Text Data.* Proceedings of the second workshop on Analytics for noisy unstructured text data : p.83-90, 2008.

4. **Yankova, M.** *Terms: Text Extraction From Redundant And Multiple Sources.* Sheffield : University of Sheffield : p.129-148 http://etheses.whiterose.ac.uk/933/2/yankova_final.pdf, 2010.