Multimedia Interpretation for Dynamic Ontology Evolution

Silvana Castano[1], Sofia Espinosa[2], Alfio Ferrara[1], Vangelis Karkaletsis[3], Atila Kaya[2], Ralf Möller[2], Stefano Montanelli[1], Georgios Petasis[3], Michael Wessel[2]

[1] Università degli Studi di Milano,
DICo, 10235 Milano, Italy,
{castano, ferrara, montanelli}@dico.unimi.it
[2] Hamburg University of Technology,
Institute for Software Systems,
21079 Hamburg, Germany,
{sofia.espinosa, at.kaya, r.f.moeller, mi.wessel}@tuhh.de
[3] Software and Knowledge Engineering Laboratory,
Institute of Informatics and Telecommunications,
National Centre for Scientific Research (N.C.S.R.) "Demokritos",
P.O. BOX 60228, Aghia Paraskevi, GR-153 10, Athens, Greece.
{vangelis, petasis}@iit.demokritos.gr

Abstract. The recent success of distributed and dynamic infrastructures for knowledge sharing has raised the need for semiautomatic/automatic ontology evolution strategies. Ontology evolution is generally defined as the timely adaptation of an ontology to changing requirements and the consistent propagation of changes to dependent artifacts. In this article, we present an ontology evolution approach in the context of multimedia interpretation. Ontology evolution in this context relies on the results obtained through reasoning for the interpretation of multimedia resources, through population of the ontology with new individuals, or through enrichment of the ontology with new concepts and new semantic relations. The article analyses the results of interpretation, population and enrichment obtained in evaluation experiments in terms of measures such as precision and recall. The evaluation reveals encouraging results.

1 Introduction

Ontology evolution is generally defined as the timely adaptation of an ontology to changing requirements and the consistent propagation of changes to dependent artifacts. Ontology evolution empowers ontology learning which is a wide domain of research, involving methods and techniques for the acquisition of ontologies from semantic information/conceptual knowledge extracted from a domain. Being closely related to the field of knowledge acquisition, a significant amount of work has been presented in the literature that concentrates on the task of knowledge acquisition from text, through the re-use of widely adopted natural language processing and machine learning techniques [1, 2]. The ontology evolution methodology proposed in this article extends existing approaches by considering modalities beyond text, such as still images, video and audio. Research presented in this article has been carried out

as part of the project BOEMIE (Bootstrapping Ontology Evolution with MultImEdia Information).[1]

In the BOEMIE project, ontologies are used for the representation of domain specific background knowledge. Additionally, ontologies are also used as repositories for storing semantic content descriptions of multimedia documents such as images, text, video and audio. Semantic content descriptions conceptualize the world in terms of concepts (sets of individuals) and relations (set of tuples of individuals). In this work description logics provide the foundations for ontologies (see Section 3 for a detailed introduction). Description logics are used to define the syntax and semantics of so-called concept descriptions. Concept descriptions are related to one another by so-called axioms. Axioms are used to impose necessary and/or sufficient conditions on concepts. Due to the semantics of description logics, implicit information can be derived from what is explicitly given in the set of axioms. This is done by an inference system, also called reasoner for short. For example, an individual can be shown to be an instance of a given concept description. Other inference problems, which are implemented as reasoning services, are defined in Section 3.

Semantic content descriptions of multimedia documents are exploited at runtime for semantic retrieval of multimedia. The principal role of ontology evolution is to increase the performance of semantic retrieval. In this work we propose a methodology for ontology evolution that can be realized as a software architecture consisting of three major components:

- A *multimodal information extraction* engine: This information extraction engine is responsible for extracting instances of concept descriptions that can be directly identified in corpora of a specific modality. These concept descriptions are referred to as mid-level concepts, compared to low-level features such as edges, tokens, phonemes, etc. For example, in the text modality the *name* or the *age* of a person is represented as an individual which is an instance of a mid-level concept because instances of these concepts can be associated with relevant text portions. On the other hand, the concept *person* is not a mid-level concept, as it is a "compound" (or "aggregate") concept in such a way that instances of this concept are related to instances of name, age, gender, or maybe compound concepts. Compound concepts are referred to as high-level concepts, and instances of such concepts are not directly identifiable in a multimedia document.[2] Thus, such instances and also relationships between these instances have to be hypothesized.
- A *semantic interpretation* engine, responsible for hypothesizing instances of high-level concepts representing the interpretation of (parts of) a document: Semantic interpretation operates on the instances of mid-level concepts and relations between them extracted by the information extraction engine. The goal of semantic interpretation is to explain why certain instances of mid-level concepts are observed in certain relations according to the domain ontology and some set of interpretation rules by creating instances of high-level concepts and relating these instances.

[1] Funded by the European Commission under IST-FP6-027538, see http://www.boemie.org.

[2] One might argue that in texts, for instance, abstract notions such as *high jump* can appear. In our view, this is just a *name* for a sports competition, and not a high-level concept instance.

Semantic interpretation is performed through calls to a non-standard reasoning service (known as explanation derivation via abduction) and is formalised as a two-level process. During the first level, semantic interpretation is performed on the extracted information (mid-level concept instances/relations) from a single modality in order to form modality-specific high-level concept instances. At a second level, the modality-specific high-level instances are fused in order to produce high-level concept instances that are not modality-specific, and contain information extracted from all involved modalities. Fusion is also formalized as explanation generation via abductive reasoning.

– An *ontology evolution* toolkit, which uses the results obtained through reasoning in the interpretation phase in order to evolve (enhance) the ontology, through *population* of the ontology with individuals, or through *enrichment* of the ontology with new concepts and new relation types. Details of the evolution methodology are described in Section 2.

In this article, we describe an ontology evolution methodology and investigate the role of semantic interpretation for ontology evolution. The article analyses the results of interpretation, population and enrichment obtained in evaluation experiments in terms of measures such as precision and recall. The evaluation reveals encouraging results of the approach presented in this article.

The article is organized as follows. In Section 2, we present an overview of the methodology. In Section 3, we introduce the use of description logics for ontology formalization and semantic interpretation. In Section 4, we describe how reasoning techniques are used for multimedia interpretation, whereas in Section 5 we discuss the role of ontology matching techniques in ontology evolution. In Section 6, we describe the activities of population and enrichment of the ontology. In Section 7, we provide some experimental results. Related work and original contribution of the article are described in Section 8. Finally, in Section 9, we provide our concluding remarks and areas of future work.

2 A methodology for ontology evolution based on media interpretation

Ontology evolution in our approach uses as input the results of the semantic interpretation performed upon information extracted from multiple modalities (combined through fusion). In order to be able to deal with all possible situations requiring evolution, a pattern-driven approach is adopted. Typical input to the evolution toolkit is in the form of Aboxes, containing the results of the semantic interpretation of the fused extracted information from multimedia resources. The term Abox stands for assertional box and denotes a set of concept and role instance assertions (a formal definition will follow in Section 3). These results typically include instances of mid-level concepts, relations between mid-level concept instances, high-level instances, relations between instances of high-level concepts, and possibly instances of the specific mid-level concept "unknown" (indicating that low-level processes could not classify a

certain object). According to the information contained in incoming Aboxes, the evolution pattern selector selects the most prominent evolution pattern, triggering either ontology population or ontology enrichment, respectively.

The ontology evolution methodology is shown in Figure 1. Ontology population is the activity of adding new individuals into the ontology, and it is performed every time at least one explanation can be found for a multimedia resource through the interpretation process.

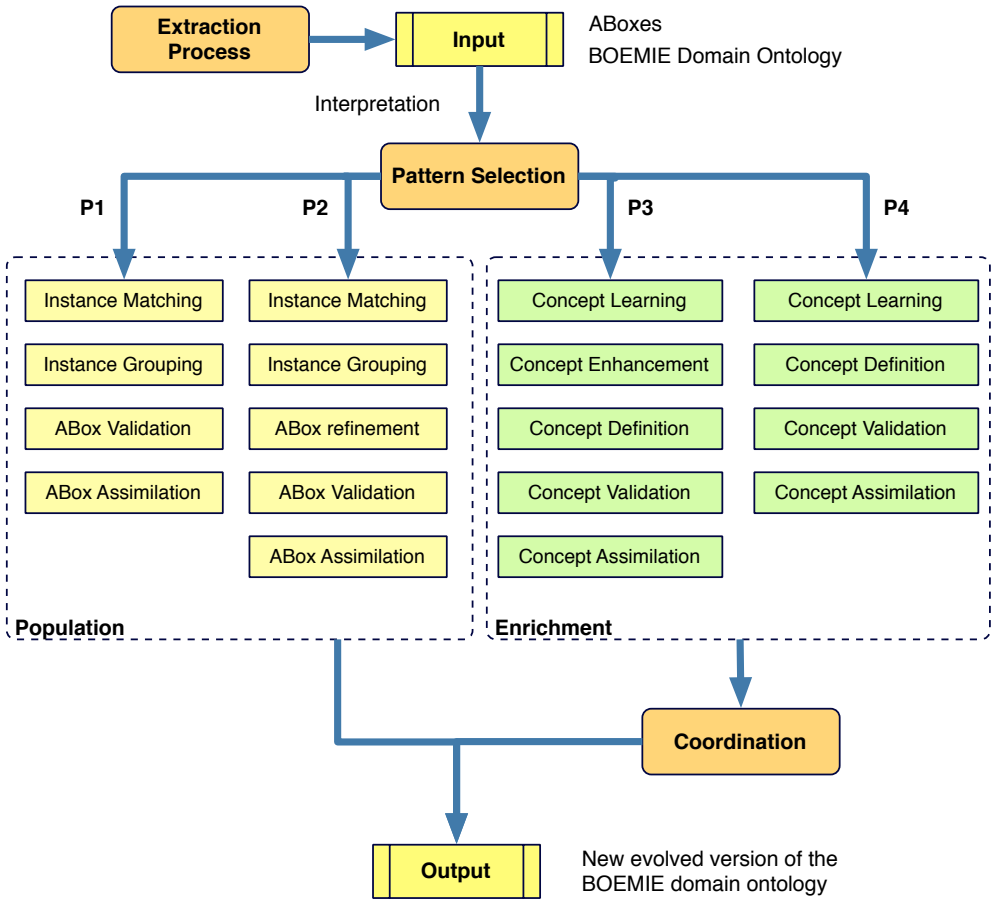

Fig. 1. The BOEMIE evolution methodology

Ontology enrichment is the activity of extending the ontology, through the addition of new elements (e.g., concepts, relations). Ontology enrichment is performed every time the background knowledge is not sufficient to explain the extracted information from the processed multimedia documents. Thus, the ontology enrichment activity is

expected to extend the background knowledge through the addition of new ontology elements.

Finally, a coordination activity is performed in order to produce a log of the changes introduced into the evolved version of the ontology with respect to the initial version. The goal of coordination is to align knowledge for several purposes, ranging from the detection of new instances referring to the same real entity in the domain, to the support of the expert in the activity of defining new concept definitions by exploiting external knowledge sources to suggest new concepts and relation names.

2.1 Evolution patterns

The evolution approach discussed in this article has emerged from two main requirements:

- The capability of classifying different situations that trigger ontology evolution in terms of the results of the semantic interpretation process (i.e., the information specified in the incoming Abox) w.r.t. the background knowledge;
- The definition of an appropriate activity articulation to correctly evolve the ontology in each specific evolution situation.

The desired result of the semantic interpretation is a single explanation for each information extracted from a multimedia document, that is, the extracted information is related to a single high-level concept instance. However, other situations can occur when the background knowledge leads to several explanations for the same extracted information or to the absence of explanations, meaning that no high-level concept can be found in the ontology for describing such an information. Finally, we can also have the case where not only the high-level concept describing the extracted information is missing, but also for one or more elements identified in the multimedia document a mid-level concept can not be assigned.

To take these requirements into account, four different *evolution patterns* have been identified for ontology evolution. An evolution pattern determines the characteristics of the input Abox it deals with, defines the kind of evolution process to be performed over the ontology (i.e., population or enrichment), and is articulated into a set of activities for implementing all the required changes. Population patterns (P1 and P2) tackle the situations where the interpretation has found one or more high-level concepts explaining an information extracted from the document and, thus, the corresponding Abox(es) are added to the ontology. Enrichment patterns (P3 and P4) describe the situations where no high-level concepts explaining the extracted information are found in the ontology, thus triggering ontology enrichment to acquire this missing knowledge. Pattern P4 has been conceived to deal with situations where not only the high-level concept is missing (like P3) but also one or more mid-level concepts are missing for the interpretation of the incoming information. In case of missing explanations for mid-level concept instances, pattern P4 is always selected as prominent, to first enrich the ontology with missing mid-level concepts, thus enabling the subsequent interpretation of all mid-level concept instances. Then, in a further interpretation cycle, the most suitable pattern will be chosen for handling the new interpretation results appropriately.

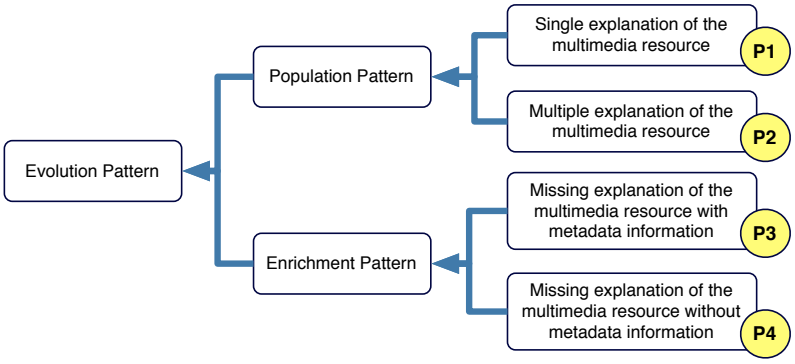

Fig. 2. BOEMIE evolution patterns

Note that Figure 2 depicts the evolution patterns for certain types of multimedia documents, e.g., images, where a single explanation maybe enough to explain the whole multimedia resource. For other types of multimedia such as a text document, where numerous information can be extracted and later explained with several explanations, one of the four patterns is selected for each extracted piece of information.

In the remainder of the paper, we focus on the reasoning activity and matching techniques for multimedia interpretation and on the activities of ontology population and enrichment, by providing some relevant examples of application in the athletics domain selected in BOEMIE.

3 Description logics as a basis for ontology languages

Ontology languages are based on description logics [3]. For studying dynamic ontology evolution we focus on the description logics $\mathcal{ALCQ}$, which corresponds to a large fragment of standard ontology languages such as OWL.

3.1 Syntax and semantics of $\mathcal{ALCQ}$

For a given application problem one chooses a set of elementary descriptions (or *atomic descriptions*) for *concepts* and *roles* representing unary and binary predicates, respectively. A set of *individuals* is fixed to denote specific objects of a certain domain (e.g., athletics). For instance, *Athlete* might be an atomic concept description and *hasParticipant* is an atomic role description in the athletics domain.

In the following, we use letters A and R for atomic concept and role descriptions, respectively. In addition, let $\{i, j, \ldots\}$ be the set of individuals. In $\mathcal{ALCQ}$ (Attributive Language with full Complement and Qualified number restrictions), descriptions for *complex concepts* C or D can be inductively built using the following grammar:
For instance, $Event \sqcap \exists_{\leq 1} hasParticipant.Athlete$ is a complex concept description in the *athletics* domain.

$$
\begin{array}{rll}
C, D \longrightarrow & A & \mid \text{atomic concept} \\
& C \sqcap D & \mid \text{conjunction} \\
& C \sqcup D & \mid \text{disjunction} \\
& \neg C & \mid \text{negation} \\
& \exists R.C & \mid \text{existential restriction} \\
& \forall R.C & \mid \text{value restriction} \\
& \exists_{\leq n} R.C & \mid \text{qualified minimum restriction} \\
& \exists_{\geq n} R.C & \mid \text{qualified maximum restriction}
\end{array}
$$

We introduce the concept descriptions $\top$ and $\bot$ as abbreviations for $A \sqcup \neg A$ and $A \sqcap \neg A$, respectively. Concept descriptions may be written in parentheses in order to avoid scoping ambiguities.

In order to define the semantics of concept and role descriptions we consider *interpretations* $\mathcal{I}$ that consist of a non-empty set $\Delta^{\mathcal{I}}$, the domain, and an interpretation function $\cdot^{\mathcal{I}}$, which assigns to every atomic concept description A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every (atomic) role R a set $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. For complex concept descriptions, the interpretation function is extended as follows:

$$
\begin{aligned}
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \{x \mid \exists y.(x, y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} &= \{x \mid \forall y. \text{ if } (x, y) \in R^{\mathcal{I}} \text{ then } y \in C^{\mathcal{I}}\} \\
(\exists_{\leq n} R.C)^{\mathcal{I}} &= \{x \mid \sharp\{y \mid (x, y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leq n\} \\
(\exists_{\geq n} R.C)^{\mathcal{I}} &= \{x \mid \sharp\{y \mid (x, y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geq n\}
\end{aligned}
$$

The semantics of description logics is based on the notion of satisfiability. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *satisfies* a concept description C if $C^{\mathcal{I}} \neq \emptyset$. In this case, $\mathcal{I}$ is called a *model* for C.

A *Tbox* is a set of so-called *generalized concept inclusions* $C \sqsubseteq D$ (e.g., $HighJump \sqsubseteq SportsTrial$). For brevity the elements of a Tbox are called *GCIs*.

An interpretation $\mathcal{I}$ *satisfies* a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation is a *model* of a Tbox if it satisfies all GCIs in the TBox. A concept description C *is subsumed by* a concept description D w.r.t. a Tbox if the GCI $C \sqsubseteq D$ is satisfied in all models of the Tbox. In this case, we also say that D *subsumes* C.

An *Abox* is a set of *assertions* of the form $i : C$ or $(i, j) : R$ where C is a concept description, R is a role description, and i, j are individuals. A concept assertion $i : C$ is satisfied w.r.t. a Tbox $\mathcal{T}$ if for all models $\mathcal{I}$ of $\mathcal{T}$ it holds that $i^{\mathcal{I}} \in C^{\mathcal{I}}$. A role assertion $(i, j) : R$ is satisfied w.r.t. a Tbox $\mathcal{T}$ if $(i^{\mathcal{I}}, j^{\mathcal{I}}) \in R^{\mathcal{I}}$ for all models $\mathcal{I}$ of $\mathcal{T}$. An interpretation satisfying all assertions in an Abox $\mathcal{A}$ is called a model for $\mathcal{A}$. An Abox $\mathcal{A}$ is called *consistent* if such a model exists, it is called *inconsistent* otherwise.

An *ontology* Σ is a tuple $(\mathcal{T}, \mathcal{A})$ with Tbox $\mathcal{T}$ and Abox $\mathcal{A}$. Let α be concept or role assertion. An ontology Σ *entails* an assertion α (α follows from Σ), denoted as $\Sigma \models \alpha$ if for all models $\mathcal{I}$ of Σ it holds that $\mathcal{I}$ satisfies α. Let $\mathcal{A}$ be an Abox. An ontology

Σ *entails* an Abox, denoted as $\Sigma \models \mathcal{A}$, if for all $\alpha \in \mathcal{A}$ it holds that $\Sigma \models \alpha$. In the following sections we slightly misuse notation and assume that $(\mathcal{T}, \mathcal{A}) \cup \mathcal{A}'$ means $(\mathcal{T}, \mathcal{A} \cup \mathcal{A}')$.

3.2 Decision problems and their reductions

The definitions given in the previous section can be paraphrased as decision problems. The *concept satisfiability* problem is to check whether a model for a concept description exists. The Tbox satisfiability problem is to check whether a model for the Tbox exists. The *concept subsumption* problem is to check whether $C \sqsubseteq D$ holds in all models of the Tbox.

The *Abox consistency problem* for an Abox $\mathcal{A}$ (w.r.t. a Tbox) is the problem of determining whether there exists a model of $\mathcal{A}$ (that is also a model of the Tbox). Another problem is to test whether an individual i is an instance of a concept description C w.r.t. a Tbox and an Abox (*instance test* or *instance problem*: $\Sigma \models i : C$). The *instance retrieval* problem w.r.t. a query to the query concept C and the ontology $\Sigma$is to find all individuals i mentioned in the assertions of an Abox such that i is an instance of C. For roles and pairs of individuals, similar definitions can be given.

The latter problem is a retrieval problem but, in theory, it can be reduced to several instance problems. In order to solve the instance problem for an individual i and a concept description C one can check if the Abox $\{i : (\neg C)\}$ is inconsistent [4]. Furthermore, the satisfiability problem for a concept description C can be reduced to the consistency problem for the Abox $\{i : C\}$. Thus, in theory, all problems introduced above can be reduced to the Abox consistency problem. In practical systems, specific optimization techniques are used to decide a certain decision problem.

3.3 Sequences, variable substitutions, transformations

For the introduction of the interpretation algorithm, we need some additional definitions. A *variable* is a name of the form *?name* where name is a string of characters from $\{a..z\}$. In the follow definitions, we denote places where variables can appear with uppercase letters.

Let V be a set of variables, and let $\underline{X}, \underline{Y_1}, \ldots, \underline{Y_n}$ be sequences $\langle \ldots \rangle$ of variables from V. $\underline{Z}$ denotes a sequence of individuals. We consider sequences of length 1 or 2 only, if not indicated otherwise, and assume that $(\langle X \rangle)$ is to be read as (X) and $(\langle X, Y \rangle)$ is to be read as (X, Y) etc. Furthermore, we assume that sequences are automatically flattened. A function *as_set* turns a sequence into a set in the obvious way.

A *variable substitution* $\sigma = [X \leftarrow i, Y \leftarrow j, \ldots]$ is a mapping from variables to individuals. The application of a variable substitution σ to a sequence of variables $\langle X \rangle$ or $\langle X, Y \rangle$ is defined as $\langle \sigma(X) \rangle$ or $\langle \sigma(X), \sigma(Y) \rangle$, respectively, with $\sigma(X) = i$ and $\sigma(Y) = j$. In this case, a sequence of individuals is defined. If a substitution is applied to a variable X for which there exists no mapping $X \leftarrow k$ in σ then the result is undefined. A variable for which all required mappings are defined is called *admissible* (w.r.t. the context).

3.4 Grounded conjunctive queries

Let $\underline{X}, \underline{Y_1}, \underline{Y_n}$ be sequences of variables, and let $Q_1, \ldots, Q_n$ denote atomic concept or role descriptions.

A query is defined by the following syntax.

$$\{(\underline{X}) \mid Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})\}$$

The sequence $\underline{X}$ may be of arbitrary length but all variables mentioned in $\underline{X}$ must also appear in at least one of the $\underline{Y_1}, \cdots, \underline{Y_n}$: $as\_set(\underline{X}) \subseteq as\_set(\underline{Y_1}) \cup \cdots \cup as\_set(\underline{Y_n})$.

Informally speaking, $Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$ defines a conjunction of so-called *query atoms* $Q_i(\underline{Y_i})$. The list of variables to the left of the sign $\mid$ is called the *head* and the atoms to the right of are called the query *body*. The variables in the head are called distinguished variables. They define the query result. The variables that appear only in the body are called non-distinguished variables and are existentially quantified.

Answering a query with respect to an ontology Σ means finding admissible variable substitutions σ such that $\Sigma \models \{(\sigma(\underline{Y_1})) : Q_1, \ldots, (\sigma(\underline{Y_n})) : Q_n\}$. We say that a variable substitution $\sigma = [X \leftarrow i, Y \leftarrow j, \ldots]$ introduces *bindings* $i, j, \ldots$ for variables $X, Y, \ldots$. Given all possible variable substitutions σ, the *result* of a query is defined as

$$\{(\sigma(\underline{X}))\}$$

Note that the variable substitution σ is applied before checking whether $\Sigma \models \{(\sigma(\underline{Y_1})) : Q_1, \ldots, (\sigma(\underline{Y_n})) : Q_n\}$, i.e., the query is *grounded* first.

For a query $\{(?x) \mid Person(?x), hasParticipant(?y, ?x)\}$ and the Abox $\Gamma_1 = \{ind_1 : HighJump,\ ind_2 : Person,\ (ind_1,\ ind_2) : hasParticipant\}$, the substitution $[?x \leftarrow ind_2, ?y \leftarrow ind_1]$ allows for answering the query, and defines bindings for $?x$.

A *boolean* query is a query with $\underline{X}$ being of length zero. If for a boolean query there exists a variable substitution σ such that $\Sigma \models \{(\sigma(\underline{Y_1})) : Q_1, \ldots, (\sigma(\underline{Y_n})) : Q_n\}$ holds, we say that the query is answered with *true*, otherwise the answer is *false*.

Later on, we will have to convert query atoms into Abox assertions. This is done with the function *transform*. The function *transform* applied to a set of query atoms $\{\gamma_1, \ldots \gamma_n\}$ is defined as $\{transform(\gamma_1, \sigma), \ldots, transform(\gamma_n, \sigma)\}$ where
$transform(P(\underline{X}), \sigma) := (\sigma(\underline{X})) : P$.

3.5 Rules

A rule r has the following form

$$P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})$$

where P, $Q_1, \ldots, Q_n$ denote atomic concept or role descriptions with the additional restriction that $as\_set(\underline{X}) \subseteq as\_set(\underline{Y_1}) \cup \cdots \cup as\_set(\underline{Y_n})$.

Rules are used to derive new Abox assertions, and we say that a rule r is *applied* to an Abox $\mathcal{A}$. The function application $apply(\Sigma, P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n}), \mathcal{A})$

returns a set of Abox assertions $\{(\sigma(\underline{X})) : P\}$ if there exists an admissible variable substitution σ such that the answer to the query

$$\{() \mid Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n})\}$$

is *true* with respect to $\Sigma \cup \mathcal{A}$. If no such σ can be found, the result of the call to $apply(\Sigma, r, \mathcal{A})$ is the empty set. The application of a set of rules $\mathcal{R} = \{r_1, \ldots r_n\}$ to an Abox is defined as follows.

$$apply(\Sigma, \mathcal{R}, \mathcal{A}) = \bigcup_{r \in \mathcal{R}} apply(\Sigma, r, \mathcal{A})$$

The result of $forward\_chain(\Sigma, \mathcal{R}, \mathcal{A})$ is $\emptyset$ if $apply(\Sigma, \mathcal{R}, \mathcal{A}) \cup \mathcal{A} = \mathcal{A}$ and $apply(\Sigma, \mathcal{R}, \mathcal{A}) \cup$ $forward\_chain(\Sigma, \mathcal{R}, \mathcal{A} \cup apply(\Sigma, \mathcal{R}, \mathcal{A}))$ otherwise.

3.6 Computing explanations via abduction

We assume that a set of rules $\mathcal{R}$ as defined above is specified, and define a non-deterministic function *compute_explanation* as follows.

- $compute\_explanation(\Sigma, \mathcal{R}, \mathcal{A}, (\underline{Z}) : P) = transform(\Phi, \sigma)$ if there exists a rule $r = P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n}) \in \mathcal{R}$ such that a set of query atoms Φ and an admissible variable substitution σ with $\sigma(\underline{X}) = \underline{Z}$ can be found, and the query

  $$Q := \{() \mid expand(P(\underline{X}), r, \mathcal{R}) \setminus \Phi\}$$

  is answered with *true*.
- If no such rule r exists in $\mathcal{R}$ it holds that $compute\_explanation(\Sigma, \mathcal{R}, \mathcal{A}, (\underline{Z}) : P) = \emptyset$.

The goal of the function *compute_explanation* is to determine what must be added (Φ) such that an entailment $\Sigma \cup \mathcal{A} \cup \Phi \models (\underline{Z}) : P$ holds. Hence, for *compute_explanation*, abductive reasoning is used. The set of query atoms Φ defines what must be hypothesized in order to answer the query Q with *true* such that

$$\Phi \subseteq expand(P(\underline{X}), r, \mathcal{R})$$

holds. The definition of *compute_explanation* is non-deterministic due to several possible choices for Φ.

The function application

$$expand(P(\underline{X}), P(\underline{X}) \leftarrow Q_1(\underline{Y_1}), \ldots, Q_n(\underline{Y_n}), \mathcal{R})$$

is also defined in a non-deterministic way as

$$expand'(Q_1(\underline{Y_1}), \mathcal{R}) \cup \cdots \cup expand'(Q_n(\underline{Y_n}), \mathcal{R})$$

with $expand'(P(\underline{X}), \mathcal{R})$ being $expand(P(\underline{X}), r, \mathcal{R})$ if there exist a rule $r = P(\underline{X}) \leftarrow \ldots \in \mathcal{R}$ and $\langle P(\underline{X}) \rangle$ otherwise. We say the set of rules is backward-chained, and since there might be multiple rules in $\mathcal{R}$, backward-chaining is non-deterministic.

3.7 Interpreting Aboxes in terms of rules

In the following we devise an abstract computational engine for "interpreting" Abox assertions in terms of a given set of rules. Interpretation in this sense is not to be confused with the interpretation of a concept description (which is defined as a set of objects from the domain). Interpretation of Abox assertions w.r.t. a set of rules is meant in the sense that using the rules some high-level explanation is constructed such that the Abox assertions are entailed. The interpretation of an Abox is again an Abox. For instance, the output Abox might represent results of a content interpretation process (see below for an example).

Let Γ be an Abox whose assertions are to be interpreted. The goal of the interpretation process is to use a set of rules $\mathcal{R}$ to derive "explanations" for elements in Γ. The definition of $requires\_fiat$ depends on the application context. For our multimedia interpretation scenario (see Section 4), we might use the following definition.

$$requires\_fiat((\underline{X}) : P) = true \text{ iff } P \in \{near, adjacent\_to, \ldots\}$$

The interpretation algorithm implemented by the interpretation engine works on a set of (possible) interpretations $\mathfrak{I}$, i.e., a set of Aboxes. Initially, $\mathfrak{I} \Leftarrow \{\Gamma\}$, i.e. , at this stage, the interpretation is just the input Abox Γ.[3] The function *interpret* is applied to an Abox Γ and applies a strategy function Ω in order to decide which assertion to interpret, and uses a termination function Ξ in order to check whether to terminate due to resource constraints.

function $interpret(\Omega, \Xi, \Sigma, \mathcal{R}, S, \Gamma)$:
  $\mathfrak{I}' \Leftarrow \{\Gamma\}$
  repeat
    $\mathfrak{I} \Leftarrow \mathfrak{I}'$
    $(\mathcal{A}, \alpha) \Leftarrow \Omega(\mathfrak{I})$ // $\mathcal{A} \in \mathfrak{I}$, $\alpha \in \mathcal{A}$ s.th. $requires\_fiat(\alpha)$ holds
    $\mathfrak{I}' \Leftarrow (\mathfrak{I} \setminus \{\mathcal{A}\}) \cup interpretation\_step(\Sigma, \mathcal{R}, S, \mathcal{A}, \alpha)$.
  until $\Xi(\mathfrak{I})$ or no $\mathcal{A}$ and α can be selected such that $\mathfrak{I}' \neq \mathfrak{I}$
  return $\mathfrak{I}$

The function Ω for the interpretation strategy and Ξ for the termination condition are used as an oracle and must be defined in an application-specific way.

The function $interpretation\_step(\Sigma, \mathcal{R}, S, \mathcal{A}, \alpha)$ is defined as

$$\bigcup_{\Delta \in compute\_all\_explanations(\Sigma, \mathcal{R}, S, \mathcal{A}, \alpha)} \{\Delta \cup \mathcal{A} \cup forward\_chain(\Sigma, \mathcal{R}, \Delta \cup \mathcal{A})\}.$$

The function $compute\_all\_explanations(\Sigma, \mathcal{R}, S, \mathcal{A}, \alpha)$ is defined as

$$maximize(\{\Delta \mid \Delta = compute\_explanation(\Sigma, \mathcal{R}, \mathcal{A}, \alpha)\}, S).$$

We impose restrictions on the choice of the Δ's returned by *interpret*. In particular, a scoring function S evaluates an explanation Δ. The function *maximize* select those

[3] $\Leftarrow$ denotes the assignment operator

Δ's for which the score $S(\Delta)$ is maximal, i.e., there exists no other $\Delta' \in \Im$ s.th. $S(\Delta') > S(\Delta)$. The scoring function has to be defined in an application-dependent way.

4 Reasoning for multimedia interpretation

After the discussion of description logics and reasoning techniques, in this section, we start with describing the application of these techniques for multimedia interpretation. Later, we present the stepwise interpretation of an example multimedia document in detail.

4.1 Interpretation of multimedia documents

Abductive reasoning is usually defined as a form of reasoning from effects to causes. Multimedia interpretation can be achieved through reasoning, in particular through abduction where we reason from observations (effects) to explanations (causes). In this view, abduction aims to find explanations for observations. In general, abduction is formalized as

$$\Sigma \cup \Delta \models \Gamma \tag{1}$$

where background knowledge (Σ), and observations (Γ) are given and explanations (Δ) are to be computed. If description logics are used as the underlying knowledge representation formalism, Δ and Γ are Aboxes and the background knowledge Σ is an ontology that consists of a Tbox $\mathcal{T}$ and an Abox $\mathcal{A}$.

We consider Abox abduction in description logics, the *compute_explanation* function in Section 3.6, as the key inference service for multimedia interpretation. For this purpose we modify the previous equation to:

$$\Sigma \cup \Gamma_1 \cup \Delta \models_{\mathcal{R}} \Gamma_2 \tag{2}$$

by splitting the assertions in Γ into two parts: *bona fide assertions* (Γ_1) and assertions requiring fiats (Γ_2). Bona fide assertions are assumed to be true by default, whereas *fiat assertions* are aimed to be explained. In order to compute explanations, Abox abduction can be implemented as a non-standard retrieval inference service in description logics. Different from the standard retrieval inference services, answers to a given query cannot be found by simply exploiting the ontology. In fact, the abductive retrieval inference service has the task of acquiring what should be added to the ontology in order to positively answer a query. In order to make this practical it has to be defined what can be abduced. In our approach the space of abducibles is defined by a set of rules.

The $\models_{\mathcal{R}}$ symbol in Formula 2 reflects the fact that a set of rules $\mathcal{R}$ is necessary for deriving explanations.

In practice, one is not interested in retrieving every consistent explanation, but the most preferred explanation for every query. To achieve this goal, we transform the set Δ's into a poset according to a preference score. The preference score reflects the two

criteria proposed by Thagard for selecting explanations [5], namely simplicity and consilience. I.e., the less hypothesized assertions an explanation contains (simplicity) and the more ground assertions (observations) an explanation involves (consilience), the higher its preference score. The following formula to compute the preference score of each explanation has been implemented: $S(\Sigma, \Gamma_1, \Delta) := S_f(\Sigma, \Gamma_1, \Delta) - S_h(\Sigma, \Gamma_1, \Delta)$. In this formula S_f represents the number of assertions in the explanation (Δ) that follow from $\Sigma \cup \Gamma_1$, and S_h represents the number of assertions in the explanation that do not follow from the same set, and therefore are hypothesized. Thus, S_f and S_h can be defined as follows:

$$S_f(\Sigma, \Gamma_1, \Delta) := \sharp\{\alpha \in \Delta \mid \Sigma \cup \Gamma_1 \models \alpha\}$$
$$S_h(\Sigma, \Gamma_1, \Delta) := \sharp\{\alpha \in \Delta \mid \Sigma \cup \Gamma_1 \not\models \alpha\}$$

In some cases, more than one explanation can have the highest preference score. In order to prefer one of the explanations with the highest score, these explanations can be considered as separate Aboxes and compared for entailment. If one of the explanations with the highest preference score entails others explanations with the highest preference score, it is preferred to others because it is the most-specific explanation.

For example, assume that for a given query the abductive retrieval inference service generates three consistent explanations that have the following preference scores: $S(\Sigma, \Gamma_1, \Delta_1)$=2, $S(\Sigma, \Gamma_1, \Delta_2)$=2, $S(\Sigma, \Gamma_1, \Delta_3)$=-1. In the first step, the set of explanations is transformed into a poset and all explanations with a score lower than the highest score found in the poset are discarded. In our example multiple explanations, namely Δ_1 and Δ_2, have the highest score. Therefore in the second step, these explanations are transformed into separate Aboxes and compared for entailment. Assuming that following entailment relation exists between these explanations: $\Sigma \cup \Delta_1 \models \Delta_2$ the abductive retrieval inference service returns Δ_1 as the most preferred explanation, due to the fact that Δ_1 is a more specific and, thus, more valuable explanation than Δ_2.

As discussed in Section 3.7 with the introduction of an abstract interpretation engine, rules can be exploited for interpreting Aboxes (e.g. so-called analysis Aboxes describing information extracted from a multimedia document). In the following a more detailed definition of the *interpret* function (see Section 3.7), is given for multimedia interpretation (Algorithm 1).

Algorithm 1 $interpret\_multimedia(\Sigma, \mathcal{R}, \Gamma_a)$

Input: *ontology*, *rules*, *analysis Abox* : $\Sigma, \mathcal{R}, \Gamma_a$
Output: *set of interpretation Aboxes* : $\{\Gamma_{i1}, ..., \Gamma_{in}\}$
  $\Gamma_2 \Leftarrow select\_fiat\_assertions(\Gamma_a)$
  $\Gamma_1 \Leftarrow \Gamma_a \setminus \Gamma_2$
  return *derive_explanations* $(\Sigma, \mathcal{R}, \Gamma_1, \Gamma_2)$

The *interpret_multimedia* algorithm takes an ontology, a set of interpretation rules and a multimedia analysis Abox as input and returns a set of interpretation Aboxes

as output. The function *select_fiat_assertions* uses the *requires_fiat* predicate introduced in Section 3.7 as a filter to select the fiat assertions from the analysis Abox. In general, all assertions from the analysis Abox can be considered as fiat assertions. However, the set of interpretation rules alone defines the space of abducibles (and the explanations that can be derived). Therefore, in the practical implementation the *select_fiat_assertions* function selects only those assertions for which interpretation rules exist. All assertions of the analysis Abox that do not require fiats become bona fide assertions. Finally, the *derive_explanations* algorithm (Algorithm 2) is called to compute all interpretations for the given multimedia document.

The *derive_explanations* algorithm takes an ontology, a set of interpretation rules, a set of bona fide assertions and a set of fiat assertions as input. For each fiat assertion it computes all explanations. The fiat assertion is then removed from the set of fiat assertions and added to the set of bona fide assertions. Afterwards for each explanation, first, all assertions of the explanation are added to the set of bona fide assertions. Second, the *derive_explanations* is called for the new set of fiat assertions. The new set of fiat assertions is the union of the current set of fiat assertions and the assertions that are inferred by the function $forward\_chain$ from the ontology, the set of rules and the current set of bona fide assertions. Finally, all bona fide assertions are collected in a set and returned as the set of interpretation Aboxes.

Algorithm 2 *derive_explanations*$(\Sigma, \mathcal{R}, \Gamma_1, \Gamma_2)$

Input: *ontology*, *rules*, *bona fide assertions*, *fiat assertions* : $\Sigma, \mathcal{R}, \Gamma_1, \Gamma_2$
Output: *set of interpretation Aboxes* : $\mathfrak{I} = \{\Gamma_{i1}, ..., \Gamma_{in}\}$

```
ℑ = ∅
while Γ2 ≠ ∅ do
  for each α ∈ Γ2 do
    {Δ1, ..., Δn} ⇐ compute_all_explanations (Σ, R, S, Γ1, α)
    Γ1 ⇐ Γ1 ∪ {α}
    Γ2 ⇐ Γ2 \ {α}
    for each Δi ∈ {Δ1, ..., Δn} do
      Γ1 ⇐ Γ1 ∪ Δi
      ℑ ⇐ ℑ ∪ derive_explanations (Σ, R, Γ1, Γ2 ∪ forward_chain(Σ, R, Γ1))
    end for
  end for
end while
ℑ ⇐ ℑ ∪ {Γ1}
return ℑ
```

Note that the *derive_explanations* algorithm is called recursively for each newly computed explanation. In general, there is no guarantee for the termination of this algorithm for arbitrary ontologies and sets of interpretation rules. However, our current implementation of the *interpret_multimedia* always terminates in a practical setting due to the definition of the rules we use (see below). Thus, we introduce no specific strategy function Ω and no specific termination criterion Ξ here.

4.2 An interpretation example

In this subsection we discuss the stepwise interpretation of multimedia documents with an example document. Generally speaking, multimedia documents contain information in different modalities such as image, text, audio or video. These information can be extracted and interpreted by modality-specific analysis and interpretation processes.

We proceed with the interpretation of a sample multimedia document to discuss the details of the interpretation process. Figure 3 shows an image and its caption taken from a web page with athletics news. It contains information both in visual and textual modality. The underlined words in Figure 3 are key words of this text, which have to be detected by the text analysis process. Similarly, several objects (like a person body, person face, a horizontal bar, etc.) have to be identified by the image analysis process. Due to space limitations, in this article, we will focus on the stepwise interpretation of the textual part of the document.

The analysis results for the textual part of the document in Figure 3 are represented in an Abox, which is shown in Figure 4. To continue with the interpretation example we assume that the ontology contains the axioms shown in Figure 5. In Figure 6 a set of rules for the interpretation of the athletics example is specified.

'13 August 2002 - Helsinki. Russia's newly crowned European champion Jaroslav Rybakov won the high jump with 2.29 m. Oskari Fronensis from Finland cleared 2.26 and won silver.'

Fig. 3. Sample multimedia document.

In the following the set of rules shown in Figure 6 define the space of abducibles. For the sake of brevity the Tbox and the set of rules show only a small excerpt of

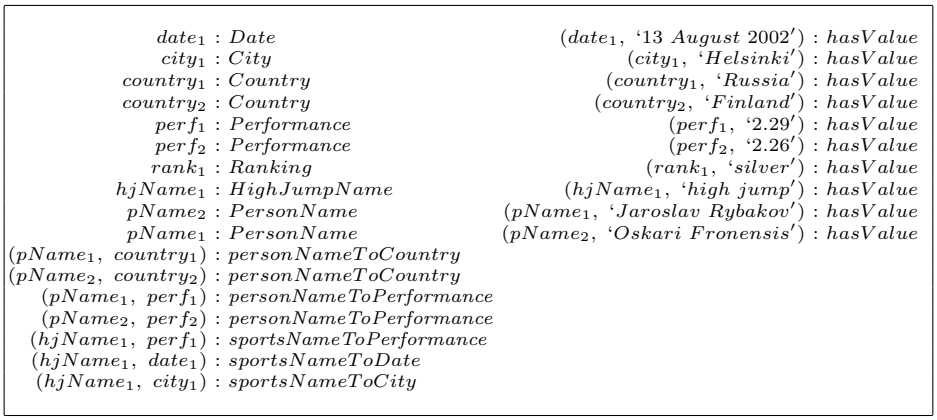

Fig. 4. Abox representing the results of text analysis.

the BOEMIE athletics ontology, which is relevant for the text interpretation example discussed here.

To construct an interpretation for a multimedia document w.r.t the text modality, explanations are searched to find out why some words are related with some other words. Such explanations are then used to construct interpretation(s) of the textual part of a multimedia document. The multimedia interpretation algorithm presented in Section 4.1 has been implemented to generate explanations and, therefore, constitutes the foundation of the semantic interpretation engine.

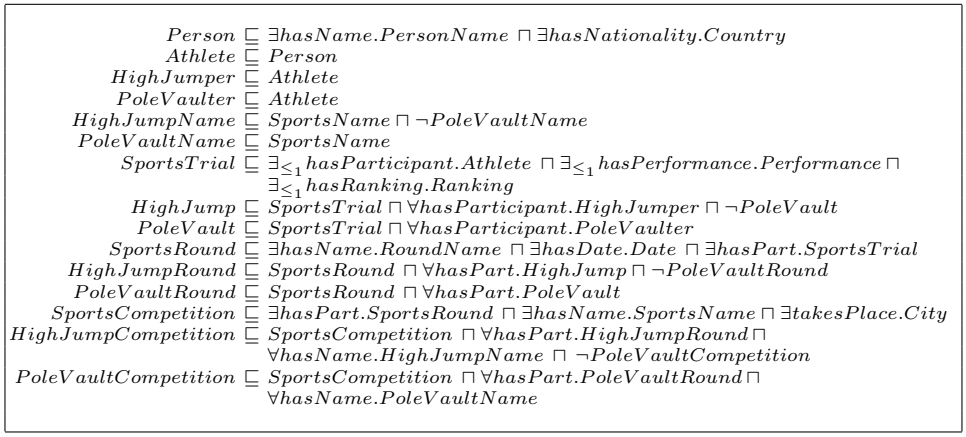

Fig. 5. An example Tbox Σ for the athletics domain.

$$
\begin{aligned}
personNameToCountry(X,Y) \leftarrow\ & Person(Z),\\
& hasPersonName(Z,X), PersonName(X),\\
& hasNationality(Z,Y), Country(Y).\\
personToPerformance(X,Y) \leftarrow\ & Person(X),\\
& hasPersonName(X,Z), PersonName(Z),\\
& personNameToPerformance(Z,Y).\\
personToPerformance(X,Y) \leftarrow\ & SportsTrial(Z),\\
& hasParticipant(Z,X), Athlete(X),\\
& hasPerformance(Z,Y), Performance(Y).\\
sportsNameToCity(X,Y) \leftarrow\ & HighJumpCompetition(Z),\\
& hasSportsName(Z,X), HighJumpName(X),\\
& takesPlace(Z,Y), City(Y).\\
sportsNameToCity(X,Y) \leftarrow\ & PoleVaultCompetition(Z),\\
& hasSportsName(Z,X), PoleVaultName(X),\\
& takesPlace(Z,Y), City(Y).\\
sportsNameToDate(X,Y) \leftarrow\ & HighJumpCompetition(Z),\\
& hasSportsName(Z,X), HighJumpName(X),\\
& hasDate(Z,Y), Date(Y).\\
sportsNameToDate(X,Y) \leftarrow\ & PoleVaultCompetition(Z),\\
& hasSportsName(Z,X), PoleVaultName(X),\\
& hasDate(Z,Y), Date(Y).\\
sportsCompetitionToPerformance(X,Y) \leftarrow\ & SportsCompetition(X),\\
& hasSportsName(X,Z), SportsName(Z),\\
& sportsNameToPerformance(Z,Y).\\
sportsCompetitionToPerformance(X,Y) \leftarrow\ & HighJumpCompetition(X),\\
& hasPart(X,Z), HighJumpRound(Z),\\
& hasPart(Z,W), HighJump(W)\\
& hasPerformance(W,Y).\\
sportsCompetitionToPerformance(X,Y) \leftarrow\ & PoleVaultCompetition(X),\\
& hasPart(X,Z), PoleVaultRound(Z),\\
& hasPart(Z,W), PoleVault(W)\\
& hasPerformance(W,Y).
\end{aligned}
$$

Fig. 6. Text interpretation rules for the athletics domain.

To start with the interpretation of the text paragraph in Figure 3, text analysis results for this text paragraph, namely the Abox in Figure 4, is considered as Γ. According to the Formula 2 we divide Γ into a part Γ_2 that the agent would like to have explained (fiat assertions), and a part Γ_1 that the interpretation agent takes as granted (bona fide assertions). In our example, Γ_2 contains the following assertions:

$(hjName_1,\ date_1) : sportsNameToDate,$
$(pName_1,\ perf_1) : personNameToPerformance,$
$(hjName_1,\ city_1) : sportsNameToCity,$
$(pName_2, perf_2) : personNameToPerformance,$
$(pName_1,\ country_1) : personNameToCountry,$
$(hjName_1,\ perf_1) : sportsNameToPerformance,$
$(pName_2,\ country_2) : personNameToCountry.$

In the first step, these assertions are transformed into corresponding queries and the abductive retrieval inference service is asked for explanations. For example, from the role assertion $(hjName_1,\ date_1) : sportsNameToDate$ the following query is derived:

$$Q_1 := \{() \mid sportsNameToDate(hjName_1,\ date_1)\}$$

The task of the abductive retrieval inference service is to compute what should be added to the ontology in order to answer this query with true. In the given set of rules (see Figure 6), two rules have the atom *sportsNameToDate* in the rule head (consequences). Therefore, both rules are applied in a backward chaining way (i.e. from left to right) and corresponding terms are unified and we get variable bindings for X and Y. The unbound variable Z is instantiated with fresh individuals (e.g. *new_ind_1*). Note that for one of these rules, namely for the one that hypothesizes a pole vault competition, all bindings that are found for Y produce explanations (Δ's) that are inconsistent w.r.t. Σ. This is caused by the disjointness axioms in the Tbox (e.g. the concepts *HighJumpName* and *PoleVaultName* are disjoint). The abductive retrieval service discards inconsistent explanations. Therefore, the generated explanation to answer Q_1 with true is:

$$\Delta = \{new\_ind_1 : HighJumpCompetition,\ (new\_ind_1,\ hjName_1) : hasSportsName,\\ (new\_ind_1,\ date_1) : hasDate\}$$

The assertions shown in Δ are added to Γ_1. Furthermore, the assertion $(hjName_1,\ date_1) : sportsNameToDate$ is removed from Γ_2 and added to Γ_1. This procedure is applied to the remaining assertions in Γ_2 until Γ_2 is empty. At the end of the first interpretation step, Γ_1 contains (beside the assertions shown in Figure 4) following newly created assertions:

$new\_ind_1 : HighJumpCompetition,\ (new\_ind_1,\ hjName_1) : hasSportsName,$
$(new\_ind_1,\ date_1) : hasDate, (new\_ind_1,\ city_1) : takePlace, new\_ind_2 : Person,$
$(new\_ind_2,\ pName_1) : hasPersonName, (new\_ind_2,\ country_1) : hasNationality,$
$new\_ind_3 : Person, (new\_ind_3,\ pName_2) : hasPersonName,$
$(new\_ind_3,\ country_2) : hasNationality$

Note that the preference score presented in Section 4.1 guarantees that explanations that involve less hypothesized assertions and more observations are preferred. This is why Γ_1 contains a single *HighJumpCompetition* instance at the end of the first interpretation step.

In the second step, the interpretation process applies the set of rules in a forward chaining way (from right to left, i.e., from antecedent to consequence) to check whether new information can be deduced. This yields the following assertions:

$(new\_ind_2,\ perf_1) : personToPerformance,\ (new\_ind_3,\ perf_2) : personToPerformance$
$(new\_ind_1,\ perf_1) : sportsCompetitionToPerformance$

which are also added to Γ_1. At this state, the interpretation process defines a new Γ_2 by selecting all newly inferred assertions as fiat assertions and starting a new iteration. I.e., the first interpretation step is applied to the assertions in the new Γ_2. At the end of this step, following newly created assertions are added to Γ_1:

$new\_ind_4 : HighJumpRound,\ (new\_ind_1,\ new\_ind_4) : hasPart,$
$new\_ind_5 : HighJump,\ (new\_ind_4,\ new\_ind_5) : hasPart,$
$(new\_ind_5,\ perf_1) : hasPerformance,\ (new\_ind_5,\ new\_ind_2) : hasParticipant,$
$new\_ind_6 : SportsTrial,\ (new\_ind_6,\ new\_ind_3) : hasParticipant,$
$(new\_ind_6,\ perf_2) : hasPerformance$

In the second step of the second iteration no new information can be deduced by applying the set of rules in a forward chaining way. Therefore, the interpretation process terminates by returning the current Γ_1 as the interpretation Abox. The interpretation Abox contains (beside the assertions in Figure 4) the following newly inferred assertions:

$new\_ind_1 : HighJumpCompetition,\ new\_ind_2 : Person, new\_ind_3 : Person,$
$new\_ind_4 : HighJumpRound,\ new\_ind_5 : HighJump, new\_ind_6 : SportsTrial,$
$(new\_ind_1,\ hjName_1) : hasSportsName,\ (new\_ind_1,\ date_1) : hasDate,$
$(new\_ind_1,\ city_1) : takePlace,\ (new\_ind_1,\ new\_ind_4) : hasPart,$
$(new\_ind_4,\ new\_ind_5) : hasPart,\ (new\_ind_5,\ perf_1) : hasPerformance,$
$(new\_ind_5,\ new\_ind_2) : hasParticipant,$
$(new\_ind_6,\ new\_ind_3) : hasParticipant,\ (new\_ind_6,\ perf_2) : hasPerformance$
$(new\_ind_2,\ pName_1) : hasPersonName,\ (new\_ind_2,\ country_1) : hasNationality,$
$(new\_ind_3,\ pName_2) : hasPersonName,\ (new\_ind_3,\ country_2) : hasNationality$

Note that in the interpretation Abox the person instance new_ind_2 participates in a high jump trial (new_ind_5) and, therefore, is also an instance of the concept $HighJumper$ (see the Tbox in Figure 5). Thus, information about high-level events, e.g. high jump trials, also influences information that is available about the related parts. With queries for $HighJumper$s the corresponding media objects would not have been found otherwise. Thus, recognizing high-level events is of utmost importance in information retrieval systems (and pure content-based retrieval does not help).

In the previous example, an explanation is found for every assertion in Γ_2. However it might be the case that no explanation can be obtained due to a lack of relevant axioms in the background knowledge (Σ) or due to a lack of relevant observations in Γ_2. An example for the lack of relevant axioms in the background knowledge is given, if Γ_2 contains the assertion $(sName_1,\ rank_1) : sportsNameToRanking$ and there is no rule in Σ containing $sportsNameToRanking(X, Y)$ in the head. The lack of relevant axioms in the background knowledge for interpreting text can be considered as a motivation for the agent to learn. An example for the lack of relevant observations is given, if Γ_2 contains isolated individuals, i.e. individuals that are not in any relation with other instances (see the $rank_1 : Ranking$ assertion in Figure 4). In this case relevant evidence, namely relations between individuals, is missing for interpreting the document.

The example discussed here covers the interpretation of multimedia documents in text modality. However, the same engine implementing the semantic interpretation process is used in BOEMIE to interpret also other modalities. In [7] we have discussed the interpretation of images and presented an evaluation of experimental results obtained on a corpus of athletics images. Observe that given the nature of visual modalities, where visibility restrictions have to be considered, ambiguities often lead to multiple explanations. For example, imagine that the image analysis process extracts a person body, a person face, a horizontal bar, and spatial relations among these objects (e.g., the person body is above the horizontal bar) for the image in Figure 3. The interpretation engine generates two explanations for this image: one interpreting it as showing a high jump trial and another one interpreting it as showing a pole vault trial. This is due to the fact that, different from the detection of a pole, the detection of a horizontal bar is not enough to discriminate between high jump and pole vault trials.

So far, we have presented the modality-specific interpretation of multimedia documents only. However, the semantic interpretation of multimedia documents involves also a second level of interpretation, where modality-specific interpretation results are fused to provide fused interpretation Aboxes. Due to the isolated interpretation of modality-specific analysis Aboxes, in our approach, a fused Abox is not only the union of modality-specific interpretation Aboxes, but also contains information about individuals that depict the same object in different modalities (identification). For example, the multimedia document in Figure 3 is a multi-modal document and its interpretation requires not only the integration of single-modality interpretation results, but also the identification of instances of high-level concepts such as a $Person$. Note that high-level concepts such as $Person$ or $SportsTrial$ are the only candidates for identification because (different from mid-level concepts) they are modality-independent and can appear in different modality-specific interpretation Aboxes. In terms of description logics the fact that two individuals are the same is represented with a so-called *same-as* assertion, e.g., $(new\_ind_4,\ new\_ind_9) : same\text{-}as$.

The fusion approach processes captioned images such as the one in Figure 3 as follows: First, for each image interpretation every high-level concept instance found in the image interpretation Abox is hypothesized to be the same as every high-level concept instance found in the caption/text interpretation Abox (identification). Second, a fused candidate interpretation Abox is generated, which contains the same-as assertions and the union of image and caption interpretation Aboxes. Third, each fused candidate interpretation Abox is checked for consistency w.r.t. the domain ontology. All inconsistent Aboxes are discarded and only consistent ones are provided as fused interpretation Aboxes for the multimedia document.

5 Matching for augmenting multimedia interpretation

Ontology matching is exploited for supporting all the activities of the evolution methodology, from population to coordination. Interpretations of new incoming multimedia resources are continuously produced by using reasoning techniques, leading to the assimilation of new individuals and assertions into the ontology. Moreover, the ontology used to this end needs to be updated as well, by introducing new concepts and roles in the ontology, in order to deal with changes of the domain over time. New information provided by external knowledge sources is part of the evolution process in the methodology introduced in Section 2. The idea is that the knowledge provided by other ontologies, web directories, and, in general, knowledge repositories can be used to find appropriate descriptions of a missing concept when a resource remains unexplained after interpretation. Moreover, the alignment of the domain ontology with the external knowledge sources is maintained over time in order to increase the domain knowledge available for subsequent ontology evolutions. Ontology matching techniques are used both for supporting population and enrichment and for the goal of ontology alignment and coordination. Ontology matching is defined as a process *match* which takes two ontologies O_1 and O_2 as input and returns a set of mappings among the elements of O_1 and O_2, as follows:

$$match(O_1, O_2) \rightarrow \mathcal{M}_{O_1,O_2}$$

The resulting mapping set $\mathcal{M}_{O_1,O_2}$ is a set of 5-tuples of the form:

$$\mathcal{M}_{O_1,O_2} = \{\langle E_1, E_2, \approx^{\mathcal{R}}, \mathcal{V}, \mathcal{S} \mid (E_1 \approx^{\mathcal{R}} E_2), E_1 \in O_1, E_2 \in O_2, \mathcal{V} \in [0,1], \mathcal{S} \in [0,1]\rangle\}$$

where,

- E_1 and E_2 denote two ontology elements (i.e., concepts, roles, individuals);
- $\approx^{\mathcal{R}}$ is a matching relation which denotes that E_1 and E_2 are similar and, possibly, the kind of relation holding between them (e.g., $\equiv$, $\sqsubseteq$);
- $\mathcal{V}$ denotes a confidence value associated with $\approx^{\mathcal{R}}$;
- $\mathcal{S}$ denotes the level of similarity between E_1 and E_2 as a measure in the range [0,1].

The different goals of ontology matching require matching techniques that work at the schema level (Tbox) as well as at the instance level (Abox). In order to provide a highly flexible toolkit for ontology matching, the ontology matching service HMatch 2.0 is conceived as a framework of matching techniques and components that can be used separately or in combination for the different purposes. Each matching component is used to support one or more specific tasks of the ontology evolution process, as summarized in Table 1.

Table 1. Usage of HMatch 2.0 components in BOEMIE

Evolution Activity	**Task**	HMatch 2.0 **component**	**Goal**
Population	Instance grouping	HMatch($\mathcal{I}$)	To group together instances referring to the same individual in the domain (i.e., matching instances)
Enrichment	Concept enhancement	HMatch($\mathcal{L}$)	To suggest names for new concepts and roles
Coordination	Alignment	HMatch($\mathcal{C}$)	To align a new version of the domain ontology with other external knowledge sources
Coordination	Versioning	HMatch($\mathcal{C}$)	To evaluate a measure of the difference between two versions of the domain ontology

In the population activity, the instance grouping task (see Figure 1) is responsible for grouping all the instances that represent the same real object or event. The instance matching component (HMatch($\mathcal{I}$)) is invoked to match every incoming concept instance against a set of other instances that already populate the involved concept. In the enrichment activity, concept enhancement is responsible for improving a concept identified by concept learning, through terminological knowledge acquired from external sources, such as external domain ontologies or taxonomies. In order to acquire relevant knowledge, the linguistic matching component (HMatch($\mathcal{L}$)) is used for concept matching. In the coordination activity, contextual matching (HMatch($\mathcal{C}$)) is used for concept matching for two different evolution tasks (see Figure 1): in ontology alignment, the goal is to establish mappings connecting a new version of the athletics

domain ontology with other external knowledge sources; in ontology versioning, concept matching is used to evaluate the measure of change between two different versions of the domain ontology.

5.1 Matching at the schema level

The goal of schema-level matching is to provide a set of techniques for determining the degree of similarity between two ontology concepts by considering their formal specifications (i.e., Tbox specifications). To this end, schema-level matching is enforced through linguistic matching techniques and a *concept affinity function.*

Linguistic Similarity. The linguistic similarity function is exploited for calculating the linguistic affinity between two atomic ontology elements (i.e., concepts, roles).

Definition 1. *Linguistic similarity function. Given two ontology elements e_1 and e_2, the linguistic similarity function L_S is defined as*

$$L_S(e_1, e_2) = \begin{cases} 1, \textit{ if } sim(n^{e_1}, n^{e_2}) \geq t; \\ 0, \textit{ otherwise.} \end{cases}$$

where $sim \in [0, 1]$ is the linguistic similarity between n^{e_1} and n^{e_2}, that are the labels (i.e., names) of e_1 and e_2, and t is a threshold expressing the minimum level of linguistic similarity required for considering e_1 and e_2 as matching elements.

A whole family of techniques is provided by the $\mathsf{HMatch}(\mathcal{L})$ component to implement $sim(n^{e_1}, n^{e_2})$. In particular, $\mathsf{HMatch}(\mathcal{L})$ provides the following techniques:

- **Syntactic**: linguistic matching is performed by using a string matching technique (i.e., QGram, i_Sub).
- **Semantic**: linguistic matching is performed by using a thesaurus or a lexical system (i.e., WordNet) of terms and terminological relationships and a notion of weighted distance between terms.
- **Mixed**: linguistic matching is performed by using a combination of both syntactic and semantic techniques.

The choice of the most suitable linguistic matching technique to be used for calculating *sim* depends on both the nature and typology of linguistic features of ontology elements and on the expected response time. For instance, syntactic matching techniques are suggested for linguistic matching when poorly informative labels are used for ontology elements (e.g., acronyms, person names). Semantic matching techniques are suitable where structured and informative labels are used, and when terminological relationships (e.g., synonyms, hypernyms, hyponyms) between n^{e_1} and n^{e_2} need to be considered for linguistic matching.

Concept Affinity. Concept affinity is defined to determine the level of semantic affinity between two ontology concepts by considering the number of matching elements belonging to their contexts.

Definition 2. *Concept context. We define the context $Ctx(c)$ of a concept c as a set of adjacent elements differently composed according to three matching models:*

- **Shallow***: $Ctx(c) = \{n^c, P^c\}$, where P^c denotes the role applied to c.*
- **Deep***: $Ctx(c) = \{n^c, P^c, S^c_{\sqsubseteq}, S^c_{\sqsupseteq}\}$, where $S^c_{\sqsubseteq}$ denotes the concepts that subsume c, while $S^c_{\sqsupseteq}$ denotes the concepts subsumed by c.*
- **Intensive***: $Ctx(c) = \{n^c, P^c, S^c_{\sqsubseteq}, S^c_{\sqsupseteq}, R^c\}$, where R^c denotes elements used as role range that are applied to c.*

According to the considered matching model, concept affinity of two concepts c_1 and c_2 is evaluated on the basis of the number of matching elements in $Ctx(c_1)$ and $Ctx(c_2)$. To this end, ontology elements in $Ctx(c_1)$ are compared with the ontology elements in $Ctx(c_2)$ by relying on their linguistic similarity L_S.

Definition 3. *Concept affinity. Given two ontology concepts c_1 and c_2, the concept function $CA \in [0, 1]$ is defined as*

$$CA(c_1, c_2) = \frac{2 \cdot \sum_{k=1}^{k=|Ctx(c_1)|} \sum_{h=1}^{h=|Ctx(c_2)|} N_S(e_k^{c_1}, e_h^{c_2})}{|Ctx(c_1)| + |Ctx(c_2)|}$$

where $e_k^{c_1} \in Ctx(c_1)$ and $e_h^{c_2} \in Ctx(c_2)$ are the ontology elements in the context of c_1 and c_2, respectively.

Concept affinity is evaluated by the HMatch($\mathcal{C}$) component which can be configured to perform matching according to the various matching models. Moreover, a surface matching model is also available that calculates the concept affinity by considering only concept names. The choice of the most suitable matching model highly depends on the level of detail of the concept descriptions as well as on the expected degree of precision and recall of the results. For example, in ontology enrichment, the more information is available for the missing concept, the deeper the matching model can be and the more precise descriptions of external concepts can be retrieved. Otherwise, with poor concept descriptions, only simple external concepts can be retrieved since we can rely on linguistic similarity only. Both HMatch($\mathcal{L}$) and HMatch($\mathcal{C}$) have been extensively evaluated on a general benchmark of real ontology matching cases [6, 7].

5.2 Matching at the instance level

The goal of instance matching is to determine instances that represent the same real object or event in two Aboxes. HMatch($\mathcal{I}$) evaluates the degree of similarity among different instances by considering those assertions which provide a description of the instance features. Since assertions in an Abox are provided according to the terminology defined into a corresponding Tbox, HMatch($\mathcal{I}$) takes into account a set of mappings between the two Tboxes associated with the Aboxes to be matched, in order to compare individuals that are instances of matching concepts and to establish a

correspondence among their roles [4]. Standard reasoning is used in order to determine the individuals that are instances of a concept in the Tbox. Then, the similarity of role filler values as well as the similarity of their direct types is evaluated.[5] When two individuals are compared, their similarity is proportional to the number of similar role fillers they share. This kind of similarity evaluation between individuals is not enough: in fact, we need to map together individuals which denote the same real object in the domain and not just similar individuals (i.e., individuals featured by similar values). To this end, we classify the key capability of roles by weighting different individual roles for similarity evaluation. The idea behind this approach is that some roles are more important than others for univocally identifying the real object denoted by an individual, because they are relevant for object identification. For example, the name of a person is more prominent than his age for person identification.

The approach adopted in $\mathsf{HMatch}(\mathcal{I})$ is based on the idea of considering roles as connections between individuals and propagating similarity values through them. In particular, the $\mathsf{HMatch}(\mathcal{I})$ process is composed by two main functions, namely *Instance affinity* (IA) and *Filler similarity* (F_S).

Instance Affinity. Given two individuals i_1 and i_2 that are instances of the same (or matching) concept, the instance affinity function $IA(i_1, i_2) \rightarrow [0, 1]$ provides a measure of their affinity in the range [0,1]. For each individual i, instance affinity is calculated by taking into account all the roles $p_1^i, \ldots, p_n^i$ featuring i together with their corresponding role fillers $f_1^i, \ldots, f_n^i$. For each Role p_j, a weight $W_{p_j} \in [0, 1]$ is defined expressing the capability of p_i for the goal of univocally identifying the individual i in the domain of interest. Role weights are manually defined for the considered domain by taking into account also the results of the extraction process from a corpus of (manually) annotated multimedia resources.

Definition 4. *Instance affinity. Given two individuals i_1 and i_2, the instance affinity function $IA(i_1, i_2)$ between them is calculated as follows:*

$$IA(i_1, i_2) = \frac{2 \cdot \sum_{k=1}^{k=n} F_S(f_k^{i_1}, f_k^{i_2}) \cdot W_{p_k^{i_1}}}{\sum_{k=1}^{k=n} W_{p_k^{i_1}} + \sum_{k=1}^{k=n} W_{p_k^{i_2}}}$$

For each role filler $f_k^{i_1}$ featuring i_1, we execute the function $F_S(f_k^{i_1}, f_k^{i_2}) \rightarrow \{0, 1\}$ between $f_k^{i_1}$ and the corresponding role filler $f_k^{i_2}$ of i_2, where two fillers correspond if they denote the value of the same (or matching) role. The filler similarity function returns 1 if the two fillers match, and 0 otherwise. The goal of this step is to consider only the matching role fillers of the two individuals. Then, we sum the weights $W_{p_k^{i_1}}$ associated with each role filler. Note that we take into account only the weights of the roles featuring i_1, since the same (or matching) roles are featured with the same

[4] In case of two Aboxes defined with respect to the same Tbox, the correspondences among concepts and roles is trivial.

[5] The direct types of an individual are the most specific concepts from the ontology of which the individual is an instance of.

weight. Finally, we calculate the ratio of the sum of weights associated with matching filler roles and the sum of the weight of all the roles featuring the two individuals. During the comparison of two individuals, there can be situations in which a defined filler for a role is missing, which means that we have no information about that role at all. In this case, we can use two different strategies: pessimistic or optimistic. The former considers the lack of information as an evidence of the difference between two individuals; the latter ignores the missing role value, interpreting it as undefined, so it does not take part in the computation of similarity. In $\mathsf{HMatch}(\mathcal{I})$, the optimistic strategy is adopted because we want to evaluate the knowledge explicitly asserted about an individual. If all the essential roles – the roles which differentiate each individual from the others – are valued, then the absence of the other roles has not influence on the similarity evaluation. Role fillers can be datatypes (e.g., strings, numbers, dates) or other individuals. In order to adopt the matching technique more suitable for a given pair of fillers, we define the specific function filler similarity.

Filler Similarity. The filler similarity function $F_S(f_1, f_2) \rightarrow \{0, 1\}$ previously introduced is defined as follows:

Definition 5. *Filler similarity. Given two role fillers f_1 and f_2 and a threshold t, the filler similarity function $F_S(f_1, f_2) \rightarrow \{0, 1\}$ is defined as:*

$$F_S(f_1, f_2) = \begin{cases} 1, \textit{ if } sim(f_1, f_2) \geq t; \\ 0, \textit{ otherwise} \end{cases}$$

$Sim(f_1, f_2)$ is a value in the range [0,1] and is calculated in different ways depending on the type of the role fillers f_1 and f_2, according to the following rules:

- **Rule 1:** if f_1 and f_2 are both datatypes, a suitable matcher is executed which evaluates the similarity between datatype values according to the semantic meaning of the roles and to their datatype category. A detailed description of these datatype matchers is provided in [7].
- **Rule 2:** if f_1 and f_2 are both individuals, we check if they are featured by role assertions. If not, $sim(f_1, f_2)$ returns 1 if f_1 and f_2 are instances of the same (or matching) concepts, and 0 otherwise. If f_1 and f_2 are featured by role assertions, $sim(f_1, f_2) = IA(f_1, f_2)$, thus leading to a recursive step, which is iterated until all the datatype role fillers have been compared.
- **Rule 3:** if f_1 is a dataype and f_2 is an individual (or vice versa), we check is f_2 is featured by a role filler f_k matching with f_1. In this case, we apply Rule 1 in order to obtain $sim(f_1, f_2) = sim(f_1, f_k)$. Otherwise, $sim(f_1, f_2) = 0$.

Example. Consider two individuals *PersonName_7368* and *PersonName_4352* that instantiate the concept *PersonName* and their roles as shown in Figure 7.

The two individuals denote (different) person names referring to different athletes. The two considered individuals share a high number of common characteristics. In fact, the two athletes come from the same country (i.e., Poland), have the same gender (i.e., male), and are associated with the same performance (i.e., 2.36). So, the instance

(PersonName_7368, 'Michal BIENIEK') : hasPersonNameValue
(PersonName_7368, Country_3415) : personNameToCountry
(PersonName_7368, Performance_4389) : personNameToPerformance
(PersonName_7368, Male_640) : personNameToGender
(Country_3415, 'POL') : hasCountryNameValue
(Performance_4389, 2.36) : hasPerformanceValue

(a) Abox of individual PersonName_7368

(PersonName_4352, 'Artur PARTYKA') : hasPersonNameValue
(PersonName_4352, Country_5567) : personNameToCountry
(PersonName_4352, Performance_6732) : personNameToPerformance
(PersonName_4352, Male_640) : personNameToGender
(Country_5567, 'POL') : hasCountryNameValue
(Performance_6732, 2.36) : hasPerformanceValue

(b) Abox of individual PersonName_4352

Fig. 7. Two individuals that instantiate the concept *PersonName* in the athletics ontology

matching should be able to recognize the athlete similarity while capturing the identity diversity of the two athletes at the same time.

Since the similarity between *PersonName_7368* and *PersonName_4352* depends on the similarity among their role fillers, the evaluation of such a similarity is based on the results obtained by applying the filler similarity function, that are:

$$F_S(Country\_3415, Country\_5567) = 1$$

$$F_S(Performance\_4389, Performance\_6732) = 1$$

$$F_S(Male\_640, Male\_640) = 1$$

$$F_S(`MichalBIENIEK', `ArturPARTYKA') = 0$$

In order to deal with the fact that the two individuals are referred to different persons in spite of the fact that they share a high number of features, we need to capture the intuition that some roles, such as the name, are more important than others, such as the performance, for the sake of object identification. In the example, we rely on role identification weights defined for the athletics domain, i.e., $W_N = 1.0$ for *hasPersonNameValue*, $W_G = 0.3$ for *personNameToGender*, $W_C = 0.3$ for *personNameToCountryName*, and $W_P = 0.0$ for *personNameToPerformance*, respectively. These weights state that the name is considered to be the most important attribute for the identification of a person. Gender is relevant, since it does not change in time, but it is not a key of a person record because there are many person with the same gender. The same can be said about the country. Finally, the performance is not relevant for the identification of a person because it is not an attribute of the person but a result obtained by the person in some kind of sport event. On the basis of these weights, we apply the instance affinity function described above, with the following results:

$$IA(P_{7368}, P_{4352}) = \frac{W_G + W_C + W_P}{W_P + W_G + W_C + W_P} = \frac{0.3 + 0.3 + 0.0}{1.0 + 0.3 + 0.3 + 0.0} = 0.375$$

The example shows how, using the instance affinity function and the role weights, is capable of providing both a measure of identification among the two individuals. We note that, if we apply the same weight to all the roles, the instance affinity function can also be used to evaluate a general degree of similarity between two individuals rather than the fact that they denote the same real object in the domain. In the instance grouping task of the population activity, the aim is to group together individuals which represent the same real object. Thus, specific weights for the roles in the athletics domain ontology are used.

6 Ontology population and enrichment

Ontology evolution is driven by the results of the semantic interpretation: multimedia resources that are fully explained evolve the ontology through population, while unexplained resources signal the need for ontology extensions through enrichment. Ontology population is the process of inserting concept instances and role assertions into an existing ontology, while ontology enrichment is the activity of extending an ontology, through the addition of new concepts, roles and rules.

6.1 Ontology population

The proposed approach for ontology population is built upon two axes: entity disambiguation and consistency maintenance. With entity disambiguation we refer to the process of identifying instances that refer to the same real object or event. If an ontology is populated with an instance without checking if the real object or event represented by the instance already exists in the ontology (as an instance that has populated the ontology at an earlier population step), then redundant information (in the best case) will be inserted into the ontology. A worst case scenario is the existance of redundant instances containing contradicting information, which may lead to an inconsistent ontology. At the same time maintaining the consistency of an ontology is crucial (mainly through the elimination of contradictory information), as an inconsistent ontology cannot be used for reasoning.

Ontology population can be decomposed into the following tasks:

- *Instance matching*: The first population task is the identification of similar instances contained in the ontology. Assuming an Abox containing the explanation of a multimedia document, a similarity matrix is constructed: this similarity matrix contains a similarity measure of each HLC instance (HLCi) with any other HLCi (of the same HLC) found in the ontology.
- *Instance grouping*: This task is responsible for grouping all the instances that represent the same real object or event, by exploiting the results of the instance matching task, where every incoming HLCi has been measured with respect to similarity with the set of other instances that populate this HLC. Instance grouping employs clustering techniques operating on the similarity matrix returned by the instance matching task to decide which of these instances will be grouped together to form a group that represents the same real object or event.

– *Abox refinement (evolution pattern P2 only)*: In case of multiple explanations the most suitable explanation is selected by exploiting the results of the instance matching/grouping tasks. Assertions related to the rest of the explanations are removed from the Abox, thus leading to a refined version of it.
– *Abox validation*: This task performs consistency checking, to detect possible inconsistencies due to the additions that will be performed to the ontology.
– *Abox assimilation*: The final task is responsible for performing the needed changes in the ontology (by creating all instances/roles in all ontological modules), in order to incorporate the information in the new Abox into the ontology.

Instance grouping Instance grouping is responsible for identifying which instances found in both the ontology and the Abox produced from the analysis of a multimedia resource refer to the same real object or event. This identification is performed under the assumption that instances that are similar represent the same real object or event. Exploiting the degree of similarity among all instances provided by instance matching, instance grouping must decide which instances are similar enough to represent the same real object or event. This decision can be taken by either placing some arbitrary threshold or by performing clustering on the similarity matrix calculated by instance matching for all instances in both the ontology and the Abox explaining a multimedia resource.

Table 2. Example of an instance similarity matrix

	Athlete1	Athlete2	Athlete3	...	AthleteN
Athlete1	-				
Athlete2	0.9	-			
Athlete3	0.9	0.9	-		
...				-	
AthleteN	0.6	0.0	0.9		-

– Thresholding: in this approach, a threshold is defined and instances whose degree of similarity is above the threshold are grouped together. There are two alternative procedures in forming groups by satisfying a threshold, depending whether *complete linkage* or *single linkage* is used. In complete linkage, all degrees of similarity among all instances of a group must be equal to or above the threshold. In single linkage it suffices for an instance to have a degree of similarity that satisfies the threshold with any single instance of the group, in order to become a member of the group. For example, assuming the similarity matrix of Table 2 and a threshold of 0.9, the group formed through complete linkage is (Athlete1, Athlete2, Athlete3). AthleteN cannot be a member of the group, as the similarity degree with Athlete1, Athlete2 is lower than the threshold. On the other hand, the group formed with single linkage (Athlete1, Athlete2, Athlete3, AthleteN) includes AthleteN, as AthleteN has a degree of similarity with Athlete3 that satisfies the threshold.

– Clustering: a slightly more complex approach is to employ a clustering algorithm in order to group instances. Clustering is a form of machine learning and as such two questions arise when it is used: which clustering algorithm should be used, and how the data to be clustered can be represented more effectively in order to obtain the desired results. Regarding data representation, since the degree of similarity between two instances is already available, a clustering algorithm that exploits the distance among instances such as k-means [8] can be easily applied. Given a fixed number k of (desired or hypothesized) clusters, the k-means algorithm assigns observations (instances) to those clusters so that the means across clusters are as different from each other as possible. In addition, k-means can be easily combined with n-fold cross validation in order to estimate the "optimal" number of clusters from the data. An interesting property of the k-means algorithm is that the centre of each cluster is calculated during the formation of the cluster. The instance that is closest to this centre is the instance that conveys the most information about the real object or event which the group (cluster) represents, and thus can constitute the basis over which the consistency of all other instances of the group can be tested.

Example Population is performed when there is enough information in the ontology to fully explain a multimedia resource. For example, the text document presented in Figure 3 can be fully analysed (with a single explanation) if the ontology of Figure 5 and the rules of Figure 6 are assumed. In such a case, the evolution pattern P1 will be triggered and all instances in the Abox that contains the results of the semantic interpretation will be matched against all individuals contained in the ontology, and the degree of similarity among all instances will be stored in a similarity matrix. The similarities stored in the matrix will be examined by the instance grouping task, in order to find the instances that represent a single real object or event. Since Abox refinement is not needed (as the Abox contains a single explanation), ontology population proceeds with validating what will be added to the ontology. If no contradicting information is detected, the Abox is assimilated into the ontology. During assimilation, instances from the ontology that are identical (i.e. have a degree of similarity equal to 1.0) to instances from the multimedia resource Abox can be reused, in order to avoid the addition of instances that are duplicates into the ontology.

However, in case the semantic interpretation is unable to explain a multimedia resource with a single explanation and produces two or more competing explanations, disambiguation must be performed in order to select the most prominent explanation. Assume for example that the text document of Figure 3 was explained (erroneously) also with the following assertions (in addition to those of section 4.2), denoting also a pole vault in addition to high jump sport:

$new\_ind_7 : PoleVaultCompetition$,
$new\_ind_8 : PoleVaultRound, new\_ind_9 : PoleVault$,
$(new\_ind_7,\ hjName_1) : hasSportsName, (new\_ind_7,\ date_1) : hasDate$,
$(new\_ind_7,\ city_1) : takePlace, (new\_ind_7,\ new\_ind_8) : hasPart$,
$(new\_ind_8,\ new\_ind_9) : hasPart, (new\_ind_8,\ perf_1) : hasPerformance$,
$(new\_ind_9,\ new\_ind_3) : hasParticipant, (new\_ind_9,\ perf_2) : hasPerformance$

An Abox with more than one explanations (i.e. instances of both HighJumpCompetition and PoleVaultCompetition) will trigger evolution pattern P2. Again instance matching will be performed, which will produce an updated similarity matrix. Then, instance grouping will be performed on the contents of the similarity matrix in order to obtain groups. However, a single explanation must be selected from the available ones. Explanation disambiguation is performed as follows: the explanation with the greatest degree of similarity with any instance in the ontology is selected by examining the similarity matrix. In most cases this will result in a single instance, which is the selected explanation. In case however more than one instances that are parts of competing explanations share the same (maximum) degree of similarity, disambiguation will be performed through exploitation of the grouping information, selecting the instance that it is closer to the centre of its group.

Once a single explanation has been selected, all instances not related with the selected explanation will be discarded from the Abox of the multimedia resource. Finally, during Abox validation the resulting Abox will be checked for consistency and if it is found to be consistent, it will be assimilated into the ontology.

6.2 Ontology enrichment

Ontology enrichment is performed every time the background knowledge is not sufficient to explain the extracted information from the processed multimedia documents. Thus, the ontology enrichment activity is expected to extend this background knowledge through the addition of new concepts/roles/rules, in order to better explain extracted information in the future.

The ontology enrichment activity is triggered by either the P3 or the P4 evolution patterns. The evolution pattern P3 is selected when no explanation (i.e., an HLCi) has be found for a given Abox, and can lead to the insertion of a new HLC, a new role or a new rule into the ontology, or in the accumulation of the Abox in a "waiting" queue if available evidence cannot justify the addition of a new concept/role/rule. On the other hand, evolution pattern P4 is selected when the background knowledge is not sufficient to even assign MLCs to all of the extracted elements of a multimedia resource, thus inserting instances of the "unknown" MLC in the Abox. In this case, pattern P4 can result in the addition of a new MLC in the ontology. In fact, the detection of new MLCs is considered to have priority over the identification of new HLCs, because knowledge about a new MLC can lead to different semantic interpretation results about the same resources. As a result, when instances of the "unknown" MLC are found in Aboxes that contain no explanations, evolution pattern P4 is selected instead of P3.

Ontology enrichment is decomposed into the following tasks:

- Concept learning: The goal of this task is to propose new concepts (either HLCs or MLCs) and roles by exploiting similarities found through clustering, either in unexplained documents (evolution pattern P3) or in unknown objects recognized by the information extraction engine (evolution pattern P4). It can be decomposed into two main sub-tasks, clustering and concept formation.
  - Clustering: The main objective of the clustering task is to provide evidence that can support the creation of new concepts or roles.

    - Concept formation: This task is applicable only if a new HLC has been proposed by clustering. Exploiting the results of clustering, concept formation examines the clustered elements in order to extract common information (such as concepts/roles) and use this common information to form a new concept, as a result of this task.
- Concept enhancement (evolution pattern P3 only): This task is responsible for improving a concept identified by concept learning, through knowledge acquired from external sources, such as external domain ontologies or taxonomies.
- Concept definition: This task receives the new concept (either a new MLC or HLC) or role as defined through the previous tasks, and shows the concept/role definition to the ontology expert. The ontology expert must approve the new concept/role in order to be assimilated into the ontology and additionally can revise the definition of the new concept/role.
- Concept validation: This task performs consistency checking, by trying to detect possible inconsistencies due to the addition of the new concept/ role to the ontology.
- Concept assimilation: The last task of ontology enrichment is responsible for performing the required changes in the ontology in order to incorporate the newly formed concept/role into the ontology.

Example Let as assume an Abox where the semantic interpretation activity was unable to find an explanation. In addition, instances of the "unknown" MLC have been extracted by the information extraction toolkit. When such an Abox is processed by the evolution pattern selector, pattern P4 will be selected. In such a case, the Abox will be placed in a "waiting" stage. Once a significant number of instances of the "unknown" MLC have been assimilated, then clustering will be performed, during concept formation. If enough (modality-specific) information is available that can lead to the formation of clusters, the concept definition task is responsible to present the results of the clustering to the ontology expert. The ontology expert must decide if the presented information is enough to justify the addition of a new MLC. In such a case, the expert must define the new concept(s), and associate all presented instances represented by the new concept with it.

In case of an Abox that has no explanation and also no instances of the "unknown" MLC, evolution pattern P3 will be selected. In such a case, the Abox will be also placed in a "waiting" stage, until enough Aboxes have been gathered. The Aboxes gathered are clustered in order to obtain clusters of Aboxes that are similar, and all Aboxes in a cluster can possibly be explained by a single concept, which is not contained into the current version of the ontology. Once clusters have been identified, a new concept is formed for each found cluster by using all common information among all Aboxes of the cluster. Each newly formed concept will be further enhanced during concept enhancement, by trying to locate the formed concept in external knowledge sources through coordination and exploiting information from these knowledge sources, like concept/role names and properties. Once new concepts/roles have been formed and possibly enhanced, they must be approved and reviewed by the ontology expert during the concept definition task: the expert must decide which of these proposed

concepts/roles will be kept, what their definition will be and which Aboxes can be associated with them. Each concept/role definition must be checked for consistency and assimilated into the ontology, if no inconsistencies have been found.

7 Experimentation and evaluation

In this section, we present the evaluation of the most relevant components that contribute to the ontology evolution methodology. For this purpose we will base the evaluation of each component on the comparison of expected and effective results by means of precision, recall and F-measure. While the parameters for the evaluation differ from component to component, a general definition of precision, recall and F-measure will be given. For a more specific definition refer to the subsection of each component.

Definition 6. ***Precision and Recall.*** *Given a ground truth of expected results E and the retrieved results R, precision P is defined as*

$$P = \frac{\mid E \cap R \mid}{\mid R \mid}$$

and recall R as

$$R = \frac{\mid E \cap R \mid}{\mid E \mid}$$

Definition 7. ***F-Measure.*** *In order to combine precision and recall into a unique value, giving more emphasis to the balance of the two measures, F-measure F is introduced as the harmonic mean of precision P and recall R [9]:*

$$F = \frac{2 \cdot P \cdot R}{P + R}$$

7.1 Evaluation of proposed multimedia interpretation techniques

The utility of the abduction-based multimedia interpretation framework is analyzed through an empirical evaluation of its results over a collection of web pages. The evaluation will focus on the results of text interpretation. For an evaluation of image interpretation refer to [10]. The core component of this implementation is the DL-reasoner RacerPro in version 1-9-2 [11] that supports various inference services. The abductive retrieval inference service, which is the key inference service for multimedia interpretation, is integrated into the DL-reasoner.

Experimental setting and criteria To test the approach, an ontology about the athletics domain was used as well as two corpora of 104 web pages each, containing daily news about athletics events. The first corpora is a gold standard containing manually annotated metadata, while the second corpora is system annotated. Analysis algorithms that implement shallow text processing and machine learning techniques were trained in order to obtain concept instances as well as relations between the

instances (see Figure 4). The training process was performed with the help of an annotation tool applied to the first corpus of web pages. The annotation process is two-fold: in the first step annotators manually associate words in the text with corresponding mid-level concepts in the ontology. For this purpose, the following concepts have been used, i.e. *PersonName*, *Country*, *City*, *Age*, *Gender*, *Performance*, *Ranking*, *SportsName*, *RoundName*, *Date* and *EventName*. Second, the segments annotated with mid-level concepts are grouped and each group is associated with a high-level concept such as *Athlete*, *SportTrial*, *SportsRound*, *SportEvents* and *SportsCompetition*. The resulting annotations are considered as ground truth for our evaluation. After finalizing the training process, the second corpus has been automatically analyzed and interpreted to detect concept instances and relations between them. The result of automatic analysis and interpretation over the second corpus will be evaluated in this section.

To set up the evaluation, a set of queries has been defined in order to ask for the number of HLCi (high-level concept instances) in both the gold standard corpus and in the system-annotated corpus. In this way, names of high-level concepts constitute the parameters to evaluate the precision and recall of the multimedia interpretation framework. Accordingly, E and R of definition 6 are replaced by E_{HLC} and R_{HLC} respectively, to obtain a ratio of correctly retrieve instances of HLCs ($E_{HLC} \cap R_{HLC}$) over the total number of retrieved instances of HLCs (R_{HLC}) and the total number of expected instance of HLCs (E_{HLC}).

Evaluation results The evaluation results (see Table 3) shows that expected high-level concepts (explanations) were extracted in most of the cases. There are exceptional cases in which no explanations were extracted due to the lack of necessary rules or lack of low-level text analysis results. For example, in the case of *SportsRound*, the interpretation framework expects input about $SportsRoundName$ and *Date* instances in a relation $sportsRoundNameToStartDate$, but this structures are rarely found during text analysis. Therefore, the rule necessary to create instances of $SporstRound$ is never applied.

Table 3. Evaluation of the interpretation framework about the text modality

HLC	E	R	$R \cap E$	**Precision**	**Recall**	**F-Measure**
Athlete	783	591	496	0.84	0.63	0.72
SportsTrial	729	641	513	0.80	0.70	0.74
SportsCompetition	443	200	188	0.94	0.43	0.80
SportsEvent	304	304	266	0.87	0.87	0.87
SportsRound	375	0	0	0	0	0

Furthermore in Table 4 it can be observed that from a total of 200 extracted $SportCompetition$s, five of them have been specilized to $HighJumpCompetition$s and ten to $PoleVaultCompetition$s. This is due to low-level analysis results where

Table 4. Discovery of more specific high-level concepts

SportsCompetition	HighJumpCompetition	PoleVaultCompetition
200	5	10

Table 5. Information gain from text-image fusion results

SportsTrials	HighJump	PoleVault
641	85	46

instances of the concepts $HighJumpName$ and $PoleVaultName$ were found. As observed in Figure 5 the definition of a $HighJumpCompetition$ ($PoleVaultCompetiton$) requires a $HighJumpName$ (respectively $PoleVaultName$) as a domain for the role $hasSportsName$.

From Table 5 it can be observed that from a total of 641 $SportTrial$s automatically extracted from text, some of them are also specialized into $HighJump$ or $PoleVault$, which is a result of fusion. In this case an instance of $SportsTrial$ in the text becomes same as an instance of $PoleVault$ or $HighJump$ in an image. Currently the fusion process asumes that whenever there is a person name in a caption, it should correspond to the person in the corresponding image. Therefore, after fusing an image with its caption, it is possible to further fuse the image with the text based on the syntactical similarity of the person's name.

7.2 Matching techniques for ontology evolution

Experimental setting and criteria The methodological approach commonly used for the evaluation of semantic matchmaking tools is based on the idea of building a benchmark constituted by several heterogeneous ontologies to be matched and a set of manually defined results, that is a set of expected mappings (E_M). Then, the matching tool to be evaluated is executed against the ontologies in the benchmark, in order to obtain a set of automatically retrieved mappings (R_M). On the basis of E_M and R_M, the precision and recall metrics are used for the evaluation of the tool:

$$P = \frac{\mid E_M \cap R_M \mid}{\mid R_M \mid}.$$

$$R = \frac{\mid E_M \cap R_M \mid}{\mid E_M \mid}.$$

The evaluation of HMatch 2.0 has been performed over the 2006 and 2007 benchmarks of the Ontology Alignment Evaluation Initiative (OAEI), an international ontology matching contest held with the goal of comparing different ontology matching tools [12] [6].

[6] http://oaei.ontologymatching.org/2007/

Concerning the evaluation of instance matching, we have created a specific benchmark by taking into account 26 Aboxes extracted by textual modality from textual resources about high jump events. The benchmark involves 15841 instances, both from mid-level concepts and from high-level concepts. Among these instances, we have focused on instances from the mid-level concept *PersonName* and from the high-level concept *Athlete*, and we have manually defined a set of 388 mappings, which represent the instance correspondences that are expected. By using this set of expected mappings, we then evaluate instance matching by calculating precision and recall, as in the case of concept matching.

Evaluation results In order to evaluate the concept matching components of HMatch 2.0, we performed the evaluation of each component separately as well as the evaluation of the results obtained by combining linguistic and contextual matching. A detailed description of these results is given in [7, 12]. As a general remark, we note that the intersection of the results obtained by using the two components separately is useful to increase precision (up to a very high level of 0.99), while union is useful to increase recall. This is because the general behavior of HMatch 2.0 is to find a quite small, but very correct number of results. Then, if we apply intersection, we reduce the number of results even further, but we increase the probability to have them correct. On the opposite, if we take the union, we increase the number of results by affecting precision. In more detail, the results show that the union provides the best balance between precision and recall. The general conclusion is that intersection is supposed to be used when the precision of the results is much more important than the number of results retrieved. In all the other cases, union is the best solution in order to combine different components of HMatch 2.0.

Concerning instance matching, we started from the small benchmark discussed in the previous sub section, where we were interested in finding athletes records. Then, we have set an identification weight of 1.0 for person names and of 0.3 for nationalities and ages. Next, we have selected a group of athletes, namely 'Michal BIENIEK', 'Fabricio ROMERO', 'Ebba JUNGMARK', and 'Germaine MASON'. For these athletes, the benchmark contains the number of instances shown in Table 6. It is easy to see that,

Table 6. Number of expected instances per person in the benchmark

Concept	BIENIEK	ROMERO	JUNGMARK	MASON
PersonName	7	3	3	22
Athlete	7	3	2	15

for each concept, given the number n of instances expected for each athlete, the number of m of mappings expected for each athlete is $m = \frac{n(n-1)}{2}$, leading to Table 7. Thus, for the whole benchmark the number of expected mappings is 388. We have executed HMatch($\mathcal{I}$) against the benchmark and, then, we have produced the transitive closure

Table 7. Number of expected mappings per person in the benchmark

Concept	BIENIEK	ROMERO	JUNGMARK	MASON
PersonName	21	3	3	231
Athlete	21	3	1	105

of the set of mappings retrieved by HMatch($\mathcal{I}$), in order to capture also the mappings produced by the transitive interpretation of similarity. The results are shown in Table 8.

Table 8. Results of instance matching evaluation

	HMatch($\mathcal{I}$)	HMatch($\mathcal{I}$) **with transitive closure**
E	388	388
R	165	226
$R \cap E$	159	202
Precision	0.96	0.89
Recall	0.41	0.58

Legenda: E = expected mappings; R = retrieved mappings

Looking at the results, we can conclude that HMatch($\mathcal{I}$) is featured by a very high precision and a low recall. This means that the results retrieved by HMatch($\mathcal{I}$) are highly reliable, but we do not capture all the expected results. Thus, the current version of HMatch($\mathcal{I}$) can be really useful for suggesting similar instances by taking into account a specific instance of interest, since the results obtained are precise. For the next version of HMatch($\mathcal{I}$) we will work on the goal of increasing the recall. The transitive closure shows that a combination of the results obtained from a single execution of HMatch($\mathcal{I}$) can be useful to this goal. Another promising solution is based on the idea of modifying the parameters and providing an automatic mechanism for detecting identification weights for roles.

7.3 Population

Experimental setting and criteria In this section the performance of the population activity will be evaluated, through evaluation of the entity disambiguation task, with the help of a manually annotated corpus. Entity disambiguation is the result of the combination of two tasks (instance matching and instance grouping), whose cumulative performance will be evaluated in terms of precision and recall. The same 26 Aboxes used for evaluating matching techniques (Section 7.2) will also be used for evaluating entity disambiguation. This corpus involves 15841 instances, both from mid-level and high-level concepts. This evaluation will focus only on instances of the

Athlete high-level concept, by measuring which instances are identified as representing the same athlete.

Evaluation results The experiment involves the application of HMatch($\mathcal{I}$) on all 26 Aboxes, in order to calculate a degree of similarity between a pair of instances, for all possible pairs that can be constructed from the 15841 instances in the Aboxes. From these instances, only instances of the concept *Athlete* were considered. Assuming that *Athlete* instances exhibiting degrees of similarity with other *Athlete* instances below 0.5 are too dissimilar to represent the same real athlete, they were also discarded. The remaining 130 *Athlete* instances (referring to 38 real athletes) with degree of similarity $>= 0.5$ were clustered, as part of the instance grouping task. K-means was utilised along with 10 fold-cross validation, in order to estimate the number of clusters prior to assigning each instance into a cluster.

The k-means clustering algorithm accepts the (maximum) number of clusters to be formed (k) and tries to distribute instances into clusters in such a way that the centres of clusters are as different from each other as possible. In order to estimate the number k of clusters, the following procedure is used: starting with $k = 1$, 10 fold-cross validation is performed on the data to be clustered. During 10 fold-cross validation, the data to be clustered is split into 10 parts. In each fold, 9 parts of the data are used for creating a clusterer, while the unused tenth part is used to measure the plausibility of the clusterer. As plausibility of a clusterer with respect to a set of instances, the mean distance of all instances from its cluster centres was used. The same procedure is repeated 10 times, so as each part has been used exactly once for measuring the clusterer plausibility. The final plausibility for this k is the mean of plausibilities of all folds. The k estimation continues by increasing k by one, and repeating 10 fold-cross validation, in order to estimate the plausibility for this new k. If the new plausibility is better than the old one, the new k is retained and the algorithm proceeds with $k + 1$. If it is not, the algorithm stops, returning $k - 1$ as the possible number of clusters. As distance between two instances A and B, $1 - degree\_of\_similarity(A, B)$ was used.

Concerning clustering, the expected number of clusters was 38, since the 130 *Athlete* instances represented 38 real athletes. The clusters obtained were 34, as shown also in Table 9. Two evaluation approaches have been followed in order to evaluate (a) the performance of correctly recognising all instances particular to a real athlete, and (b) the uniformity of calculated clusters, i.e. whether clusters are formed from instances representing two or more real athletes. The first approach (upper half of Table 9) evaluates the number of real athletes that were correctly identified, i.e. all instances corresponding to a real athlete were grouped into the same cluster, and this cluster did not contain any instances corresponding to a different real athlete. Performance figures exhibit 0.71 precision and 0.63 recall, as instances corresponding to four real athletes were split into more than one clusters (although these clusters remained uniform) and instances corresponding to ten real athletes were grouped together, sharing the common property of having only two instances per real athlete. Looking at the results obtained about the instances of the four athletes that were split into (mostly) two clusters, one can conclude that these instances belong to athletes represented by many instances (ten or more) and separate clusters were formed by the clustering algorithm

due to variations in the name of the athlete. For example, in the corpus were ten instances representing the athlete Stefan Holm, six of which were referring to the athlete as "Holm Stefan" and four of them were using the name "Stefan Holm". Despite the fact that instance matching gave a high degree of similarity among all these instances, the degree of similarity was a little lower between instances that used different way of representing the athlete's name, than the similarity between instances that used exactly the same way of writing the athlete's name, which guided the clustering algorithm to separate them into distinct clusters.

Table 9. Results of entity disambiguation

Athletes recognised					
Expected Clusters	Returned Clusters	Correct Clusters	Precision	Recall	F-Measure
38	34	24	0.71	0.63	0.67
Cluster Uniformity					
Expected Clusters	Returned Clusters	Correct Clusters	Precision	Recall	F-Measure
38	34	33	0.97	0.85	0.90

The lower part of Table 9 presents the performance figures if the coherence or uniformity of formed clusters is examined, i.e. if evaluation concentrates only on whether each cluster contains instances representing a single athlete, even if the cluster does not contain all instances representing a real athlete. These results include the instances representing the four athletes that were split into more than one cluster, with these instances being distributed into nine clusters (instances from three athletes were split into two clusters, while instances from a single athlete were split into three clusters due to a single instance written using only capital letters). The only cluster that is not uniform is the one that contains the instances from the ten athletes that were grouped together. Looking at the results as a whole, most of the failures observed can be attributed to different degrees of similarities assigned by instance matching due to variations occurring in how names are expressed, which suggests that results may improve if better string matching techniques are added to $\mathsf{HMatch}(\mathcal{I})$.

8 Related work and original contribution

8.1 Ontology evolution

The recent success of distributed and dynamic infrastructures for knowledge sharing has raised the need for semiautomatic/automatic ontology evolution strategies. An overview of some proposed approaches in this direction is presented in [13], even if limited concrete results have appeared in the literature. In most recent work, formal and logic-based approaches to ontology evolution are also being proposed. In [14], the authors provide a formal model for handling the semantics of the change phase embedded in the evolution process of an OWL ontology. The proposed formalization allows

to define and to preserve arbitrary consistency conditions (i.e., structural, logical, and user-defined consistency conditions).

A six-phase evolution methodology has been implemented within the KAON [15] infrastructure for business-oriented ontology management. The ontology evolution process starts with the capturing phase, that identifies the ontology modifications to apply either from the explicit business requirements or from the results of a change discovery activity. In the representation phase, the identified changes are described in a suitable format according to the specification language of the ontology to modify (e.g., OWL). The effects of the changes are evaluated in the semantics of change phase, where the ontology consistency check is also performed. Due to the fact that an ontology can reuse or extend other ontologies (e.g., through inclusion or mapping), the propagation phase ensures that any ontology change is propagated to the possible dependent artifacts in order to preserve the overall consistency. The subsequent implementation phase has the role to log all the performed changes in order to support the recovery facilities which are provided to reverse an ontology change in the final validation phase in case that an undesired effect had occured.

The ontology evolution approach proposed here puts significant effort in maintaining the consistency of the ontology while, at the same time, trying to identify and eliminate redundant information. Initial attempts towards redudancy elimination have been presented in the Artequakt [1] and SOBA [2] systems. Artequakt follows a heuristic approach, by applying two manually written heuristics in order to identify and merge instances that refer to the same real object or event, which are applied after all instances have populated the ontology. SOBA, on the other hand, performs simple checks during instances creation (i.e. before the instances populate the ontology), in order to re-use instances that refer to the same real object or event instead of creating new ones. BOEMIE tries to enhance the Artequakt proposal through the use of matching techniques instead of manually developed heuristics.

With respect to the state of the art in the literature on ontology evolution, original contributions of the approach presented here can be seen at two different levels, the whole methodology and the specific activities. The methodology as a whole proposes a new conceptualization of the problem of evolving multimedia ontologies, by presenting a pattern-driven evolution approach, where the most prominent evolution pattern for a specific evolution scenario is automatically identified on the basis of the results of the semantic interpretation activity against the background knowledge. Moreover, the methodology aims to minimize the human involvement by providing a set of learning, matching, and reasoning techniques that offer support in the various evolution activities, to allow the ontology expert to refine proposed working knowledge and/or to validate/choose among proposed alternative choices. Concerning novel contributions at the level of specific activities, the methodology for ontology population uses an innovative approach for the detection of instances which refer to the same real object or event, based on instance matching and non-standard clustering techniques. For ontology enrichment, clustering is used for detecting enough information to support the introduction of a new concept/relation and this supporting information is enhanced though information retrieved from external knowledge sources. Thus, the involvement of the ontology expert is reduced, as the expert is required to revise an al-

ready formed concept/relation rather than having to define this new concept/relation from scratch. Matching techniques are used in combination with reasoning and clustering techniques, thus leading to the development of a more flexible and comprehensive approach to concept/instance matching for evolution, by enforcing both structural matching and semantic matching. With respect to the state of the art in the field of knowledge representation and reasoning, the novel contribution of the evolution methodology regards is the formalization of high-level multimedia interpretation as a logical decision problem and its implementation as a non-standard inference service, namely abduction.

8.2 Reasoning for multimedia interpretation

First techniques for finding abductive explanations have been discussed long ago [16, 17]. In [18] Mayer and Pirri present a semantic tableaux for abduction in the context of first-order logic using 'reversed skolomization'. However, the first-order methods are very loose in discarding explanations that are not minimal. But the minimality in first-order abduction is the main issue. Therefore, a relaxation of the minimality requirement has been suggested, but until now only the formalization of this problem is shown.

In [19], Shanahan presents a formal theory of robot perception as a form of abduction. In this work, low-level sensor data is transformed into a symbolic representation of the world in first-order logic and abduction is used to derive explanations. In the context of scene interpretation, recently, Neumann and Möller proposed the use of DLs for the representation of aggregates that can be used by reasoning services as building blocks for the scene interpretation process [20].

In [21] a thorough discussion of abductive reasoning tasks in DLs including Abox abduction is presented. In this work, the authors consider the development of algorithmic techniques based on semantic tableaux for employing abductive inference in expressive DLs as the most promising approach. However, for the time being, the question how abductive inference in DLs can be adopted for finding 'preferred' interpretations of multimedia documents remains unanswered. In particular, a methodology for finding and selecting explanations (interpretations) is missing.

The abduction approach presented here is based on the combination of the works in [22], [19] and [20]. In contrast to approaches such as [23], which use abduction in the context of rules in logic programming only, this approach combines existing DL reasoning mechanisms and rules in a coherent framework, i.e., abduction is considered as a new type of non-standard retrieval inference service, which is integrated into an existing DL reasoner.

8.3 Matching for multimedia interpretation

The ontology matching techniques used rely both on concept matching and on instance matching. Matching techniques at the concept/schema level have been studied in the field of ontology matching. For a complete survey on concept matching, the reader can refer to [24]. In case of instance matching the challenging issue is not only in the field of ontologies, but also in the field of data integration: the same problem is also

referred to as *Record Linkage* in statistics research community, it is known as *Entity resolution*, *Entity identification* or *Object fusion* in database research community and is addressed as *Merge/Purge Problem* or *Data Cleaning* in commercial terminology. Instance matching issues heavily involve aspects related to data quality. According to the classification given in [25] data quality problems arise both in single and multiple data sources. Regarding instance matching in single sources, schema-related data quality problems occur because of the lack of appropriate model-specific or application-specific integrity constraints (e.g. due to data model limitations or poor schema design) or because only a few integrity constraints were defined to limit the overhead for integrity control. Instance-specific problems, on the other hand, refer to errors and inconsistencies in the actual data contents which are not visible and which cannot be prevented at the schema level (e.g., misspellings). Frequently, data contains errors or may have multiple representations, such as abbreviations, so an exact comparison is not sufficent for the detection of duplicates in those cases.

When multi-sources are matched and integrated, schema-level issues are related to the sphere of linguistic and structural analysis. Concepts and roles are generally featured by significant names that give a self-explanation of the semantic meaning in terms of natural language. Naming conflicts arise when the same name is used for different objects (homonyms) or different names are used for the same object (synonyms). On the other hand structural conflicts occur in many variations and refer to different representations of the same object in different sources, e.g., attribute versus table representation, different component structure, different data types, different integrity constraints. Studies related to instance matching have followed many different directions, each focusing on a particular issue of the process and aiming to reach different goals. Some researches focus on attribute values comparison in order to address issues related to different value representations and different interpretation of the values across heterogeneous sources. A solution discussed in [26] exploits an external ontology for representing conversion rules between values related to specific domains (e.g. currency change) while in [27] a statistics-based system is proposed for the the specific domain of financial data that estimates conversion functions using regression analysis and creates data value conversion rules. Data entry errors or misspellings are effectively detected with the use of string matching [28, 29], eventually with the help of domain specific rules. Duplicate detection is generally carried out by means of clustering algorithms ([30–32]) which differ mainly for the grouping criteria adopted. Other solutions take in account connections and dependencies between different instances or attributes ([33, 34]), by exploiting structural properties or contextual information derived from the schema.

When applied to ontologies, such techniques require the help of a reasoning service to obtain all individuals valid instances of a given concept. Roles establish connections and dependencies between instances in a more structured way than in databases and there is the lack of a predefined set of key attributes. These considerations introduce new information and constraints on the instance matching process when dealing with ontologies, and some initial research is appearing also in this field. A work concerning instance matching on ontology sources is proposed [35]. It is based on two measures: concept distance, that is the length of the path between two concepts according to

the computed concept hierarchical tree, and context similarity, which is computed by reasoning on the inverse properties and checking the cardinalities on them. A more focused application of instance matching is represented by *Instance-Based Schema Matching* ([36], [37]), a particular category of schema matching methods which can replace or complement other schema matching approaches. These approaches compare two entities on the basis of the affinity degree evaluated between all their instances: the more instances are similar, the more similar the two entities are considered. Some solutions are studied to determine the nature of attributes by testing if all existing values satisfy some predefined constraints or exhibit same features, according to a given datatype category.

Instance matching has the goal of comparing individuals described in different Aboxes defined with respect to the same Tbox, while concept matching has the goal of aligning different and independent ontologies with heterogeneous Tboxes. In particular, in instance matching we address misspellings, especially for names of concepts (e.g., athletes, events), and redundancy, duplicates, and contradictory values with the goal of identifying instances which describe the same real object or event.

9 Conclusions

In this article, we have presented and analysed a methodology for multimedia ontology evolution. In particular, we have discussed the role of reasoning and matching techniques for semantic interpretation of multimedia resources and for the coordinated and consistent evolution of the ontology. We have described how new resource interpretations can be correctly framed in the ontology and how new concepts can be defined to increase the background knowledge of the system by means of the population or enrichment activities, respectively. The evaluation demonstates the feasibility of the approach.

Further investigation are in progress on the enrichment activity, in order to study a combination of matching, clustering, and reasoning techniques to address the goal of providing a full support of the domain expert in the definition of new concepts. In future work, also the interpretation rules should be learned from multimedia interpretation, such that it can be evaluated whether the methodology also provides for better media interpretation.

The implementation of an ontology evolution toolkit is in progress as well, with the aim of providing an interactive environment where all the proposed techniques will be integrated into a coherent whole according to an open architecture in order to provide a new content management system for multimedia resources, improved by automatic classification, acquisition, and maintenance functionalities, together with a ontology management tool with editing, reasoning, and evolution functionalities.

References

1. Alani, H., Kim, S., Millard, D., Weal, M., Hall, W., Lewis, P., Shadbolt, N.: Automatic ontology-based knowledge extraction from web documents. IEEE Intelligent Systems **18** (2003) 14–21

2. Buitelaar, P., Cimiano, P., Racioppa, S., Siegel, M.: Ontology-based information extraction with soba. In: Proceedings of the International Conference on Language Resources and Evaluation. (2006) 2321–2324
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press (2003)
4. Baader, F., Nutt, W.: Basic description logics. In Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F., eds.: The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press (2003) 43–95
5. Thagard, R.P.: The best explanation: Criteria for theory choice. The Journal of Philosophy (1978)
6. Castano, S., Ferrara, A., Montanelli, S.: Matching ontologies in open networked systems: Techniques and applications. Journal on Data Semantics (JoDS) **V** (2006)
7. Bruno, S., Castano, S., Ferrara, A., Lorusso, D., Messa, G., Montanelli, S.: Ontology Coordination Tools: Version 2. Technical Report D4.7, BOEMIE Project, FP6-027538, 6th EU Framework Programme (2007)
8. Hartigan, J.: Clustering algorithms. John Wiley (1975)
9. Salton, G.: Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley (1989)
10. Espinosa, S., Kaya, A., Melzer, S., Möller, R., Wessel, M.: Towards a media interpretation framework for the semantic web. The 2007 IEEE/WIC/ACM International Conference on Web Intelligence (WI'07) (2007)
11. Haarslev, V., Möller, R., Wessel, M.: RacerPro User's Guide and Reference Manual Version 1.9.1 (2007)
12. Castano, S., Ferrara, A., Messa, G.: ISLab HMatch Results for OAEI 2006. In: Proc. of ISWC Int. Workshop on Ontology Matching, Athens, Georgia, USA (2006)
13. Ding, Y., Foo, S.: Ontology research and development. part 2 - a review of ontology mapping and evolving. Journal of Information Science **28** (2002) 375–388
14. Haase, P., Stojanovic, L.: Consistent evolution of owl ontologies. In Gómez-Pérez, A., Euzenat, J., eds.: ESWC. Volume 3532 of Lecture Notes in Computer Science., Springer (2005) 182–197
15. Oberle, D., Volz, R., Motik, B., Staab, S.: An extensible ontology software environment. In Staab, S., Studer, R., eds.: Handbook on Ontologies. International Handbooks on Information Systems. Springer (2004) 311–333
16. Aliseda-Llera, A.: Seeking Explanations: Abduction in Logic, Philosophy of Science and Artifical Intelligence. PhD thesis, University of Amsterdam (1997)
17. Paul, G.: Approaches to Abductive Reasoning–An Overview. AI Review 7 (1993) 109–152
18. Mayer, M.C., Pirri, F.: First-Order Abduction via Tableau and Sequent Calculi. Bulletin of the IPGL **1** (1993) 99–117
19. Shanahan, M.: Perception as Abduction: Turning Sensor Data Into Meaningful Representation. Cognitive Science **29** (2005) 103–134
20. Neumann, B., Möller, R.: On Scene Interpretation with Description Logics. In Christensen, H., Nagel, H.H., eds.: Cognitive Vision Systems: Sampling the Spectrum of Approaches. Number 3948 in LNCS. Springer (2006) 247–278
21. Elsenbroich, C., Kutz, O., Sattler, U.: A case for abductive reasoning over ontologies. In: Proc. OWL: Experiences and Directions, Athens, Georgia, USA, November 10-11. (2006)
22. Hobbs, J.R., Stickel, M., Appelt, D., Martin, P.: Interpretation as abduction. Artificial Intelligence Journal **Vol. 63** (1993) 69–142
23. Kakas, A., Denecker, M.: Abduction in logic programming. In Kakas, A., Sadri, F., eds.: Computational Logic: Logic Programming and Beyond. Part I. Number 2407 in LNAI. Springer (2002) 402–436

24. Euzenat, J., Shvaiko, P.: Ontology matching. Springer-Verlag, Heidelberg (DE) (2007)
25. Rahm, E., Do, H.H.: Data cleaning: Problems and current approaches. IEEE Data Eng. Bull. **23** (2000) 3–13
26. Zhang, D., Jing, L.: Context-based integration of numerical information. In: CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05), Washington, DC, USA, IEEE Computer Society (2005) 418–421
27. Fan, W., Lu, H., Madnick, S.E., Cheung, D.: Discovering and reconciling value conflicts for numerical data integration. Inf. Syst. **26** (2001) 635–656
28. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady **10** (1966) 707–710
29. Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S.: Adaptive name matching in information integration. IEEE Intelligent Systems **18** (2003) 16–23
30. Carey, M.J., Schneider, D.A., eds.: The Merge/Purge Problem for Large Databases. In Carey, M.J., Schneider, D.A., eds.: Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 22-25, 1995, ACM Press (1995)
31. Monge, A.E., Elkan, C.: An efficient domain-independent algorithm for detecting approximately duplicate database records. In: DMKD. (1997) 0–
32. Bertolazzi, P., DeSantis, L., Scannapieco, M.: Automatic record matching in cooperative information systems. In: Int. Workshop on Data Quality in Cooperative Information Systems. (2003)
33. Singla, P., Domingos, P.: Multi-relational record linkage. In: MRDM Workshop. (2004)
34. Ananthakrishna, R., Chaudhuri, S., Ganti, V.: Eliminating fuzzy duplicates in data warehouses. In: Proceedings of the 28th International Conference on Very Large Databases (VLDB 2002). (2002)
35. Wang, C., Lu, J., Zhang, G.: Integration of ontology data through learning instance matching. In: Web Intelligence. (2006) 536–539
36. Engmann, D., Massmann, S.: Instance matching with coma++. In: BTW Workshops. (2007) 28–37
37. Chua, C.E.H., Chiang, R.H.L., Lim, E.P.: Instance-based attribute identification in database integration. The VLDB Journal **12** (2003) 228–243