An OWL framework for rule-based recognition of places in Italian non-structured text

Domenico Cantone, Andrea Fornaia, Marianna Nicolosi-Asmundo,
Daniele Francesco Santamaria, Emiliano Tramontana

University of Catania, Dept. of Mathematics and Computer Science
email: {cantone,fornaia,nicolosi,santamaria,tramontana}@dmi.unict.it

Abstract. In this paper we present a technique for recognising location names in non-structured texts. Our approach is based on grammar rules devised for the Italian language and semantic web tools such as geographic linked datasets, *OWL* ontologies, and *SWRL* rules, to handle data and reason about them, even in presence of name ambiguities. To the best of our knowledge, this is the first attempt of addressing the problem of location recognition in the context of Italian texts with such an ontological support.

1 Introduction

Recognising location names of geographical places and of public or private buildings inside non-structured text documents is an important issue with several practical applications. For instance, in the investigative field it is important to reveal a place named in the transcription of an interception (i.e., by means of wiretapping) and in the social media context to reveal the places visited by users to provide targeted advertisements. This problem falls both in the category of *information extraction* from unlabeled texts and of *named entity recognition*, where most of the existing approaches are focused on the English language and are in general not applicable to the Italian language, or they lead to unsatisfactory results due to the peculiarities of the language.

The problem of location name recognition has been addressed in various ways [24], e.g., using maximum entropy models [23], or with Conditional Random Fields [21], or with automatic learning techniques to infer the rules for the named entities identification inside free texts [14], or, in the last decade, also with linked data and ontologies [20] (a survey of the main geographical ontologies and datasets can be found in [15]). However, many of such approaches have been tested, or developed, vertically on top of the English language, making them hard to generalize to the Italian language. In addition, in some our preliminary experiment, widespread applications such as *Stanbol* [1] turned out to be unsatisfactory from the point of view of success ratio when applied to Italian places drawn from non-structured text written in Italian.

In this paper we focus on the problem of recognising names of geographical places in the Italian country, which appear in non-structured Italian text documents. Our approach consists in extracting location names from Italian texts

according to an extended version of the algorithm presented in [18], and then storing data and making inferences, even in presence of name ambiguities, with semantic web tools such as geographic linked datasets, *OWL* ontologies [4], and *SWRL* rules [11].

We recall that the algorithm presented in [18] relies on a set of three finite state machines, each designed to recognise several sentence patterns for the Italian language, in which location names are typically found. Such an analysis takes as input a non-structured text written in Italian and yields as output a HTML text, where candidate location names have been automatically marked by a label.

In this contribution, we unified the finite state machines presented in [18] in a single automaton, reducing the overall number of states involved during the extraction of location names from the text; each location detected by the algorithm is then searched in the *OpenStreetMap* dataset [5]. Such a search yields a list of possible matches of real places, each with its degree of reliability. Then, the data retrieved by *OpenStreetMap* are inserted in a novel ontology, namely *OntoLocEstimation*, to handle ambiguous geographical names. *OntoLocEstimation* uses the ontology *OntoLuoghi*, introduced in [17], that contains a detailed description of the administrative model of Italian places.

The use of open datasets, such as *OpenStreetMap*, allows a widespread and detailed coverage of Italian geographical places and provides a high precision in the detection of real places. In addition, the introduction of Semantic Web Rule Language (SWRL) rules [11] allows inferences on knowledge, implicitly contained in the novel ontologies which are more refined than other currently available ontologies on geographical places.

2 Background

The *semantic web* is a vision of the World Wide Web in which information carries an explicit meaning, so it can be automatically processed and integrated by machines, and data can be accessed and modified at a global level, resulting in increased coherence and dissemination of knowledge. Moreover, thanks to suitable procedures of automated reasoning, it is possible to extract implicit information present in data, leveraging a deeper knowledge of the domain. The domain is specified by means of expressions describing statements about web shared resources. Such expressions are given as triples of the form *subject-predicate-object*. The *subject* denotes the resource to describe, namely the actor of the statement, the *object* denotes the recipient or the result of the action, and the *predicate* denotes traits or aspects of the resource, i.e., a relationship between the subject and the object. For instance, if we wanted to express the concept "Michelangelo made the Sistine Chapel ceiling", we would write <#Michelangelo><#makes><#Sistine_Chapel_Ceiling>, where "#" indicates that the local context is used.[1] Every time we want to add some information about a

[1] For space reasons, we express triples using a simplified version of the Turtle notation [13].

specific resource, we reuse the resource as subject or object of a new triple, depending on the type of relationship. For example, if we additionally wanted to express the creation date of the Sistine Chapel ceiling, we would write <#Sistine_Chapel_Ceiling><#hasCreationDate>"1512".

Sharing data is an important feature of the semantic web. Because of the uniqueness of the URI of the resources and of the document, we can refer to them without ambiguity. For instance, assuming that the URL `http://wwww.unict.it/art.rdf` refers to the document containing information about "Michelangelo", every time we need to state something about him, we use the document `http://wwww.unict.it/art.rdf`, and the resource `http://wwww.unict.it/art.rdf#Michelangelo` as a resource in our statements.

Semantic web data are usually published using the Resource Description Framework (RDF) [9] (or its extension RDF Schema (RDFS) [10]) and the Ontology Web Language (OWL) [4]. An important feature of the semantic web is the capability to extract implicit information from the described data. This is why the language RDFS has been introduced. Unlike the RDF language, RDFS is powerful enough to enable such a feature since it allows to express subclass and subproperty relationships. Elements of the domain sharing common characteristics can be grouped in particular sets called *classes*. Classes, in their turn, can be organised in hierarchies. A hierarchy of classes is called a *taxonomy*. Properties can be organised in hierarchies too. For instance, the relation "has father" can be modelled as a subproperty of the relation"has ancestor". If an element of the domain is related to another element by a subproperty, then it is related to the superproperty as well. RDFS provides other interesting inference capabilities that, for space reasons, we do not report here. However, RDFS is far away from allowing complex reasoning. For instance, let us consider the relation "uncle" in a domain of persons. If we express that "Marta" is daughter of "Frank", and "Frank" is brother of "John", we are making no assumption on the relationship between "Marta" and "John". However, in the knowledge domain we have to model we might be aware of the fact that "the brother of my father is my uncle" as a result of the combination of sibling-daughter properties. Relationships of this type can be expressed in OWL using a multitude of properties.

The task of writing OWL data requires the definition of an ontology. Informally, in computer science an *ontology* defines a set of representational primitives (classes and properties) apt to model a domain of knowledge or of discourse [22]. When an OWL reasoner (namely, a given software system able to extract information from OWL files) is executed on OWL data, we can perform the task of mining data from resources. Although an RDFS schema can be potentially converted into an OWL ontology (such a task is known as *RDF alignment* [6,7]), this is not sufficient to turn an RDF document into an OWL ontology. RDF and OWL partially share the syntax but not the semantics. In fact, OWL allows one to specify far more about the properties and classes of an RDFS schema by means of a formal description of the data. Expressiveness of such ontological model depends on the OWL profile adopted. Profiles are fragments of the language that trade off some expressive power for the efficiency of reasoning,

introduced in order to deal with several types of application domains. Some of these profiles include SWRL rules. Such rules have the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must hold as well. The desirable features of the OWL language shortly outlined above strongly motivate our interest in using OWL ontologies and related reasoning tools for the location recognition task. More details about OWL reasoning capabilities, semantics, and profiles can be found in [4, 7, 11]. In particular, in [16], a reasoning system based on a fragment of set-theory is proposed particularly suitable for the ontologies presented in this paper.

3 Rule-Based Location Extraction

In this section we briefly illustrate the approach we used to extract geographical and administrative places from Italian non-structured text. Unlike machine learning approaches, both unsupervised and supervised, we proposed a rule-based approach built from simple grammar rules of the Italian language complemented by a dictionary, where each rule identifies a different pattern that characterises sentences at the end of which we usually find a location name. The devised rules are supported by a specifically compiled Italian lexicon, containing the classes of words *Articles*, *Verbs*, and *Descriptors*, and a list of *Non-places* words that are known false positives; in order to improve the overall accuracy of the extraction tool, the lexicon can also be extended by other, user-detected false positives [18]. As shown in [18], these rules provide a large coverage of the Italian grammar for what concerns statements about places.

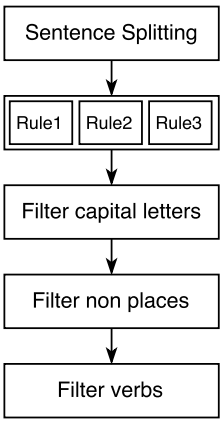

Fig. 1. Pipe and filter workflow for location extraction.

Figure 1 depicts the entire workflow used for the location extraction: in the first *sentence splitting* step, an input text T is separated into a list of sentences using occurrences of punctuation marks, i.e., *full-stop*, *ellipsis*, *exclamation-mark*, *question-mark*. Any other non-letter symbol is ignored, e.g., dollar sign, percent sign, etc. Each sentence is further segmented into words using the space character as a separator. The tokens (words) are then fed to a finite state machine implementing three different *rules*, i.e., grammar cases possibly implying the use of a place name at the end of a sentence: if the accepting state (6) of the automaton is reached, the current token is marked as a location candidate. The result is a list of candidate words that must pass through a configurable sequence of *filters* before being actually labelled as a place name.

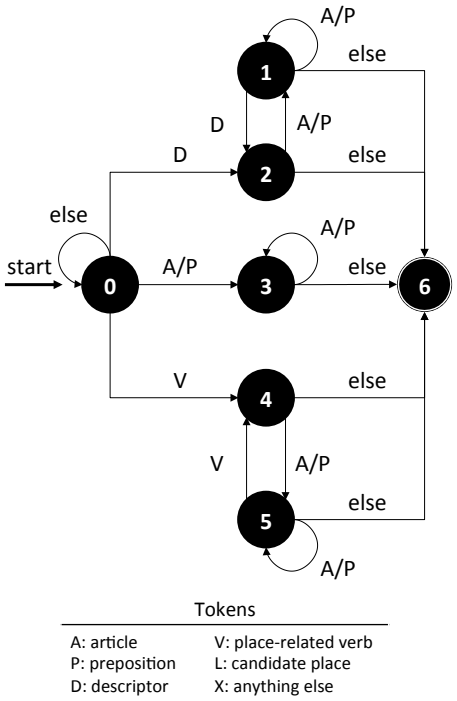

Fig. 2. Unified finite state machine of the three Italian grammar rules in [18].

Figure 2 shows the unified automaton used in this contribution to implement the three grammar rules defined in [18]; those rules were devised to accommodate the mentioning of a place name in sentences such as the rule names suggest, namely, *Da Roma* (from Rome), *Vicino a Roma* (Near Rome) and *Andando a Roma* (Going to Rome). To give a simple detection example, Figure 3 shows an excerpt of the unified automaton giving a finite state machine expressing only the first rule, (*Da Roma*), whose name recalls the grammar pattern responsible for

the matching. The automaton scans the tokens of a given sentence and remains in state 0 until a preposition (P) or an article (A) is found, at which point the automaton changes its internal state to 1. Subsequently, the automaton remains in state 1 until a different kind of word is encountered, in which case the final state is reached and a new candidate word is found. However, several candidates will be actually dropped afterwards by the filters, e.g., known false positives or conjugated verbs.

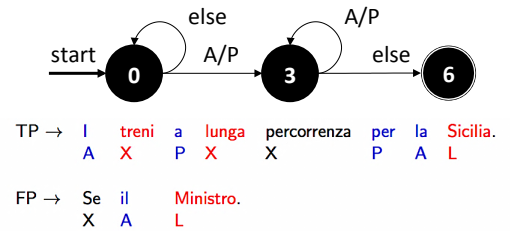

Fig. 3. A finite state machine implementing only the first rule *Da Roma* (from Rome) in [18], with application examples.

All the candidates found by the automaton are then given as input to a sequence of filters in order to remove trivial false positives that may have been selected:

- Filter0: the candidate for a location name must begin with a capital letter; even if it may considerably improve the detection accuracy in several cases, this is not a mandatory filter. In fact it is applied only when we can assume that location names are written with a leading capital letter (e.g., if we are analysing newspaper articles);
- Filter1: remove all known false positives using the devised lexicon of non-places;
- Filter2: remove conjugated verbs.

Any word surviving the above filters is labelled as a place name, and is given to the ontologic support to store the results and automatically retrieve the information concerning the algorithm and dataset used, and the spotted place.

4 An ontology for reasoning with places

In this section we first describe the ontology *OntoLocEstimation*, developed with the purpose of reasoning with geographical and administrative places. Then we show how the ontology is populated by means of a Java framework. *OntoLocEstimation* is associated to the algorithm introduced in Section 3 and it allows us to manage the identification of a specific location even in presence of uncertainty.

4.1 The ontological model

We illustrate how the ontology *OntoLocEstimation* is structured. *OntoLocEstimation* extends the ontology *OntoLuoghi* [17] with OWL constructs allowing us to deal with the administration of Italian places and with the algorithm described in Section 3. In its turn, *OntoLuoghi* reuses some concepts and properties of *LinkedGeoData* [3].

Locations are modelled by means of a taxonomy of OWL classes. Association among locations is performed through a taxonomy of object-properties. The path allowed, namely the hierarchy of such classes, is shown in Figure 5. Double-hoop entities in Figure 5 are considered as optional. Names of the OWL classes and object-properties involved are shown in Figure 4.[2] Reasoning capabilities concerning locations are strengthened using the SWRL rules shown in Figure 6.

The entity "Localisation" is equivalent to "LinkedGeoData:Place" (equivalences among subclasses of "Localisation" and of "Place" are not reported here for space reasons). The object-property "hasLocalisation" is defined as an OWL transitive property. Thus, if the pairs of objects (x, y) and (y, z) are in the property "hasLocalisation", then the pair (x, z) is included in the property "hasLocalisation" too. The property "hasLocalisation" can be used together with its subproperties to infer the administrative hierarchy of a location providing only the top level of a place (i.e., the superproperty). For instance, if we write the statements (Sicily hasState Italy), (Catania hasRegion Italy), (Acireale hasProvince Catania), then (Sicily hasLocalisation Italy), (Catania hasLocalisation Italy), (Acireale hasLocalisation Catania), (Acireale hasLocalisation Sicily), (Acireale hasLocalisation Italy) are inferred thanks to the subproperty relationships and to the transitivity of "hasLocalisation". In addition, the set of SWRL rules depicted in Figure 6 allows one to infer the statements (Catania hasState Italy), (Acireale hasRegion Sicilia), (Acireale hasState Italy). Moreover, subproperties are defined as functional properties so as to guarantee that places can not be associated to different locations. For instance, Acireale can not be associated to the province of Catania and Palermo at the same time.

Usually, an algorithm recognising a geographical location from a keyword in a non-structured text yields several candidates, each in combination with a *degree of belief*. The degree of belief indicates how much the association between the keyword and the candidate geographical place is considered reliable by the approach in use. The ontology should model this situation and also manage additional information such as the algorithm applied for extracting location names from the text and the dataset adopted. Such information is relevant if one wishes to compare the accuracy that different algorithms and datasets provide for the task of recognising location names. Our approach works as follows. Every time the algorithm finds a keyword, an instance of the class "TextKey" is added to the dataset. The class "LocEstimation" models the fact that the algorithm of Section 3 is executed on the keyword and some results are provided. The instances of "Textkey" and "LocEstimation" are related to each other by the

[2] Images in Figures 4, 7, 8, 9, and 10 are drawn from the interface of the editor Protégé [8].

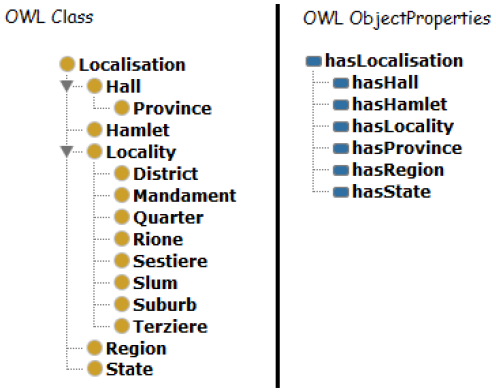

Fig. 4. Classes and properties for locations.

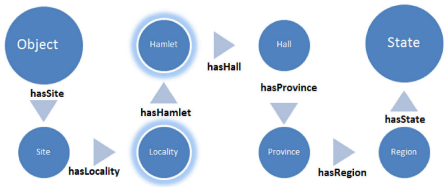

Fig. 5. Allowed path for locations.

OntoCeramic Rules
Locality(y), hasLocalisation(?x,?y) → hasLocality(?x,?y)
Region(y), hasLocalisation(?x,?y) → hasRegion(?x,?y)
State(y), hasLocalisation(?x,?y) → hasState(?x,?y)
Hamlet(y), hasLocalisation(?x,?y) → hasHamlet(?x,?y)
Hall(y), hasLocalisation(?x,?y) → hasHall(?x,?y)
Province(y), hasLocalisation(?x,?y) → hasProvince(?x,?y)

Fig. 6. SWRL rules for reasoning with locations.

object-property "hasLocEstimation". For every estimation there may be zero, one, or more matches. Each match of the algorithm is modelled by the "GuessedLocation" class. Instances of the class "GuessedLocation" are associated to the ones of the class "LocEstimation" by the object-property "hasGuessedLocation". Each instance of the class "GuessedLocation" provides information concerning the candidate geographical place and the relative degree of belief. The degree of belief is introduced by means of the data-property "hasGuessedValue" having the data-type double as range. The geographical place is specified with the object-property "hasReferredLoction" having as range every instance of the class "Localisation". Figure 7 illustrates classes and properties introduced in order to model the recognition of places, an example is shown in Figure 8.

The ontology provided takes also into account information concerning the algorithm for the extraction of the location and the dataset used. Classes and object-properties defined for this purpose are shown in Figure 9.

A class "Algorithm" provides information about the algorithm applied. Currently, we identify two types of algorithms, but others can be added. The class "DetectionAlgorithm" includes all the algorithms used to establish in a non-structured text whether a word represents a geographical place while the class

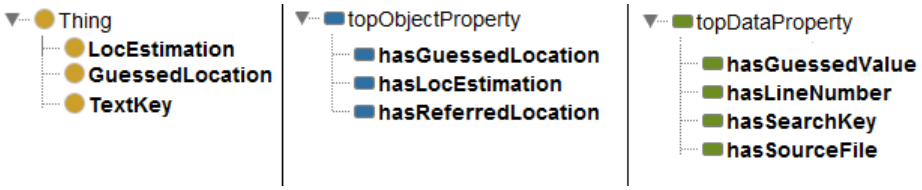

Fig. 7. Classes and properties for locations recognition.

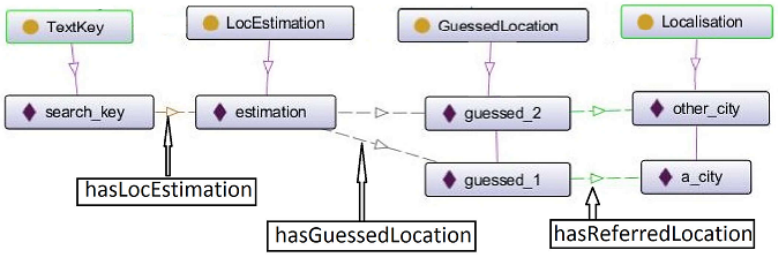

Fig. 8. Example of location estimation.

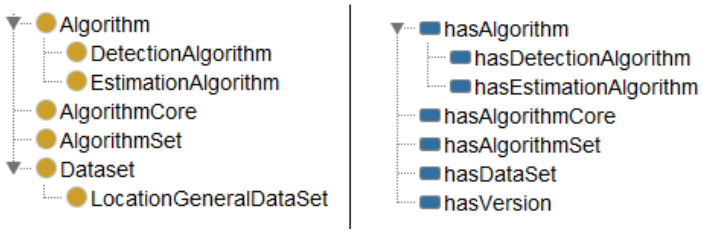

Fig. 9. Classes and object-property for algorithms.

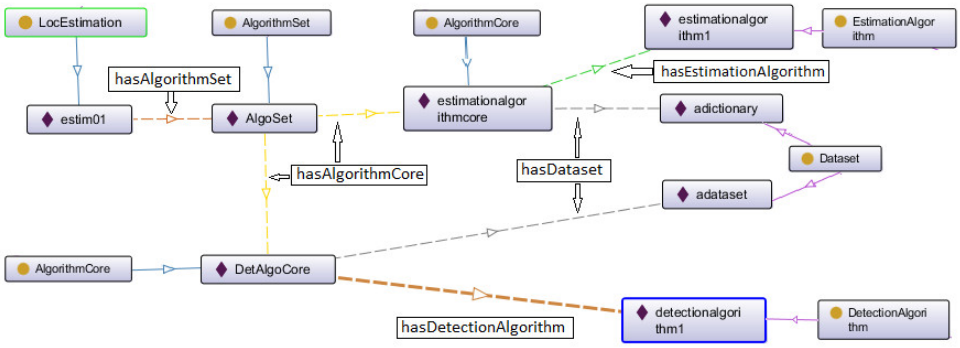

Fig. 10. Example of describing algorithms.

"EstimationAlgorithm" includes information about the algorithm used to choose a set of possible geographical places and to assign them a degree of belief. In addition, it is possible to specify the dataset used by the algorithm by means of the class "Dataset". The subclasses of the class "Dataset" describe the type of the dataset. For instance, the class "LocationGeneralDataset" is used to represent the datasets containing general geographical information.

In order to keep track of the algorithms used, we provide the class "AlgorithmSet" that is associated to an instance of the class "LocEstimation" by means of the object-property "hasAlgorithmSet". To each instance of the class "AlgorithmSet", one or more instances of the class "AlgorithmCore" are associated by means of the object-property "hasAlgorithmCore". The latter class relates an algorithm to the datasets used. In order to indicate the datasets used, the "hasDataSet" object-property is provided, having as range the "DataSet" class. Analogously, to associate an algorithm to an instance of the class "DataCore", the object-property "hasAlgorithm" is provided, having as range the class "Algorithm". In particular, two subproperties of the class "hasAlgorithm" are provided: the property "hasDetectionAlgorithm", that associates an instance of the class "AlgorithmCore" to an instance of the class "DetectionAlgorithm", and the property "hasEstimationAlgorithmCore" for the instances of the class "EstimationAlgorithm". How these classes and relations are used is shown in the example of Figure 10. Note that, by means of the class "AlgorithmCore", additional information about why, how, and when the algorithm set is used can be tracked.

4.2 Populating the ontology

We describe shortly how the ontology *OntoLocEstimation* is populated by means of a built ad-hoc Java framework. The first step of the process is to retrieve an open dataset of locations, that is as rich as possible, and to map such knowledge inside the ontology. We take into account the *OpenStreetMap* dataset and implement a Java parser in order to populate the ontology with the *OpenStreetMap* entries. The parser exploits the OWL API library [19] together with Jena Ontology API [2] that we adopted to perform SPARQL [12] queries. We used the reasoner Pellet [25] to carry out inferences on our ontology. As far as we know, Pellet provides the best deal between efficiency and data-type reasoning capabilities.

In a preliminary phase of this work we also considered the *LinkedGeoData* dataset that provides a semi-automated conversion of a subset of the *OpenStreetMap* dataset in RDF format. However, as outlined at the beginning of this paper, conversion of RDF in OWL is not straightforward. Thus, being mainly interested in OWL reasoning, we opted to provide the mapping of *OpenStreetMap* data in OWL statements by means of a simple parser.

5 Conclusions

We have presented an application that automatically recognises locations inside non-structured texts written in Italian and that is supported by an OWL ontology for locations management. The ontology also keeps track of the algorithm applied for detecting places and stores the degree of belief of each candidate location. Different algorithmic approaches can be stored in the ontology and, consequently, compared. Data of geographical locations are retrieved from an open dataset, i.e. *OpenStreetMap*, and adapted to the logical model of the ontology.

The results presented here can find applications in several contexts, such as the digital humanities field. The techniques illustrated can be adapted to recognise other kinds of elements inside non-structured text, such as descriptors of archaeological findings and finding places. This process simplifies the task of the automatic processing of archaeological archives, digitalisation and subsequent conversion in linked data format. In [17] we proposed an ontology for pottery classification, cataloguing, and reasoning that lends itself particularly well to such a task. This approach, combined with OWL reasoning capabilities, allows one to gain a deeper knowledge and better dissemination of the considered application domain.

We plan to introduce additional filters and rules in the algorithm so as to gain a larger coverage of the Italian grammar. In particular, we aim to make the algorithm sensitive to the contexts from which words are drown, namely non-structured texts. Finally, we plan to abandon external datasets, i.e. *OpenStreetMap*, in favour of internal built-in datasets that can be integrated by information provided by local governments or final users.

Acknowledgements

Work partially supported by the project *PRIME: Piattaforma di Reasoning Integrata, Multimedia, Esperta* within PO FESR Sicilia 2007/2013, and by the FIR projects *COMPACT: Computazione affidabile su testi firmati*, code D84C46, and *Organizzazione e trattamento di trascrizioni e testi in scenari di security*, code 375E90.

References

1. "Apache stanbol," https://stanbol.apache.org.
2. "Jena Ontology API," https://jena.apache.org.
3. "LinkedGeoData," http://linkedgeodata.org.
4. "Ontology Web Language," http://www.w3.org/2001/sw/wiki/OWL.
5. "OpenStreetMap," http://www.openstreetmap.org.
6. "OWL 2 Mapping to RDF," http://www.w3.org/TR/owl2-mapping-to-rdf/.
7. "OWL Semantics Mapping," http://www.w3.org/TR/owl-semantics/mapping.html.

8. "Protégé," http://webprotege.stanford.edu.
9. "The Resource Description Framework," http://www.w3.org/RDF/.
10. "The Resource Description Framework Schema," http://www.w3.org/TR/rdf-schema/.
11. "Semantic Web Rule Language," http://www.w3.org/Submission/SWRL/.
12. "SPARQL Query Language for RDF," https://www.w3.org/TR/rdf-sparql-query/.
13. "Turtle Notation," http://www.w3.org/TR/turtle/.
14. E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *Proceedings of the fifth ACM conference on Digital Libraries*. ACM, 2000, pp. 85–94.
15. A. Ballatore, D. C. Wilson, and M. Bertolotto, "A survey of volunteered open geo-knowledge bases in the semantic web," in *Quality issues in the management of web information*. Springer, 2013, pp. 93–120.
16. D. Cantone, C. Longo, M. Nicolosi-Asmundo, and D. F. Santamaria, "Web ontology representation and reasoning via fragments of set theory," in *Web Reasoning and Rule Systems - 9th International Conference, RR 2015, Berlin, Germany, August 4-5, 2015, Proceedings*, 2015, pp. 61–76.
17. D. Cantone, M. Nicolosi-Asmundo, D. F. Santamaria, and F. Trapani, "Onto-ceramic: an OWL ontology for ceramics classification," in *Proceedings of the 30th Italian Conference on Computational Logic, CILC 2015*, vol. 1459. Genova: CEUR Electronic Workshop Proceedings, July 1-3 2015, pp. 122–127.
18. D. Caruso, R. Giunta, D. Messina, G. Pappalardo, and E. Tramontana, "Rule-based location extraction from Italian unstructured text," in *Proceedings of XVI Workshop "From Object to Agents"(WOA)*, vol. 1382, July 17-19 2015, pp. 46–52.
19. M. Horridge and S. Bechhofer, "The OWL API: A Java API for OWL ontologies," *Semantic Web*, vol. 2, no. 1, pp. 11–21, 2011.
20. J. Kleb and R. Volz, "Ontology based entity disambiguation with natural language patterns," in *Fourth International Conference on Digital Information Management, 2009. ICDIM 2009.* IEEE, 2009, pp. 1–8.
21. J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, 2001, pp. 282–289.
22. L. Liu and M. T. Zsu, *Encyclopedia of database systems*. Springer Publishing Company, Incorporated, 2009.
23. K. Nigam, J. Lafferty, and A. McCallum, "Using maximum entropy for text classification," in *IJCAI-99 Workshop on Machine Learning for Information Filtering*, vol. 1, 1999, pp. 61–67.
24. S. Sarawagi, "Information extraction," *Foundations and trends in databases*, vol. 1, no. 3, pp. 261–377, 2008.
25. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, "Pellet: a practical OWL-DL reasoner," *Web Semantics: science, services and agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.