

# Semantics extraction from multimedia data: an ontology-based machine learning approach

Sergios Petridis and Stavros J. Perantonis

**Abstract** It is often the case that related pieces of information lie in adjacent but different types of data sources. Besides extracting such information from each particular type of source, an important issue raised is how to put together all the pieces of information extracted by each source, or, more generally, what is the optimal way to collectively extract information, considering all media sources together. This chapter presents a machine learning method for extracting complex semantics stemming from multimedia sources. The method is based on transforming the inference problem into a graph expansion problem, expressing graph expansion operators as a combination of elementary ones and optimally seeking elementary graph operators. The latter issue is then reduced to learn a set of soft classifiers, based on features each one corresponding to a unique graph path. The advantages of the method are demonstrated on an athletics web pages corpus, comprising images and text.

## 1 Introduction

It is often the case that related pieces of information lie in adjacent but different types of data sources, such as text, images, audio and video streams. Automatically extracting such information involves being able to analyse each particular type of source, which is undeniably a challenging task and as such the focus of devoted research. However, an additional issue raised in this case is how to put together all the pieces of information extracted by each source, or, more generally, what is the optimal way to collectively extract information, considering all media sources together.

Being of wide applicability, this issue has been tackled from several different angles and under different constraints. This accounts for a variety of names used to

---

Sergios Petridis and Stavros J. Perantonis  
Institute of Informatics and Telecommunications, NCSR "Demokritos", Greece, e-mail:  
(petridis, sper)@iit.demokritos.gr

identify it in the literature, such as the combinations of any of the terms below:

$$\begin{bmatrix} \text{homo(hetero)geneous} \\ \text{(a)synchronous} \\ \text{(non)commensurated} \end{bmatrix} \times \begin{bmatrix} \text{sources} \\ \text{data} \\ \text{information} \\ \text{sensor} \\ \text{decision} \end{bmatrix} \times \begin{bmatrix} \text{fusion} \\ \text{integration} \\ \text{merging} \\ \text{combination} \\ \text{assimilation} \end{bmatrix}$$

In this chapter, we use the term *fusion* and deal with the most general setting, where data sources are not necessary homogeneous, neither do they need to be of a particular type. As will be discussed later, this prompts for a methodology that deals with fusion at high level, i.e. where a substantial analysis of each data source has been preceded before bringing the results together.

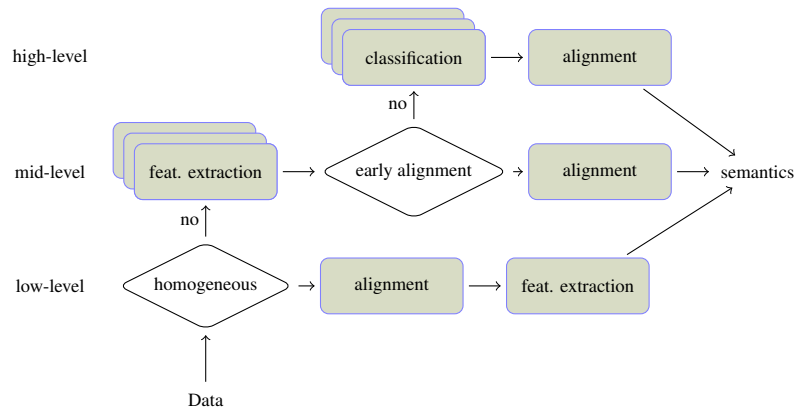
Our approach uses effectively the machine learning/soft classification paradigm to drive approximate inference in ontologies. It allows to extract semantics from multimedia data, though the core algorithm is general and may be applied to any kind of data, once these are in form of ontological assertions. It has three significant features. First, it deals inherently with uncertainty, allowing to represent both uncertain data and rules. Second, it is completely learnable, allowing for its adaptation to any domain and any ontology. Third, its application has polynomial complexity which makes it suitable for processing large scales of data.

To the best of the authors knowledge, this is the first approach in the literature that uses machine learning classification algorithms to infer semantics for data extracted from multimedia content. The significant benefits of using this method are that:

- it allows the model to be learned using a training set of input-output assertions set, instead of manually building axioms and rules,
- it takes into account the uncertainty of assertions during inference,
- it benefits from existing mature soft classification literature to infer complex structured ontological knowledge,
- it balances complexity vs accuracy by means of the size of subgraphs considered during evaluation and the classification models used, as will be explained later, and
- it guarantees polynomial complexity.

Note that, independently of the approach to jointly handle multimedia data, a very important part of the semantics extraction from multimedia content task is delegated to media-specific analysis algorithms. The effectiveness of these methods are, needless to say, crucial for the overall performance of the fusion approach. In this chapter, we will not however go into details of any such algorithm, but rather concentrate on the fusion of their outcomes.

This chapter has the following structure. Section 2 describes the general setting of the fusion methodology, puts forward the advantages of semantic fusion and explains the steps required for expressing extracted multimedia information in the semantic level. Section 3 analyses our methodology for extracting high level semantic knowledge from multimedia data within the machine learning framework. In Section 4, we describe an evaluation procedure and analyze the performance of



**Fig. 1** Levels of fusion. Notice that the difference lies in which order does the alignment step takes place.

our methodology. Finally, Section 5 gives a brief review of related work and Section 6 summarizes our concluding remarks.

## 2 Fusing at the semantic level

This section describes the semantic setting under which we consider multimedia data fusion of. In Section 2.1 the differences of low, mid and high-level fusion are highlighted. Section 2.2 analyses the concepts of redundancy and complementarity when fusing at the semantic level and Section 2.3 describes a method of representing location related information between medium segments by means of logical relations. Finally, Section 2.4 proposes a decomposition-fusion approach to efficiently deal with practical implications of a multimedia analysis system.

### 2.1 Low, mid and high-level fusion

Fusion approaches that may be used to jointly extract information from multimedia data are generally categorized in three types: low-level fusion, mid-level fusion and high-level fusion. These differ at the level of representation in which fusion takes place. The reader may refer to Figure 1 for an overview of their differences.

### 2.1.1 Low-level Fusion

When sources are *homogeneous* i.e. of the same type, the fusion approach may take advantage of very specific techniques, developed for the specific data structure and data structural elements. A characteristic case is *image fusion* [2], where the correspondence between images can be done on a pixel basis either in the spatial or the spectral domain. This enables the use of medium-specific algorithms (e.g. the *discrete wavelet transform* in images), to be applied on the fused data.

Notice that, low level fusion demands data to be prior aligned. Sometimes, this task has a difficulty of its own and appropriate techniques should be devised. As a particular example, in image fusion, the process of spatially aligning images in order to correctly fuse them is referred to as *image registration* [5].

### 2.1.2 Mid-level fusion

Another case arise when the considered data, although not homogeneous, are alignable, i.e. one can establish a correspondence between their structural elements. A typical case of alignable data are audiovisual documents, where video and audio are aligned on their time dimension. These are typically stored using a combination of compressions techniques and a container format (e.g. MP4). Decomposing an audiovisual document into its audio and video parts is a rather straightforward procedure, if one knows the particular container format that is used.

A possible approach here is to homogenize the elements by extracting features from all different media [1]. Since feature-vectors are medium-neutral, they can be concatenated, thus producing a unified feature vector for the joint multimedia element. Subsequently, one may apply algorithms specific to the common multimedia document structure.

Notice that modality feature extraction rates may differ, due to different sampling rates. In this case, aligning the data may require some effort, such as element-wise interpolating the features of one modality to the the others modality frame rate. On the other hand, this approach has the advantage of allowing the application of medium-specific techniques to extract useful features from each medium, while retaining the structural alignment of the media.

This approach is commonly referred to as *mid-level* fusion or *feature-level* fusion. Note, however, that as an alternative to feature extraction, one may as well use kernel-based techniques to combine data from different sources, as long as these have been aligned.

### 2.1.3 High-level fusion

Another way to proceed is to defer the fusion between corresponding elements to a common, medium neutral, symbolic level. This implies that a complete analysis is done in each medium, aiming to achieve information at a symbolic level of repre-

sentation prior to fusion. This method is referred to as *high-level* fusion, though the terms *symbolic-level* fusion or *decision-level* fusion are sometimes used. A characteristic of this approach is that it does not impose feature vector generation from each medium, but allows for particular methods to be applied independently until the symbolic level is reached.

An important advantage of high-level fusion is that it can handle the case of weaker alignment of documents. In particular, it is possible that alignment of documents can be done in coarse documents elements, within which the generated features by each medium can change significantly. As an example, in audiovisual speech recognition, synchronization may be achieved in the phoneme or syllable level, rather than in static sub-phonemic audio elements from which the features are traditionally extracted. A number of approaches relying on this weak alignment, such as *Coupled Hidden Markov Models* and *Factorial Hidden Markov Models* have been successfully applied [7].

The methodology presented in Section 3 falls into the high-level fusion type, since it relies on symbolic results obtained by separately analyzing each medium. In particular, fusion takes place at the semantic level, since it is assumed that the extracted results are described as a set of description logics assertions conforming to an assumed ontology.

## 2.2 Redundancy and complementarity of multimedia information

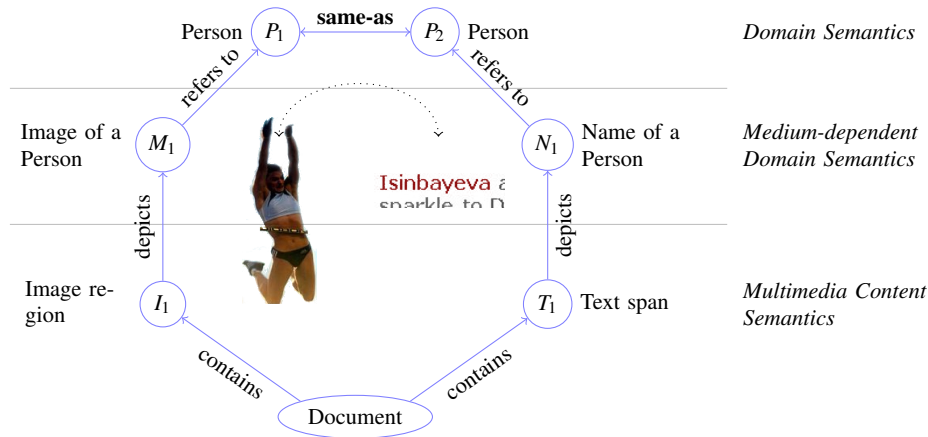
There are two good reasons for jointly extracting information from multimedia sources at the semantic level: *complementarity* and *redundancy*. We will briefly review here these two concepts.

### 2.2.1 Complementarity

Complementarity of information refers to the case where different pieces of related information are expressed in different media. Hence the integral information, together with further implied information it induces, may be gained only by their joint consideration.

Here is an example. Assume that a system can detect and analyze, though not recognize, persons in photos and person names in text. Then by jointly analyzing a photo showing a person face together with the caption “John Lennon”, the system may additionally infer that John Lennon was wearing glasses, something that cannot be deduced neither solely by the text neither solely by the photo.

To fuse concepts across modalities, an indirect matching of instances extracted across modalities takes place. Namely, each medium-specific algorithm aims at extracting instances of medium-specific concepts. These are typically results of classifications among a set of possible labels. For example, an image processing algorithm may detect a photograph of a *person's face* whereas the text processing



**Fig. 2** Fusing multimedia concepts. The image and text samples correspond to a web page containing a captioned image. The concepts and relation names used in this figure are indicative. Notice that data of different modalities are indirectly linked at the fusion level via the *same-as* relation.

algorithm may detect a written form of a *person's name*. The existence of medium-specific instances imply the existence of referring medium-independent instances. For example, both *person's face* and *person's name* imply the existence of a *person*, although not necessarily the same one. In these terms, finding correct pairs of medium-independent instances (i.e. persons) is the goal of the fusion algorithm. The relation used in ontology to denote instance matching is termed *same-as*. Figure 2 depicts the instance matching procedure.

### 2.2.2 Redundancy

Redundancy of information refers to the case where information regarding the same object is expressed in different media sources, and hence information pieces may be either confirming or contradicting to each other in various degrees. In this case, jointly extracting the information may lead to increasing the accuracy of the extracted information.

Here is another example. Assume that a system recognize an object in a photo, but gives similar probabilities that is a candy or a bow tie. Adjacent to the photo, there is a caption displaying the text “a red bow”, which the system may relate it to either a weapon, a music instrument, a ship’s bow etc. By combining the information from the text and image sources, the system may infer with great confidence what could not be inferred by considering each media in isolation: that the referenced object is a red bow tie.

### 2.3 *Physical and Logical document structure*

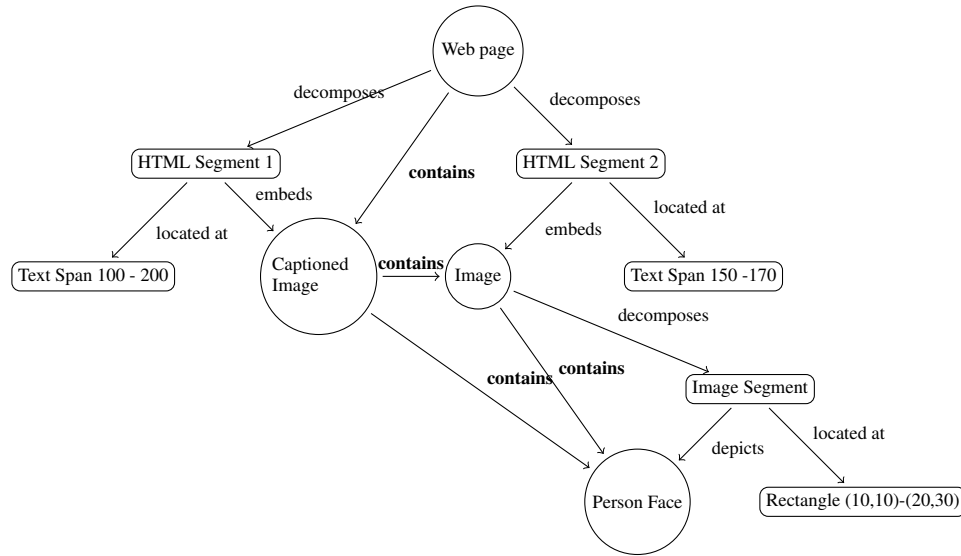
An inherent characteristic of multimedia data is that they can be localized by means of spatiotemporal locations within a specific context, such as a document. A specifically-crafted ontology may provide adequate means for specifying such a *physical* structure of a multimedia document. In particular, consider the *decompose* relation to denote a segmentation of image into image regions, an audio stream to audio segment etc. Clearly, each localisation is medium-specific and hence particular concepts have been used to allow describing the location of a segment in function of the medium used. Note also that information within a given segment may as well be represented in different modality, which fact accounts for the usage of the *embed* relation in the ontology. As an example, an image segment may contain textual information, which once detected and extracted by means of image Optical Character Recognition (OCR) techniques, is further subject to text analysis.

On the other hand, one should also stress that any document can be decomposed into a *logical* structure of elements which represent how distinct components of the document are logically related to each other. Such logical relations may be denoted via a distinct relation *contains*. For instance, a web-page can be decomposed into a set of text items and captioned images items, where each captioned image is further decomposed into an image and a caption. What is worth noticing about the logical structure, as opposed to the physical one, is that it can be defined in a medium-independent way. This holds because the logical structure between components, although induced by the physical structure, does no longer require medium-specific location information.

For instance, most of the times, the caption is a piece of text found *spatially* nearby an image. This physical relation can be exploited to derive a logical relation between these two items, namely that the first has the role of a caption and the second the role of an image in a captioned image entity. Once this relation has been discovered, we can consider to omit from further consideration how the text and image items are related spatially, and hence retain only a cross-media uniform logical representation.

Based on the above, prior to applying the fusion methodology described in Section 3, the physical relation between data items is taken into account in order to discover higher level logical relations between them (see also the graph at Figure 3). Namely, the following steps are applied:

1. Analysis results corresponding either to the entire document under examination or to its particular parts of it are taken into account. E.g. When dealing with web-pages, the results of text analysis of textual content, the results of image analysis of images and the results of decomposing the web-page in coherent segments are considered.
2. Common logical relation between components of the image are derived based on physical relation of segments. To give a simplified example, when text analysis results refer to the fact that there is a person name between the characters say 1000 and 1010 of the html page, on one hand, and the html analysis results refer



**Fig. 3** Abstracting structural relation This figure depicts an example of deriving a logical relation between data based on their physical relation. Notice that, once the *contains* relations are inferred using location information, only the concepts denoted with circles remain important for consideration.

to the fact that there is a caption between characters 900 and 1020, then the fact that the given person name is contained in the given caption is derived.

3. Once logical relations are extracted, all other medium-specific location information regarding the analysis is discarded.

## 2.4 Practical considerations

In several occasions, such as in OCR embedded text in images or web pages, multimedia data are not given as separate sources but as an integral multimedia document. In this section we present a general engineering approach to deal with practical issue when dealing particularly with such multimedia documents. This approach, refer to as the *decomposition-fusion* approach, has been successfully applied in the BOEMIE system [10] to semantically annotate audiovisual streams and web pages.

Namely, the decomposition-fusion approach consist of three steps:

- decompose** In a first step, the multimedia document is decomposed in its constituent parts using an appropriate algorithm, depending on the nature of the document. Decomposing a multimedia document refers to both finding its single medium elements as well as their relative structural position.

- analyze** Then, each constituent part of the multimedia document is given for analysis to appropriate single medium processing techniques.
- fuse** Finally, the results produced by individually processing the single-medium elements of the document, together with the information regarding the decomposition of the multimedia document, are jointly given as input into the “fusion” module that aims at providing the final outcome, i.e. the extracted semantics for the whole multimedia document.

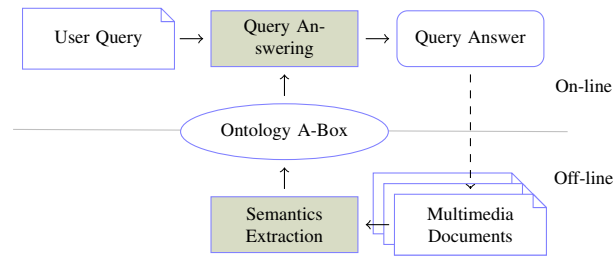
Importantly, the interaction between the decomposition, the analysis and the fusion step is accomplished through the usage of the same semantic model, i.e. as ontological assertions conforming to a specially-crafted ontology that accounts for both the semantics needed to describe the domain of application as well as those to describe the multimedia content structure.

Note that an important feature of this approach is that it decouples the media-specific analysis algorithms from the multimedia decomposition and fusion approaches. This has the significant advantage of allowing the overall semantics extraction process to be open to new achievements in single-medium data analysis, while also enabling to focus on enhancing the algorithms that perform the fusion step.

To give an example of the procedure, the initially given multimedia document (eg. a web page), is processed by a *decomposition* module that separates it into sub-documents corresponding to different media formats (i.e. video, audio, image and text).

Each one of these sub-documents is then given for analysis to the corresponding analysis module. The analysis modules, based on the semantic model, analyze the sub-documents so as to identify and classify elements within, such as 2-dimensional regions in images, word sequences in text, temporal segments in audio and spatio-temporal segments in video. The set of allowable classification labels, corresponds to a subset of the ontology concepts which are directly identifiable in the media. The analysis also identifies relations between the extracted elements, such as *adjacent to* in image, *before* in audio or *subject of* in text. The output of the analysis modules is a set of assertions containing the list of the extracted elements and element relations together with sufficient properties to describe the related instances, the position of the elements in the sub-document, the extracted features used to conduct the analysis, as well as the confidence estimation of the classification. Then, a *result analysis* module, analyses the results in order to re-route potentially identified encapsulated media (such as OCR extracted text) to the cross-modality coordinator for further analysis. Finally, once all single media sources have been processed, fusion takes place to extract further semantics based on the complementarity and redundancy of the sources.

**On-line / Off-line modes** A common use of semantics extraction systems is to enable answering complex queries over the multimedia content. To give an example, an end-user of such a system may wish to search for particular multimedia content that satisfy some criteria. Since multimedia analysis may be a time-consuming task, it is not recommended to perform it once the user has placed a particular query,



**Fig. 4** Querying semantics extraction results. A principal goal, that has to be taken into account by the fusion approach, is to allow answering semantically rich user queries.

Rather, it is better to have previously analyzed the content and match the query against the saved analysis results. Figure 4 depicts this approach, distinguishing the steps that take place in an off-line mode, from those taking place in an on-line mode.

### 3 Methodology

In this section, we analyze our approach to extract semantics from fused multimedia data. Section 3.1 explains our motivation for using the machine learning framework for inferring structure knowledge. In Section 3.2 we formally define the semantics extraction problem as a system identification problem and in Section 3.3 we show how it may be translated as a graph expansion problem. In Section 3.4 we study how to decompose the problem in elementary graph expansion operators and, finally, in Section 3.5 we propose metrics to learn optimal ones.

#### 3.1 Motivation

The approach presented is based on knowledge expressed in term of ontological assertions. However, it avoids using exact reasoning. In deed, clarity and unambiguity in ontologies has allowed reasoning to be embedded in ontologies, in the form of the description logics formalism. Exact reasoning may however hinder using ontologies for large scale of data, due to the high computational times it demands: problems to be solved with exact reasoning have typically EXPTIME or even NEXPTIME complexity. This becomes a critical issue when extracting semantics from multimedia content, which may require reasoning over thousands of instances. To overcome this issue, the approach followed here is based on approximate inference.

On the other hand, approximate inference sacrifices soundness or completeness for a significant speed-up of reasoning [12]. In addition, it fits more with data, such as results of a multimedia extraction algorithm, which are inherently uncertain. In

deed, it does not make perfect sense to perform exact reasoning with input that are not exact: (a) valuable information regarding the uncertainty of data is lost during exact reasoning and (b) the fact that reasoning results are exact does not make them correct, since the input to reasoning may not be correct itself.

Another critical factor for successfully applying ontologies in real world problems is the complexity of building ontologies. Research on this issue has made several advances, such as constructing subsumption hierarchies from text corpora [14], though, research for inducing axioms and/or safe rules based on the supervised machine learning paradigm has not yet received significant interest. In what follows, we will show how to build ontologies based on a reference set of assertions, where a subset of them, the *input*, would be some given explicit knowledge and another subset, the *output*, would be the knowledge induced by the axioms/rules to be learned.

Our approach has a significant difference in respect to the traditional approach for inference within ontologies. Namely, ontologies are commonly viewed as systems that contain implicit information, whereas our approach is based on considering them as input-output systems. In particular, we consider DL axioms, DL safe rules and abduction rules as particular ways of performing a single task: given a set of assertions  $A$ , and a model  $T$ , generate a set of assertions  $A^+$ , which is a superset of the original set of assertions  $A$ . Considering ontologies as a particular class of input-output systems may not be very convenient in cases where there is a need to keep knowledge only in its implicit form. However, it may be convenient as a way to integrate methods that share this input-output view, such as machine learnable classifiers. The issue then reduces to investigating how such methods may fit into the inference within the ontology framework.

### 3.2 Problem Formulation

In this section we formulate the approximate inference in ontologies problem as an input-output problem.

Let an A-box  $A$  be a set of assertions which are valid according to a DL vocabulary of terms (concepts and roles) and let  $\mathcal{A}$  be the set of all such A-boxes. Let also a family of functions:

$$\mathcal{T} = \{t : \mathcal{A} \rightarrow \mathcal{A} : \forall A \in \mathcal{A}, t(A) \in S(A)\}, \quad (1)$$

i.e. the family of “operators” that, given an A-box, they generate an A-box that belongs to  $S(A)$ , where  $S(A)$  is the set of all A-boxes that are supersets of  $A \in \mathcal{A}$ , i.e. they contain *at least* the same assertions as  $A$ . Let also a family of approximation functions,

$$\mathcal{F} = \{f : \mathcal{A} \rightarrow \mathcal{A}\}. \quad (2)$$

Given a reference set of  $N$  A-boxes  $\{A_i\}_{i=0}^N, A_i \in \mathcal{A}$ , a function  $t \in \mathcal{T}$  and a distance metric between A-boxes  $d : \mathcal{A} \times \mathcal{A} \rightarrow \mathfrak{R}$ , find the optimal function  $\hat{f} \in \mathcal{F}$ , such that the expected distance between A-boxes is minimal:

$$\hat{f} = \arg \min_{f \in \mathcal{F}} E_i [d(f(A^i), t(A^i))] \quad (3)$$

### 3.2.1 Reference functions

Note that family of functions  $\mathcal{T}$  may be constrained to particular procedures that are valid in the DL formalism:

- A first family is one that allows making explicit the implicit information deduced through the DL axioms to the existing assertions.
- A second family is the one that corresponds to the application of a set of DL-safe rules.
- A third family is the one that corresponds to the application of abduction rules in ontologies, which results in the addition of new concept instances.

In all cases, any assertion box  $t(A)$  is assumed to be valid in respect to the given ontology. Notice also that many DL axioms and/or rules correspond to operators  $t$ , where  $t(A)$  may have infinite elements for some  $A$ .

### 3.2.2 Approximation functions

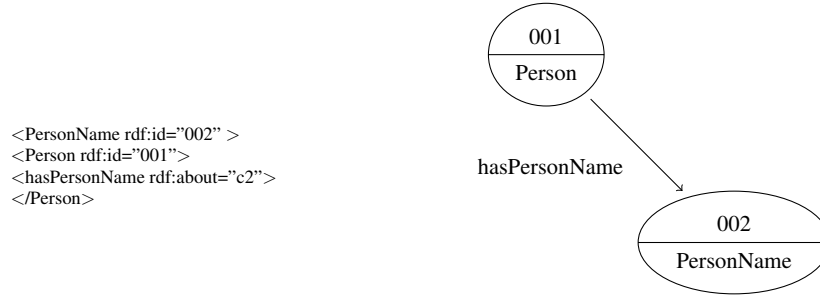
On the other hand,  $\hat{\mathcal{T}}$  is a family of functions that approximate functions in  $\mathcal{T}$ , but need not be the same as  $\mathcal{T}$ . Any set of operators that, given a set of assertions, produce a superset of these assertions, according to the given vocabulary of concepts and roles, may be used. The family of functions that is described later in this section is of quite different nature than the ones employed in logic-based systems.

Approximating  $\mathcal{T}$  by  $\hat{\mathcal{T}}$  may lead to sacrificing soundness or completeness of the results: although  $t(A)$  is always sound and complete,  $\hat{f}(A)$  may not be. However a careful selection of  $\hat{\mathcal{T}}$  may have other benefits, such as:

- $f(A)$  may be much faster to evaluate than  $t(A)$ ,
- functions in  $\hat{\mathcal{T}}$  may handle uncertainty regarding the input assertions, and postpone the thresholding step to until after the inference and
- $\hat{f}(A)$  may have nice properties, such that the optimal function  $\hat{f}$  can be found through *learning*, i.e. through the reference set of A-boxes.

### 3.2.3 Distance between A-boxes

Note that defining the problem requires providing a distance metric that measures the distance between two set of assertions. In what follows, we discuss such measures based on the isomorphism between DL A-boxes and directed graphs. As a preliminary comment, one should take care that the distance should reflect the degree of semantic equivalence between the two sets of assertions.



**Fig. 5** A set of DL assertions as a graph. PersonName, Person and hasPersonName are terms included in the T-BOX of the ontology

### 3.3 Using directed graphs

#### 3.3.1 Set of DL assertions as directed graphs

In this section, we establish a one-to-one correspondence between set of DL assertions  $A$ , commonly referred to as an Assertion box (A-box) and a directed graph  $G = (V, A)$  with vertices  $V$  and directed edges  $A$  belonging to the family of directed graphs  $\mathcal{G}$ , where vertices take values in  $\mathcal{V}$  and edges take values in  $\mathcal{E}$ . We hence show that sets of description logics assertions are isomorphic to directed graphs. This will allow us to deal with the problem formulated in Section 3.2 by means of functions where both their input and outputs are graphs.

Let a pair of mappings  $M$  and  $M^{-1}$  from A-boxes to graphs and vice-versa, such that

$$\forall A \in \mathcal{A} : A = M^{-1}[M[A]] \quad (4)$$

The mapping discussed here involves only the assertions, leaving out the terminological part of the ontology. The mapping is as follows:

- Every individual is mapped to exactly one vertex in the graph. The value of the vertex is (a) the unique identifier of the individual and (b) information regarding the concept that the individual is member of. In the simplest case, this is the name of the concept. In this case,  $\mathcal{V}$  is the set of all concepts of the ontology.
- Every statement that a *relation* holds between *two individuals* is mapped to exactly one edge in the graph, namely the edge that connects the respective vertices. The edge is directed, reflecting the fact that the relation between instances is not necessarily symmetric. The value of the edge is information regarding the type of relation holding between individuals. In the simplest case, this is the unique name of the relation, and thus  $\mathcal{E}$  is the set of all relation types in the ontology.

Figure 5 shows an example of a some DL assertions expressed as OWL statements and the respective graph. There are overall two instances (001, Person and 002, PersonName) and one relation that holds between the instances (hasPersonName).

*The inverse mapping* It is quite straightforward to define the inverse mapping: each vertex maps to an individual of identifier and concept membership as provided through the vertex values, while edges are mapped to relation between two individuals, as indicated by the edge value.

Notice that the mapping described allowing to preserve the instances unique identifiers. Application of the inverse mapping to the mapped A-box results then to exactly the same A-box. Were the information regarding the unique identifiers of the individuals omitted, the inverse mapping would generate *semantically* equivalent assertions as the original ones, although with different unique identifiers for the individuals.

**Problem reformulation** The problem discussed in Section 3.2 involves sets of description logic assertions. Since sets of description logics assertions and graphs are isomorphic, a way to proceed, in order to solve it, is to first formulate it using graphs, find optimal graph operators and then transform back the optimal graphs into set of assertions. The first and last step correspond to applying the mapping and inverse mapping respectively. The core problem, then, reduces to the following:

Let  $\mathcal{G}$  be a family of directed graphs. Let  $S(G)$  be the set of graphs that are supersets of  $G$ , i.e. graphs which have *at least* the same vertices and edges with  $G$ . Let also the family of graph functions

$$\mathcal{T} = \{t : \mathcal{G} \rightarrow \mathcal{G} : \forall G \in \mathcal{G}, t(G) \in S(G)\}, \quad (5)$$

i.e. specific “operators” that generate supersets of graphs. Let also a family of approximation functions,

$$\mathcal{F} = \{f : \mathcal{G} \rightarrow \mathcal{G}\}. \quad (6)$$

Given a reference set of  $N$  graphs  $\{G\}_{i=0}^N$  and a distance metric  $d : \mathcal{G} \times \mathcal{G} \rightarrow \mathfrak{R}$  between directed graphs, find the optimal operator  $\hat{f} \in \mathcal{F}$ , such that the expected distance between graphs is minimal:

$$\hat{f} = \arg \min_{f \in \mathcal{F}} E_i [d(f(G^i), t(G^i))] \quad (7)$$

Notice that we have used the same symbols as in Section 3.2 to denote functions and family of functions, since they are practically equivalent. Moreover, all observations regarding the family of functions  $\mathcal{T}$  and  $\mathcal{F}$  apply to the graph formulation. Notice also that the issue of defining a distance metric between set of assertions translates into defining a distance metric between directed graphs.

Since all operators  $f$  considered produce graphs that are supersets of the original graph, it makes sense to call them *graph expansion* operators. In what follows, we describe a way to find optimal such operators of a particular family.

### 3.4 Optimal graph expansion operators

#### 3.4.1 Elementary operators

Before examining how to find optimal graph expansion operators, let us first consider how the graph expansion functions may look like. Following the graph-based problem formulation, the graph functions considered result in graphs that have more edges and possibly more vertices than the ones given as input. It is easy to see that any such function can be decomposed as a recursive application of elementary graph expansions operators with two variants:

- An operator that adds an edge between two existing vertices that are not connected

$$h_1 = (e_{new} \leftarrow \text{addEdge}(v_i, v_j, e)) \quad (8)$$

where  $v_i, v_j$  are two vertices of the graph and  $e$  is the value of the new edge.

- An operator that adds a vertex to the graph, along with an edge that connects it to an existing vertex

$$h_2 = (v_{new} \leftarrow \text{addVertex}(v), (e_{new} \leftarrow \text{addEdge}(v_i, v_{new}, e)) \quad (9)$$

where  $v$  is the value of the new vertex,  $v_{new}$  is the newly inserted vertex,  $v_i$  is an existing vertex of the graph and  $e$  is the value of the edge to be inserted. In other words, denoting the family of elementary graph expansion operators as  $\mathcal{H}$ , any graph expansion operator may be expressed as

$$\forall f \in \mathcal{F}, \exists h_t \in H, t = 1 \dots T : f = h_1 \circ h_2 \circ \dots \circ h_T, \quad (10)$$

where  $T$  is the number of edges or vertices + edges added. In what follows, we exploit the decomposition of graph expansion operators in elementary ones in order to find optimal ones with respect to our problem.

Note that, by translating these operators back to the initial problem formulation, these variants correspond to adding a relation between two instances of the a-box or adding a new instance to the assertions box and connecting it to an existing one respectively.

#### 3.4.2 Greedy search for optimal operators

Expressing a graph expansion operator as a composition of elementary ones also suggest a way to find near-optimal graph expansion operators. In this work, we apply a greed approach which consists of the following steps:

- Step 0 (*initialization*):  $t = 0, G^t \leftarrow G$ .
- Step 1 (*optimization*): Find an optimal operator  $\hat{h}^t$  in respect to  $G^t$ .
- Step 2 (*expansion*) Set  $G^{t+1} \leftarrow \hat{h}^t(G^t)$
- Step 3 (*recursion*): If  $t < T$  set  $t \leftarrow t + 1$  and go to Step 1; else stop

where  $T$  is the number of steps, which may be either specified beforehand, or optimally decided using a threshold. Using this procedure, the optimal function found is

$$\hat{f}^{\text{greedy}} = \hat{h}_1 \circ \hat{h}_2 \circ \dots \circ \hat{h}_T. \quad (11)$$

Using the greedy approach has the benefit of simplicity and speed, though  $\hat{f}^{\text{greedy}}$  may *not* be as good as the optimal one  $\hat{f}$ . Alternative methods, as for instance those of the forward-backward family, could also be considered to improve the optimality of the final operator. The core approach, though, stays the same: taking into account the decomposition of the optimal operator as a sequence of elementary operators and trying to iteratively find optimal values of each one of them.

### 3.4.3 Optimal elementary operators

We now describe how to find the optimal elementary operator  $h$  that adds a single edge or a vertex and an edge to the graph. Our approach consists of three steps:

- Define a scoring function that evaluates how good the result of an elementary operator is. This step is where *learning* takes place. In our case, learning is conducted once using the training material, whereas the scoring function may be used repeatedly every time we search for the optimal elementary operator
- Exhaustively search at the domain of operators for an optimal elementary operators, and
- Keep the elementary operator that scores most.

The scoring function differentiates in respect to the operator variant. Namely:

### 3.4.4 Optimal Edge Addition

For the first elementary operator  $h_1$  that adds an edge between two existing vertices, the search space is  $V \times V \times \mathcal{E}$ . Optimal choice of the pair of vertices  $v_i$  and  $v_j$  as well as of the value of the new edge can be done by defining a *scoring* function  $S_1$

$$S_1 : V \times V \times \mathcal{E} \rightarrow \mathfrak{R} \quad (12)$$

Once this function is defined, the optimal operator can be chosen as

$$\hat{h}_1 = h_1(\hat{v}_i, \hat{v}_j, \hat{e}) = \arg \max_{v_i, v_j, e} S_1 [h_1(v_i, v_j, e)] \quad (13)$$

### 3.4.5 Optimal Vertex Addition

Similarly, for the second elementary operator  $h_2$  that adds a vertex and connects it to an existing vertex with an edge, the search space is  $V \times \mathcal{V} \times \mathcal{E}$ . Optimal choice

of the pair of vertex  $v_i$ , the value of the new vertex  $v$  as well as of the value of the new edge can be done by defining a *scoring* function  $S_2$

$$S_2 : V \times \mathcal{V} \times \mathcal{E} \rightarrow \mathfrak{R} \quad (14)$$

Once this function is defined, the optimal operator can be chosen as

$$\hat{h}_2 = h_2(\hat{v}_i, \hat{v}, \hat{e}) = \operatorname{argmax}_{v_i, v, e} S_2 [h_2(v_i, v, e)] \quad (15)$$

### 3.4.6 Complexity Issues

Since an exhaustive search is conducted, the complexity of performing a single elementary graph operation is bounded by  $O(|V|^2 \cdot |\mathcal{E}|)$  for the first operator and  $O(|V| \cdot |\mathcal{V}| \cdot |\mathcal{E}|)$  for the second one, where

- $|V|$  is the number of vertices of the graph,
- $|\mathcal{V}|$  is the size of vocabulary for vertices, i.e. the number of possible concepts, and
- $|\mathcal{E}|$  is the size of vocabulary for edges, i.e. the number of possible relations.

Notice that complexity is polynomial in respect to the graph quantities. This compares favourably, with the EXPTIME complexity which is commonly associated with exact reasoning approaches. However, the above complexity does not take into account the complexity of calculating the scoring function. To restrain the complexity in polynomial number steps, one should take care that calculating the scoring function is also bounded in polynomial number of steps. The issue of defining a scoring function is discussed in the next section.

## 3.5 Scoring functions for graph expansion operators

So far, we have re-formulated the problem as an input-output system identification problem, we have defined a mapping between set of assertions and directed graphs and stated the problem as finding an optimal graph expansion operator. We have then proposed to find such optimal operators by decomposing them in elementary ones and greedily find optimal elementary operators by exhaustively evaluating a scoring function for each one of them. Therefore, we have described the way approximate inference in ontologies may be reduced to defining scoring functions for elementary graph expansion operators. We now describe a way to define scoring functions.

The definition of a scoring function is crucial for the success of the approach. Since a scoring function is evaluated in each step, its properties affect significantly the properties of the overall approach. Good properties of a scoring function are:

- |              |  |
|--------------|--|
| learnability | It should be possible to adjust the scoring function parameters given a reference set of input and output a-boxes. |
|--------------|--|

low complexity	Evaluation of the function should be bounded by polynomial number of steps in respect to the graph quantities
uncertainty handling	Scoring should take into account uncertainty values that are possibly associated with the graph vertices and edges.

### 3.5.1 Graph Local Representations

The choice of a scoring function should be such that the resulting graph maximally resembles the one that would have been obtained through exact reasoning. By constrained the range of the scoring function to  $[0..1]$ , the score for an operator that adds an edge and/or vertex that would have been added by exact reasoning should be maximum, i.e. 1.0, whereas the score for an addition that would *not* have been should be minimum, i.e. 0.0. In other terms, the ideal scoring function for a proposed expanded graph should be inverse monotone to the *distance* between the proposed graph and the optimal graph. Note that learning and applying the ideal scoring function may be possible but prohibitively expensive in terms of complexity. Here, we propose a family of scoring functions that approximates the ideal ones by taking into account vertices and edges that are *close*, in term of graph distance, to the considered edge/vertex addition.

Namely, to insert an edge between two specific vertices  $v_1$  and  $v_2$ , one could take into account information from the whole graph. However, it makes sense to take *more* into account vertices and edges that are close to  $v_1$  and  $v_2$ , e.g. directly connected, and *less* into account vertices and edges which are connected via a long path. More formally, let  $\mathcal{X}$  be a convenient local representation of the graph and let the family of approximate local mappings of the graph *at the neighborhood of* a vertex as:

$$\mathcal{U} = \{u : \mathcal{G} \times V \times N \rightarrow \mathcal{X}\} \quad (16)$$

where

- $G \in \mathcal{G}$  is the input directed graph under consideration
- $v \in V$  is a particular vertex of the graph
- $n \in N$  is an integer that specifies the maximum path length for any other vertex of the graph, so that it is taken into account at  $u$ . For example,  $n = 1$  is equivalent to say that only direct edges and neighbors are consider at the local representation.
- $X \in \mathcal{X}$  is the local representation of the graph.

Depending on our choices,  $\mathcal{X}$  may be simple or sophisticated. Two trivial cases are the following:

1.  $X$  is the subgraph of  $G$  that contains  $v$ , its  $n$ -close neighbors and the relevant edges.
2.  $X$  is a “bag of words” consisting of the names of concepts and names of relations of vertices and edges that are around  $v$ .

The importance in defining the graph local representation is that the complexity of the learning and evaluation of the scoring function is controlled by the degree  $n$  of

locality considered in combination with the structure of the representation space  $\mathcal{N}$ . By letting  $n$  take small values, e.g.  $n < 3$ , and simplifying the structure, e.g. bag of words, the scoring function complexity is significantly reduced

### 3.5.2 Representing Graph Paths as Features

The particular graph local representation we have devised and used for our experiments consists of converting the local neighborhood of a graph around a vertex or edge into a vector of features, each one reflecting the path from the center vertex or edge to a vertex or edge within the neighborhood. Each path is considered distinct if the values of the vertices and edges that form it, in their particular order, are unique. Each distinct path corresponds to a distinct feature. The value of the feature, in a particular neighborhood, corresponds to the number of cases the respective type of path connects the central vertex to other vertices or edges.

Namely, let  $v$  and  $e$  be a vertex and an edge that correspond to a concept instance and a relation respectively. Let  $c[v]$  be the value of the vertex, i.e. the concept of the instance, and  $r[e]$  be the value of the edge, i.e. the type of relation. Then, the path

$$v_1 \xrightarrow{e_1} v_2 \xleftarrow{e_2} \dots v_n \quad (17)$$

corresponds to the feature

$$c[v_1], r[e_1], c[v_2], \bar{r}[e_2] \dots, c[v_n]. \quad (18)$$

where  $\bar{r}$  denotes the inverse relation. For convenience, since the concepts and relations are commonly given as unique alphanumeric strings, the unique name of the feature is made as a concatenation of the names they take part, in their specific order. In particular, the relation names are suffixed distinctively so as to denote whether the direct or inverse relation is part of the path.

### 3.5.3 Example

Consider the concepts `PoleVault`, `Pole` and `Pillar` and the relations `isAboveRight` and `isRightOf`. Let the vertices  $v_1$ ,  $v_2$  and  $v_3$  have value `PoleVault`, `Pole` and `Pillar` respectively. and the edges  $e_1$  from  $v_1$  to  $v_2$  and  $e_2$  from  $v_3$  to  $v_1$  value have values `isAboveRight` and `isRightOf`. Also assume that  $v_1$  is not connected to any other vertices in the graph. Then, the local neighborhood of vertex  $v_1$  is represented as follows:

- The value of the center vertex, `PoleVault`,
- The feature `isAboveRightS_Pole` has value 1,
- The feature `isRightOfO_Pillar` has value 1,
- All other features have zero value

Note that relation names have been suffixed with 'S' and 'O' to denote the direction of relation. Note also that the `PoleVaulteR`, i.e. the center value of the neighborhood is not part of the names of the features. Rather, this value is to be *predicted* based on the values of all other features. Assuming that the number of all other unique paths in the graph is  $M$ , and the features above correspond to the third and fifth, then the above information is compactly represented as

$$[0, 0, 1, 0, 1, \dots 0] \rightarrow \text{PoleVaulteR} \quad (19)$$

### 3.5.4 Representing Uncertainty

The uncertainty values of vertices and edges are used to derive an uncertainty value for each path and therefore for each feature. Namely, by letting  $w(v)$  and  $w(e)$  be the uncertain values of vertices and edges, the uncertainty value of 17 is evaluated by multiplying the uncertainties of elements in the path:

$$w(v_1) \cdot w(e_1) \cdot c(v_2) \cdot r(e_2) \cdots w(v_n).$$

Example

To continue the example above, if  $w(v_1) = 0.9$ ,  $w(v_2) = 0.9$ ,  $w(v_3) = 0.1$ ,  $w(e_1) = 0.5$  and  $w(e_2) = 0.9$ , then the uncertainty of the first feature is  $0.5 \cdot 0.9 = 0.45$ , and the uncertainty of the second feature is  $0.9 \cdot 0.1 = 0.09$ . In result, the overall information is represented as

$$[0, 0, 0.45, 0, 0.09, \dots 0] \rightarrow \{\text{PoleVaulteR}, 0.9\} \quad (20)$$

### 3.5.5 Complexity Issues

The graph path feature vector representation has the following complexity characteristics

- The number of features depends on the size of the vocabulary and the size of the graph neighborhood considered. Namely, the number of features is bounded by  $(|\mathcal{V}|^n \cdot |\mathcal{E}|^n)$  where
  - $|\mathcal{V}|$  is the size of vocabulary for vertices, i.e. the number of possible concepts,
  - $|\mathcal{E}|$  is the size of vocabulary for edges, i.e. the number of possible relations and
  - $n$  is the longest path considered.

Though this bound is not polynomial in respect to  $n$ , the actual existing distinct paths in graphs of real-world problem will tend to be of a much smaller num-

ber. Moreover, as experiments have shown us, very good performances may be attained for  $n < 4$ .

- The complexity of evaluating each feature is bounded by  $O(n \cdot \bar{E})$ , where  $E$  is the maximum degree of any vertex in the graph. Again, the typical situation in a real world graph is that most vertices are connected to few others while few vertices are connected to many. Therefore, the average complexity for evaluating each feature should typically be much less than its bound.

Based on the above, the overall worst complexity for evaluating the feature vector is  $O(n \cdot |\mathcal{V}|^n \cdot |\mathcal{E}|^n \cdot \bar{E})$ . The average complexity though is expected to be much smaller. In any case, though, for a fixed  $n$ , the complexity remains polynomial.

What's more  $n$  serves as a complexity/accuracy control parameter: For large  $n$  value, one obtains highly accurate feature vector representation. For smaller  $n$  values, the representation is less accurate, though much quicker to learn and apply. Typically, a value of  $n = 3$  has been adequate for all problems we have evaluated the algorithm.

### 3.5.6 Soft Classifiers as scoring functions

Having specified how feature vectors associated to a vertex are generated, we now return back to the question of defining the scoring function. Remember that the scoring function should evaluate the optimality of adding an edge and/or a vertex to the graph into a particular point of the graph. In what follows, we concentrate on the elementary operator regarding a vertex, though the same arguments hold for edges.

Namely, in a somehow different way, the same problem may be stated as finding the probability that the vertex  $v_{new}$  is inserted to the graph with any of value  $c \in \mathcal{V}$ . Based on the graph path feature vector representation, the problem specializes further as follows

Given the graph path feature vector representation around a point of the graph, find the optimal value of vertex.

This is a typical case of soft classification problem in machine learning literature, where

- the input is the feature vector representation around the vertex to be inserted
- the output is (a) the optimal value of the vertex to be inserted and (b) a confidence score for attributing the particular value.

The following remarks apply:

- The confidence score is not necessarily a probability value of the concept class in respect to the others. However, methods that can adapt scores found by classifiers into probability measures exist and have been applied in the literature.
- The confidence scores of classifiers are more accurately extracted as probability values of generative models for the optimal value found by classification. In this way, the score of the winning class for one class may still be low, even if has a high score in respect to all other competing values.

Soft classifiers may thus be used as scoring functions for elementary graph expansion operators. This has the significant advantage of delegating the adaptation of scoring functions for particular cases of ontologies to standard machine learning algorithms that use the feature vector representation. K-Nearest Neighbors, Support Vector Machines and Artificial Neural Networks are some of the approaches one may choose to adopt. Each one of these methods has a particular mathematical model to implement the scoring function and a specific way to learn the scoring function from training material, consisting of a set of input - correct output samples of form similar to eq. 20.

Since the scoring function approximates part of the “reasoning”, the choice of the classifier may affect the accuracy of the overall approach. Typically, classifiers with complex mathematical models will be more accurate, though they will demand a large number of training samples as inputs. Therefore, accuracy will be obtained at the cost of higher training time and larger sizes of ground-truth data to be provided.

## 4 Evaluation

In this section, we present the evaluation results of the method presented in this chapter. In section 4.1.1, the data used for evaluation is described while in section 4.1.2 the evaluation procedure is explained. In section 4.2 we present the results obtained for image, text and fused data, also showing the sensitivity of the method to the sub-graph size and classifier used respectively.

### 4.1 *Experimental Setting*

#### 4.1.1 Data

The proposed method has been evaluated on a corpus of approximately 600 web pages containing text and image from the BOEMIE athletics experiment [10]. The assertion boxes resulting from per-media analysis tools have been used as the *input*. The output (ground truth) has been formed using the BOEMIE inference engine, adapted with hand-written DL-axioms, DL-safe rules and abduction rules [9]. The fact that output may be somehow different than the ones produced manually is not significant in respect to our goal, which is to evaluate the way of the proposed approach to learn how to learn to infer based on a given reference set.

We have conducted separate experiments to test the suitability of the method for assertions sets related to image, text as well as the fused web pages content. Depending on the medium, the set of values vertices and edges to be predicted, as well as the feature vector space differ. Notice that the classes to be predicted are concepts and relations that exist only in the output assertions sets and not in the respective input ones. These correspond to the high-level concepts, the relations

# Class Set	
Image Vertices	7 HighJump, HighJumper, JavelinThrow, JavelinThrower, Person, PoleVault, PoleVaulter
Image Edges	16 contains, hasPart, hasParticipant, isAbove, isAboveLeft, isAboveRight, isAtBothSides, isBehind, isBelow, isBelowLeft, isBelowRight, isLeft, isNear, isOverlapping, isRight,
Text Vertices	17 DiscusThrowCompetition, HammerThrowCompetition, HighJumpCompetition, Hurdling110mCompetition, JavelinThrowCompetition, LongJumpCompetition, MarathonCompetition, Person, PoleVaultCompetition, Running100mCompetition, SportsCompetition, SportsEvent, SportsRound, SportsTrial, Walking10kmCompetition, Walking20kmCompetition, Walking50kmCompetition,
Text Edges	25 contains, hasAge, hasGender, hasNationality, hasPart, hasParticipant, hasPerformance, hasPersonName, hasRanking, hasSportsEventName, hasSportsName, hasSportsRoundName, hasStartDate, personToPerformance, personToRanking, sportsCompetitionToSportsRoundName, sportsCompetitionhasPartRound, sportsEventToSportsName, sportsRoundToPerformance, sportsRoundToRanking, sportsRoundhasPartSportsTrial, takesplaceInCity, takesplaceInCountry, takesplaceInSportsPOI

**Table 1** The class set used for evaluating the assertion boxes extracted from image and text. The class sets corresponding to the assertion boxes extracted from web pages are the union of the respective one for text and image.

between high-level concepts and the relations between high-level concepts and mid-level concepts. The set of these concepts is given in Table 1.

The features generated from the input data depend both on the medium and on the size of the subgraphs considered. Table 2 shows an indicative set of the features used for some cases. The paths are formed using concepts and relation that are part of both the input and the output assertions. Notice that the size of the feature set grows as the longest paths in the graph grow. For instance, the feature set to predict image vertices grows from 17 to 118 features. However, the number of features may be much less than the combinations of values, as is the case for the text vertices related features, where the feature set grow from 30 to 105. This happens because the path considered are not all the possible combinations, but only those that occur in the reference material.

#### 4.1.2 Methodology

The evaluation aimed at measuring the generalization accuracy of the method. To this end, the model was trained with a part of the reference material and tested with another part of the reference material. For image-related assertions, a 10-fold cross-validation test has been conducted. For text and web-page related assertions, which number is significantly larger, a small part of the reference material (1% - 5%) has been used for training, whereas the other part has been used for testing.

	path size	#	Indicative Feature Set
Image Vertices	1	18	hasPartS, hasParticipantO, hasParticipantS, isAboveLeftS, isAboveRightS, isAboveS, isAtBothSidesS, isBehindS, isBelowLeftS, isBelowRightS, isBelowS, isLeftS,
	2	112	hasPartS_HorizontalBar, hasParticipantS_PoleVault, isAboveS_Hammer, isAtBothSidesS, isBelowLeftS_Discus, isBelowRightS_HorizontalBar, isBelowS_Javelin, isLeftS_PersonBody, isNearS_PersonFace, isOverlappingS_Pillar
Text Vertices	1	30	hasAgeS, hasGenderS, hasNationalityS, hasPartOhasPartS, hasParticipantO, hasParticipantS, hasPerformanceS, hasPersonNameS, hasRankingS, hasSportsNameS, hasStartDateS, personToRankingS, takesplaceInCityS
	2	105	hasGenderS_Female, hasPartS_DiscusThrowCompetition, hasParticipantO_SportsTrial, hasSportsEventNameS_SportsEventName, hasSportsNameS_Running100mName, hasSportsNameS_Walking50kmName, hasSportsRoundNameS_SportsRoundName, personToPerformanceS_Performance, sportsCompetitionhasPartRoundS, takesplaceInCountryS_Country, takesplaceInSportsPOIS_Stadium

**Table 2** Indicative features used for evaluating the assertion boxes extracted from image and text

To measure the success of the method, one has to compare the ground truth assertion A-boxes (reference A-boxes) against the ones predicted by the method. To that end, we apply the following steps:

- find the optimal correspondence between each assertion of the reference A-box with each assertions of the predicted A-box
- measure the *recall* performance, by counting the number of vertices and edges of the *reference* A-box that have been matched by value with vertices and edges of the *predicted* A-box, in respect to the total number of vertices of the reference A-box.
- measure the *precision* performance, by counting the number of vertices and edges of the *predicted* A-box that have been matched by value with vertices and edges of the *reference* A-box, in respect to the total number of vertices of the predicted A-box.

Once the alignment has been established, the steps for counting the recall and precision are straightforward. On the other hand, the optimal alignment employs a more elaborate procedure.

**Optimal graph alignment** Namely, an optimal alignment could be possible by exhaustively searching all possible alignments and select the one that offers maximum performance in respect to the evaluated measures. Nevertheless, this approach is of prohibited complexity. Instead, we propose here a greedy approach to iteratively align each assertion of the reference a-box with an assertion of the predicted a-box.

The steps are the following:

- Step 0 Represent the two assertions sets as two directed graphs, using the approach that has been described in this chapter
- Step 1 Represent each edge of both graphs with the feature vector extracted using the approach described in this chapter, i.e. through the paths that connect the edge to its graph neighborhood. The neighborhood size may be controlled to balance the accuracy vs complexity of the alignment. Indicatively, a path length of 3 should be adequate for most cases.
- Step 2 For each edge of the first graph, find the closest edge of the second graph. Distance between edges is measured as follows:
- if the identifiers of adjacent vertices are identical, the distance is 0, else
  - the distance between edges is measure as the  $L_2$  distance of their respective feature vectors.
- Step 3 Match the pair of edges that have the smaller distance and remove them from the set of edges to be examined.
- Step 4 Go back to Step 2 to find the closest edges among those that are not yet matched, until all edges of either of the graph are matched
- Step 5 Match the vertices of the two graphs based on matching of the edges.

## 4.2 Evaluation results

This set of experiments focus on evaluating the feature extraction approach and measure its sensitivity to the graph path length used and the machine learning classification algorithm employed. Namely, Table 3 summarizes the performance achieved for the image modality. The results corresponds to hiding one vertex of the graph to be predicted and try to recover it using the proposed algorithm. The first column specifies whether the task was to predict the value of a vertex or of an edge. The second column specifies the maximum graph path considered. The rest of the columns give the accuracy obtained using the Naive Bayes, Logit Boost, K Nearest Neighbors and SVM classifier respectively. Notice that for the text and html edges, the graph path length examined was up to 1, since the values were satisfactory for that value.

**Sensitivity to the class value** Table 4 show that performance varies with the particular class to predict. Note that these correspond to using the Naive Bayes and thus are not the optimal one. Rather they are given to illustrated the sensitivity of methods to each class. Similar conclusion may be drawn by looking to the confusion matrix depicted in Table 5. Notice, for instance that though the relation `hasParticipant` is always predicted with maxim accuracy, the relation `isLeft` is predicted with a not satisfactory level of accuracy. The study of this table could lead to several optimizations, including the addition of supplemental material regarding these relation, to enhance the training process.

Case study	path	Naive Bayes	LogitBoost	K-NN	SVM
image vertices	1	63.75	64.54	58.57	61.35
	2	<b>99.80</b>	99.80	93.82	99.80
image edges	1	60.75	66.68	66.61	66.64
	2	70.04	<b>94.18</b>	89.80	92.62
text vertices	1	92.74	93.78	<b>93.78</b>	93.32
	2	94.76	<b>96.63</b>	95.01	95.76
text edges	1	95.48	96.67	95.12	<b>96.67</b>
html vertices	1	82.35	82.86	80.95	72.51
	2	89.11	<b>91.25</b>	90.08	89.88
html edges	1	96.37	97.03	97.36	<b>97.69</b>

**Table 3** Evaluation results for the image and text-related data

relation	precision	recall	f-measure	roc-area
hasPart	0.996	0.635	0.776	0.962
hasParticipant	1	1	1	1
isAbove	0.463	0.503	0.482	0.88
isAboveLeft	0.173	0.31	0.222	0.916
isAboveRight	0.353	0.526	0.423	0.918
isAtBothSides	0.091	0.2	0.125	0.84
isBehind	0.41	0.762	0.533	0.916
isBelow	0.601	0.688	0.642	0.889
isBelowLeft	0.135	0.28	0.182	0.936
isBelowRight	0.207	0.545	0.3	0.903
isLeft	0.218	0.267	0.24	0.825
isNear	0.178	0.31	0.226	0.929
isOverlapping	0.451	0.36	0.4	0.811
isRight	0.294	0.2	0.238	0.831

**Table 4** Analysis of the Naive Bayes results for classifying edges in image assertions.

**Sensitivity to the Path Length** By looking at Table 3, one can notice that the method accuracy depends on the graph path length. Clearly, for graph path length equal to 1, i.e. considering only direct neighbors, the accuracy is not satisfactory. However, by considering a graph path length equal 2, a very good performance is obtained. This is particularly noticeable for predicting image edges. Since the results have been satisfactory for graph length equal to 2, there has been no need to experiment with longest graph paths. Nevertheless, the conclusion that graph length up to 2 are adequate applies only to the particular case we have studied.

**Sensitivity to the Classification Method** By looking at Table 3, one sees that results depend on the soft classification method used. This is a typical situation in the machine learning, since the performance of the underlying classification model and learning algorithm may depend on the domain of application. Moreover, one may

choose, depending on the complexity requirements, to employ a “cheap” classifier, such as the Naive Bayes, or a more complicated one

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	- classified as
500	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	a = contains
0	546	0	16	6	21	9	104	59	13	21	15	35	6	9	0	b = hasPart
0	0	186	0	0	0	0	0	0	0	0	0	0	0	0	0	c = hasParticipant
0	0	0	81	11	6	9	18	7	2	2	3	6	16	0	0	d = isAbove
0	0	0	8	9	0	1	0	1	3	0	5	1	1	0	0	e = isAboveLeft
0	0	0	4	2	30	3	7	2	0	0	1	0	3	5	0	f = isAboveRight
0	1	0	7	5	5	6	1	0	2	0	1	1	1	0	0	g = isAtBothSides
0	0	0	2	0	3	1	112	4	0	11	2	1	9	2	0	h = isBehind
0	0	0	21	3	1	6	13	170	8	2	4	7	11	1	0	i = isBelow
0	1	0	0	1	0	1	3	4	7	0	1	0	7	0	0	j = isBelowLeft
0	0	0	1	0	1	2	0	0	1	12	0	0	4	1	0	k = isBelowRight
0	0	0	8	5	0	1	2	6	4	1	12	1	5	0	0	l = isLeft
0	0	0	6	0	0	0	0	7	0	2	2	13	12	0	0	m = isNear
0	0	0	16	8	10	13	5	23	12	4	9	8	64	6	0	n = isOverlapping
0	0	0	5	1	7	14	8	0	0	2	0	0	3	10	0	o = isRight

**Table 5** Confusion matrix for the results of the Naive-Bayes classifier for edge classification task using graph path length 1

## 5 Related Work

Extending strict reasoning approaches so that they are amenable to learnability and uncertainty handling has been the focus of much recent research.

Approaches such as fuzzy description logics [13] or [6] are a direct attempt to extend description logics so as to allow for uncertainty handling. However, the complexity of such approaches is prohibitive for the large scale of multimedia data we are dealing here.

On the other hand, there has been important work on the emerging field of statistical relational learning, such as probabilistic relational models [3], relational dependency networks [8], markov logic [11] and logical hidden markov models [4]. Although allowing both learning and uncertainty handling, these approaches differ to ours in two ways. First, they all considering specific network-like models which connections weights among nodes reflect the degree of association between the corresponding concepts. In contrast, the approach presented here is blind in this respect, since modeling is delegated to a pluggable soft classifier (such as LogitBoost), and hence not necessarily a network-like structure. Second, our problem formulation allows to expand the A-box in an arbitrary way, given that representative training

material has been provided. This allows, for example, adding individuals not existing at the input, such as those inferable by abductive reasoning [9],

## 6 Conclusions

In this chapter, we have presented a complete methodology that allows for extracting semantic knowledge from multimedia data sources. We have first presented that steps allow to express multimedia information by means of semantic assertions.

We have then described a machine learning method that allows to extract complex semantics, through performing inexact inference in ontologies. The method is based on transforming the inference problem into a graph expansion problem, expressing graph expansion operators as a combination of elementary ones and optimally seeking elementary graph operators. The latter issue is reduced to learn a set of soft classifiers, based on features each one corresponding to a unique graph path.

Our method has been evaluated in an web page corpus comprising images and text. A specific evaluation procedure has been devised to evaluate the corpus. The evaluation results show that predicting the concepts and relations that structure together multimedia data is possible by inexact inference. Since the method is completely learnable, its adaptability to domains other than the one presented here is an issue of further investigation by the authors.

**Acknowledgment** The study presented in this chapter has been partly supported by the research projects “BOEMIE, Bootstrapping Ontology Evolution with Multimedia Information Extraction”. FP6-027538 / STREP, 2006-2009, <http://www.boemie.org> and “CASAM, Computer-Aided Semantic Annotation of Multimedia”. ICT-217061 / STREP, 2008-, <http://www.casam-project.eu/>

## References

1. P. Aarabi and B.V. Dasarathy. Robust speech processing using multi-sensor multi-source information fusion—an overview of the state of the art. *Information Fusion*, 5(2):77–80, 2004.
2. A. Ardeshir Goshtasby and S. Nikolov. Image fusion: Advances in the state of the art. *Information Fusion*, 8(2):114–118, 2007.
3. N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 1300–1309. Citeseer, 1999.
4. K. Kersting, L. De Raedt, and T. Raiko. Logical hidden markov models. *Journal of Artificial Intelligence Research*, 25(1):425–456, 2006.
5. B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International joint conference on artificial intelligence*, volume 3, pages 674–679. Citeseer, 1981.
6. T. Lukasiewicz. Probabilistic description logic programs. *International Journal of Approximate Reasoning*, 45(2):288–307, 2007.

7. A.V. Nefian, L. Liang, X. Pi, X. Liu, and K. Murphy. Dynamic Bayesian Networks for Audio-Visual Speech Recognition. *EURASIP Journal on Applied Signal Processing*, 2002(11):1274–1288, 2002.
8. J. Neville and D. Jensen. Relational dependency networks. *The Journal of Machine Learning Research*, 8:692, 2007.
9. S.E. Peraldi, A. Kaya, S. Melzer, R. Moller, and M. Wessel. Multimedia interpretation as abduction. In *Proc. DL-2007: International Workshop on Description Logics*, 2007.
10. S. Petridis and N. Tsapatsoulis. Semantics Extraction from Multimedia Content: The BOEMIE Architecture. In *Proceeding of the first international conference on Semantics and digital Media Technology (SAMT 2006)*, pages 6–8, 2006.
11. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1):107–136, 2006.
12. S. Rudolph, T. Tserendorj, and P. Hitzler. What Is Approximate Reasoning?? In *Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems*, pages 150–164. Springer, 2008.
13. U. Straccia. Reasoning within Fuzzy Description Logics. *Journal of Artificial Intelligence Research*, 14:137–166, 2001.
14. E. Zavitsanos, G. Paliouras, G.A. Vouros, and S. Petridis. Discovering subsumption hierarchies of ontology concepts from text corpora. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 402–408. IEEE Computer Society Washington, DC, USA, 2007.



# Index

complementarity, *see* multimedia data

fusion

- high level, 4
- low level, 4
- mid level, 4

graph

- directed, 13
- graph expansion, 15
- graph path, 19
- greedy search, 15

high level fusion, *see* fusion

low level fusion, *see* fusion

mid level fusion, *see* fusion

multimedia data

- complementarity, 5
- logical structure, 7
- redundancy, 6

redundancy, *see* multimedia data

semantics

- approximate inference, 12
- assertion box, 11
- ontology, 10
- soft classification, 21