

# An Ontology-Driven Approach for Semantic Annotation of Documents with Specific Concepts

Céline Alec<sup>(✉)</sup>, Chantal Reynaud-Delaître, and Brigitte Safar

LRI, Univ. Paris-Sud, CNRS, Université Paris-Saclay, 91405 Orsay, France  
{celine.alec,chantal.reynaud,brigitte.safar}@lri.fr

**Abstract.** This paper deals with an ontology-driven approach for semantic annotation of documents from a corpus where each document describes an entity of a same domain. The goal is to annotate each document with concepts being too specific to be explicitly mentioned in texts. The only thing we know about the concepts is their labels, i.e., we have no semantic information about these concepts. Moreover, their characteristics in the texts are incomplete. We propose an ontology-based approach, named SAUPODOC, aiming to perform this particular annotation process by combining several approaches. Indeed, SAUPODOC relies on a domain ontology relative to the field under study, which has a pivotal role, on its population with property assertions coming from documents and external resources, and its enrichment with formal specific concept definitions. Experiments have been carried out in two application domains, showing the benefit of the approach compared to well-known classifiers.

**Keywords:** Ontology-driven approach · Ontology population · Ontology enrichment with specific concepts

## 1 Introduction

Nowadays, many Semantic Web applications use ontologies as rich conceptual schemas to give meanings to terms used as annotations of web contents. This paper deals with such an ontology-based approach addressing semantic annotation of documents when annotations are specific concepts. The only thing we know about these concepts is their labels. They have no definitions. We face three difficulties: (1) specific concepts are not explicitly mentioned in the documents under consideration, (2) specific concepts are not defined, even if the designer of the system knows their meaning and what kind of information is relevant to define them, (3) textual documents are incomplete, i.e., some characteristics describing these specific concepts are missing. For example, the concept “Destination where one can do Water sports during Winter” (DWW) is a very specific concept. Advertising descriptions of holiday destinations do not mention explicitly if a destination matches this concept. However, the designer is able to say whether a given textual description of a destination has to be annotated with this specific concept or not. Moreover, he knows DWW refers to a place

with the possibility of doing water sports and warm enough in winter, so that water sports are practicable. He also knows that terms referring to water sports can be found in documents but with no mention of the values of the winter temperatures. The missing information has to be searched in an external resource. In this paper, we show that an ontology can be enriched and then be used to annotate documents in such a constrained context.

The automation of this annotation process needs to formally define each specific concept, when it may be hard to do it. For instance, what is a place *warm enough* in winter? A solution is to automatically learn definitions provided that the designer is able to manually annotate documents as positive or negative examples for a given concept. Once a definition is learned, new descriptions of the same domain can be automatically annotated. Definitions are crucial in our work, which is currently used for a Business to Consumer application whose goal is to provide users with entities matching their needs in a particular domain. In that context, formal definitions can be used by reasoning to deliver partial satisfactory proposals when totally satisfactory ones do not exist. For example, if a user wants to go on vacation during winter and do water sports, entities annotated with DWW have to be proposed. If the DWW definition has lots of constraints, it is possible that no available destinations match this concept, implying no proposals to the user. This must be avoided. Close destinations have to be proposed, e.g., with a slightly lower temperature in winter.

This paper focuses on how an ontology can be populated and enriched in order to annotate documents. We investigate how several approaches can be combined in order to jointly contribute to address this annotation problem. Our contribution is then the SAUPODOC (Semantic Annotation Using Population of Ontology and Definitions of Classes) approach, relying on a domain ontology relative to the field under study, which has a pivotal role, on its population with property assertions, and on automatic generation of formal concept definitions from the populated ontology.

The remainder of the paper is organized as follows. Section 2 presents some related work. Section 3 describes the general aspect of our approach while Sect. 4 presents the various tasks involved. Section 5 presents experiments to evaluate the approach. Section 6 concludes and outlines future work.

## 2 Related Work

In this section, we review some existing literature about semantic annotation and highlight the need, in our context, for an ontology-based approach. Semantic annotation is a large area of the Semantic Web [20]. Annotating implies to attach data to some other pieces of data. Methods of semantic annotation of documents can be classified into two categories [23]: (1) pattern-based methods based on an initial set of entities and/or a set of patterns and (2) machine learning-based methods based on probability or induction. The idea is to look, in documents, for textual fragments that mention an entity. Named entity recognition [19] is part of this kind of annotation. However, our goal is a bit different.

We want to annotate a whole document, i.e., the entity described in it, not the elements mentioned in it. A few similar works deal with this idea. Their goal is to evaluate the proximity between a description of an entity and more specific elements (other documents, instances of concepts, concepts). [14] wants to match a job offer with candidates (CV, cover letters). Both are textual documents. Documents from both side are represented as vectors and various similarity measures are used to match them. [5] is to match hotel services. The hotel manager gives a description of the services of his hotel to be matched with a pre-existing list of services. The proximity is based on an n-gram calculation. Finally, [3] wants to annotate product catalogs with very fine-grained concepts. As similarity measures are not possible in this context, a first manual annotation is performed by an expert helped with an annotation tool. Then, machine learning techniques apply. Like these works, we want to match a description with specific concepts, i.e., to annotate descriptions with specific concepts. However, we face one more stake. We want comprehensive annotations. This means the process of annotation cannot be a black box. Indeed, when a concept asked by a user is not associated with any descriptions, we would like to make some refinements, i.e., to generalize its definition to be able to have some answers. This is what makes our work original.

Since the concepts used for annotation are not explicitly mentioned in the descriptions to be analyzed, a (at least) two-time process needs to be done. The first step is a classic extraction process while the second step is a reasoning one over the results of the first step. These two phases can be observed in concept-based approaches [8], such as concept-level sentiment analysis [7], which focuses also on text analysis relying on features considered as implicit because not represented in the texts but reasoning is here very specific. To the best of our knowledge, two works with a research purpose close to ours follow this idea. Both use ontologies. In the BOEMIE system [21], concepts from an ontology are divided into primitive and composite concepts. The first ones are populated via classical information extraction techniques. For the composite concepts, it is not that easy since they cannot be found in texts. However, their properties can be found. In this way, they can be defined in terms of primitive concepts. Composite concepts are populated via a reasoning performed on primitive instances. [27] aims to extract facts from texts using an ontology and natural language processing tools. New facts, which are not explicitly mentioned in texts, are then learned from the extracted facts and ontology knowledge. This learning is done via inference rules, written manually, relying on background knowledge. Compared to [21, 27], we do not have a manual definition of our concepts. These definitions need to be learned. This is one of our contributions.

### 3 The Saupodoc Approach: An Ontology-Based Approach

In this section, we present the general idea of the approach. The main point is the use of a domain ontology progressively populated with information extracted from documents under consideration and from external resources. Then, concept

definitions are learned based on this enhanced ontology and on some documents manually annotated. The last step is a reasoning one, where definitions apply in order to generate annotations of new documents. The approach, with the tasks as defined in the paper, is automatic and domain-independent. For example, it has been used to annotate destination descriptions as well as film descriptions.

### 3.1 Inputs of the Approach

Each domain needs its own inputs: (1) a domain ontology; (2) the list of concepts used to annotate, called target concepts; (3) a corpus of documents from which some of them have to be manually annotated as positive or negative examples for each target concept; (4) a specification of correspondences between the ontology properties and properties of external resources.

**The Ontology.** It defines the domain. It is a guide to analyze documents, to search for missing information in external resources and to reason with definitions. It contains all elements defining entities in the application domain. It is approach-independent. Indeed, the only constraints imposed by the approach are described in this section. The ontology can therefore be largely reused, or (semi-) automatically built, however, we do not focus on this point in this paper. More formally, the ontology  $\mathcal{O}$  is an OWL ontology defined as a tuple  $(\mathcal{C}, \mathcal{P}, \mathcal{I}, \mathcal{A})$  where  $\mathcal{C}$  is a set of classes,  $\mathcal{P}$  a set of (datatype, object and annotation) properties characterizing the classes,  $\mathcal{I}$  a set of individuals and property assertions, and  $\mathcal{A}$  a set of axioms including constraints on classes and properties: subsumption, equivalence, type, domain/range, characteristics (functional, transitive, etc.), disjunction.

Figure 1 shows an excerpt of an ontology in the domain of holiday destinations. The classes Activity, Environment and FamilyType are respectively the roots of a hierarchy, e.g., Environment expresses the natural environment (Aquatic, Desert, etc.) or its quality (Beauty, View). Some object properties represented on the figure have subproperties, not represented here. Datatype properties are represented under their domain class. Individuals are not represented.

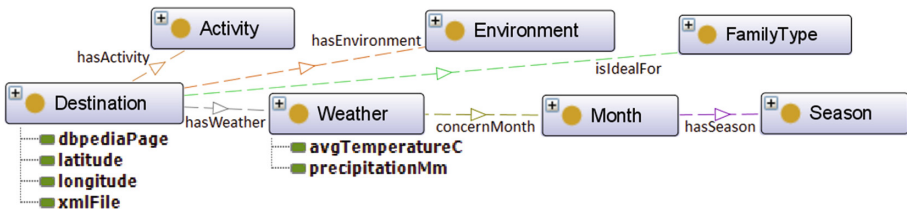


Fig. 1. The structure of the destination ontology

$\mathcal{C}$  groups two types of classes. *The main class* corresponds to the general type of entities described in the corpus, e.g., Destination. *Descriptive classes* are all other classes, useful to define the main class, e.g., Activity.

$\mathcal{P}$  is the set of properties characterizing the classes, datatype or object. A property assertion is a triple  $\langle s, p, o \rangle$  which links an individual  $s$  to an other individual or a literal  $o$  via a property  $p$  of  $\mathcal{O}$ . For example, if "Destination hasActivity Activity" is an axiom in  $\mathcal{A}$ , if  $d$  and  $a$  are respectively instances of Destination and Activity, then  $\langle d, \text{hasActivity}, a \rangle$  may be a property assertion. No object/datatype property assertions are initially expressed. Our aim is to collect them.

$\mathcal{I}$  initially contains instances of descriptive classes, e.g., `_rainForest` is an instance of `Forest` (descendant of `Environment`) and `dense forest` is one of its labels.

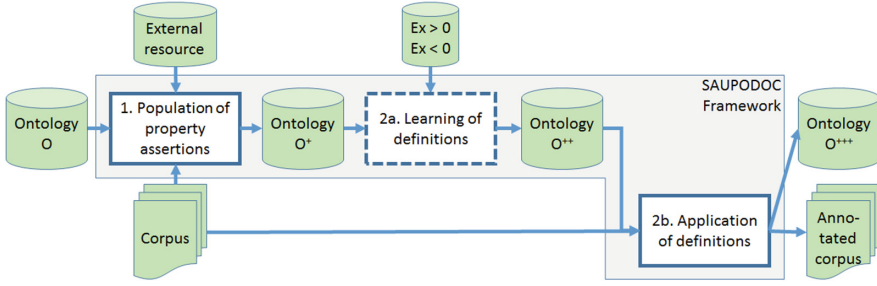
**The Target Concepts.** They are simple names of concept like "Destinations where one can do Water sports during Winter" (*DWW*), listed by the designer. Target concepts will be introduced in the ontology as specialized classes of the main class. One target concept will be denoted by  $tc$  in the following.

**The Corpus of Documents.** These are XML documents describing a domain entity, with very little structure. The structure of documents highlights the name of the entity and its textual description (containing labels of instances of descriptive classes). In our context, documents may be extracted from advertising catalogs, praising the assets of the entity described. In any cases, they describe the main features of entities and few negative expressions are present. However, names of target concepts are not explicitly mentioned. In our approach, some documents have to be manually annotated by the designer, for all the target concepts, as positive or negative examples of target concepts, i.e., either by  $tc$  or by *not tc*. It is not very time-consuming since the designer associates target concepts to a whole document. As an expert of the domain, he can provide the annotations based on his own background knowledge. He is not obliged to analyze precisely the document content or to seek information from external resources.

### **The Correspondences Between the Ontology and External Resources.**

We distinguish two types of properties, *document properties* and *external properties*. Documents from one corpus are supposed to be complete w.r.t. *document properties*. For example, the documents describing destinations mention the activities that one can do in this destination. When an activity is not mentioned, we suppose it cannot be practiced in this destination. Nevertheless, documents are incomplete w.r.t. *external properties*. This means these properties are not mentioned at all in the documents, e.g., the weather in the destination corpus.

*Document properties* can be asserted from each document. However, *external properties* need to be asserted from external resources. In this paper, we focus our completion with data from LOD (Linked Open Data). The designer indicates the *external properties* and selects the most relevant LOD datasets to populate them. This task can be performed manually because the number



**Fig. 2.** The SAUPODOC workflow

of *external properties* is not too large: these properties deal only with precise information not included in the documents (like numerical data) or with misinterpretation of documents. However, the ontology and the LOD datasets differ in regards to vocabulary terms and structure. Complex correspondences have to be established. We propose a model in Sect. 4.2 to support their specification.

### 3.2 Functional Description

The SAUPODOC approach is based on four tasks guided by the ontology. The first two tasks aim to populate the ontology (step 1) with property assertions. The next two tasks (step 2) are two alternative reasoning tasks. Step 2a discovers formal definitions of target concepts while Step 2b populates the ontology classes corresponding to those definitions. This population is equivalent to an annotation of the documents. Indeed, if a document  $d$  is (respectively is not) an instance of a class corresponding to a target concept  $tc$ , then  $d$  is annotated with  $tc$  (respectively *not*  $tc$ ). We call  $tc$  a positive annotation and *not*  $tc$  a negative annotation w.r.t. the target concept  $tc$ . The annotation process is executed off-line.

Figure 2 describes the workflow of the approach. The initial ontology  $O$  is progressively populated and enriched. First, each document entity is introduced as an instance of the main class. Each textual description is used to populate the ontology with property assertions expressing the features of each entity. These assertions are then completed thanks to information found in external resources ( $O^+$ ). Target concepts are inserted into the ontology as classes, called target classes, which are specializations of the main class. Their definition is learned based on manually annotated examples ( $O^{++}$ ). Finally, definitions apply in order to populate target classes ( $O^{+++}$ ) and annotate the corpus with target concepts.

## 4 Tasks of the Approach

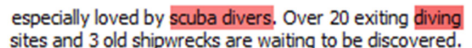
This section presents the various tasks exploiting data at different abstraction levels (classes and individuals) and having to cooperate to reach the final goal.

The pivotal role of the ontology is central in the approach. A preliminary task creates, for each document, an instance of the main class representing the entity described in the document. For example, in the destination domain, an individual `Dominican_Republic` is created from the document describing it. This individual is created such as `<Dominican_Republic rdf:type Destination>`. For each entity, the two tasks of step 1 populate the ontology with information that will be used by the two reasoning tasks of step 2. In what follows, we present how the ontology is a guide for each of these tasks.

#### 4.1 Data Extraction from Texts

The first task of step 1 extracts data from documents. Its goal is to enhance the ontology with assertions of *document properties*. This extraction is guided by the ontology, more particularly by its terms related to instances of descriptive classes being ranges of the *document properties*. If there is a match between a term and a document, then a property assertion is added in the ontology. For example, the constraint `<Destination, hasActivity, Activity>` requires that the range value of the property `hasActivity` belongs to the extension of the class `Activity`. From this constraint, if the text describing an entity `e` contains a term of an instance `a` of `Activity`, then the assertion `<e, hasActivity, a>` is built. Figure 3 represents an excerpt of a document describing the Dominican Republic. The expressions `scuba divers` and `diving` are terms of the individual `_diving`, which is an instance of a class specializing `WaterSport` (subclass of `Activity`). So, the assertion `<Dominican_Republic, hasActivity, _diving>` is added. Note that in the two study cases of our experiments, we specialized property ranges to avoid having two properties with the same range. The approach should be extended in a future work to take into account cases where such a specialization is impossible.

In our work, we use GATE [6, 10], an open source software performing a lot of text processing tasks. The GATE resource `OntoRoot Gazetteer`, in combination with other generic GATE resources, can produce labeling over textual documents, called lookups, w.r.t. an ontology given as input. GATE was chosen for its ability of using an external ontology as input, unlike other tools such as `Open Calais` [1] which annotates with named entities, facts or events but cannot be used with an external ontology. GATE can be used with a JAPE transducer, which applies JAPE (Java Annotation Patterns Engine) rules. In our context, JAPE rules transform lookups into property assertions. They are automatically created from one pattern, used whatever the ontology. The pattern is instantiated for all *document properties*, creating as many JAPE rules as *document properties*.



**Fig. 3.** Excerpt of the document on the Dominican Republic annotated by GATE

Note that we are in a context where documents describe the main features of entities and do not include negative expressions that could disrupt the process. Thus, a simple information extraction task like this one is appropriate.

## 4.2 Data Completion with External Resources

Textual descriptions are often short and do not contain all the necessary information. For instance, defining a *DWW* requires to know temperature and precipitation during winter for every destination. This data is not mentioned in descriptions. Data collection has to be enriched exploiting available on line resources. This is the second task of step 1. Again, this task is guided by the ontology. It involves to find a RDF resource dealing with entities of the corpus and to identify in this resource what properties correspond to those required by the ontology (*external properties*).

We chose to work with DBpedia [4] and we use DBpedia Spotlight [18], a tool able to automatically annotate a text with references to DBpedia entities. Applied on the entity name of each document, it gives a direct access to the DBpedia resource corresponding to the entity, unlike other tools like Wikifier [9,22] or AIDA [28], which return Wikipedia pages.

In this section, we present a model of data acquisition used by the designer to express (i) correspondences and (ii) access paths. The model of acquisition is currently used to extract information from DBpedia (but can be used for other LOD datasets) and insert property assertions into the ontology.

**Correspondences.** Since the vocabulary from the ontology and from the resource can differ, mechanisms have to be conceived to establish correspondences between the required elements and those from the resource. A model, briefly presented here, allows the designer to express these correspondences and supports automatic generation of SPARQL queries. A correspondence consists in associating an *external property* from the ontology with a property expression from the target data source, called  $PE_t$ . This  $PE_t$  is not necessarily explicitly represented in the target data source, instead, it may result from complex treatments.

A property expression in the target source ( $PE_t$ ) is a property  $p$  or its inverse  $p^{-1}$ , or an expression ( $f$ ) using one or several property expressions in the target source. A  $PE_t$  may include constraints ( $Constr$ ).

$$PE_t = p \mid p^{-1} \mid f(PE_t) \mid f(PE_t, PE_t) \mid PE_t.Constr$$

By recursion, a  $PE_t$  can be a function of  $n$   $PE_t$ . The function  $f$  is specified by the designer. He has to indicate whether it is a function of aggregation (minimum, average, etc.), of transformation (mathematical calculation, concatenation, etc.), or a set-theoretic operation (union, difference), and to clarify its nature.  $Constr$  represents any domain or range constraints.

For example, the *external property* precipitation\_in\_January is expressed by six different properties in DBpedia (janPrecipitationMm, janRainMm, janPrecipitationInch, janRainInch, janPrecipitationIn and janRainIn). So, the correspondence of precipitation\_in\_January can be expressed as the union of the values of these six properties, or even the average value of their union.



**Access Paths.** Since external resources are incomplete (like DBpedia), some properties from a  $PE_t$  may be missing. In our settings, having complete information is essential in order to achieve the best results as output of the entire process. Consequently, we propose an alternative way to get missing data, by browsing close resources. The idea is to have an approximation of the data, which is better than nothing.

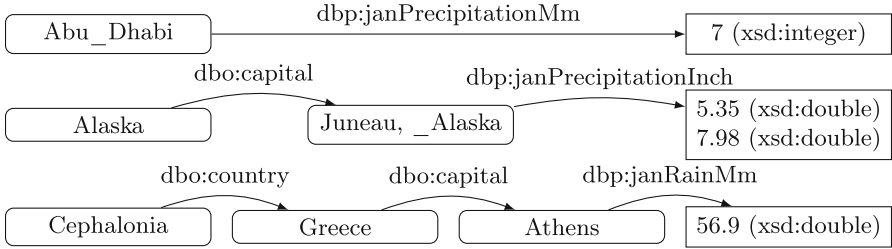


Fig. 4. Access paths in DBpedia

This mechanism is based on the composition of properties and allows the designer to establish access paths, to reach resources containing the required information. For example, Fig. 4 shows two examples (Alaska and Cephalaria) where the values for January precipitation are not available for the destination but where an approximation can be found for a close entity (the capital here).

These complex correspondences with their access paths are specified by the designer. CONSTRUCT SPARQL queries are automatically built based on these specifications, allowing to collect data via SPARQL endpoints in a transparent way and to insert ontology assertions in the source ontology. This process is not the focus of the paper. It will not be discussed here.

### 4.3 Learning the Definitions of Target Concepts

The first task of step 2 is a reasoning task, which is executed only once. It aims to learn the definitions of target concepts, based on the manual annotations of documents provided by the designer and the data collected in step 1.

Most machine learning tools do not take into account explicit specifications of relations (subsumption, object/datatype properties) between features as it is expressed in an ontology. Three tools, YinYang [11], DL-FOIL [12] and DL-Learner [15], use inductive logic programming (ILP) on Description Logics (DL) to perform concept learning. We chose to use DL-Learner, an open-source software capable of learning definitions of classes expressed in DL, from expert-provided examples, using an ontology as input. It allows us to get an explicit definition for all target concepts, an important point in concrete applications. The DL-Learner definitions are conjunctions and disjunctions of elements. An element can be a class (Destination) or an expression using object properties

(hasActivity some Nightlife), numerical datatype properties (avgTemperatureC some double[ $\geq$  23.0]), or cardinality constraints (hasCulture min 3 Culture). Ranges are conjunctions and disjunctions of elements. For example, the definition of *DWW* that can be learned by DL-Learner, could be something like this:

```
(Destination and (hasActivity some Watersport)
  and (hasWeather min 2 ((concernMonth some (hasSeason some MidWinter))
    and (avgTemperatureC some double[ $\geq$  23.0])
    and (precipitationMm some double[ $\leq$  70.0])))).
```

DL-Learner parameters were chosen based on the user manual and discussions with the software developers. We use the CELOE algorithm [16] announced as the best class learning algorithm currently available within DL-Learner, and the default reasoner, called fast instance checker, making the closed-world assumption (CWA). However, learned definitions have to be incorporated in an OWL ontology where reasoning is based on the open world assumption (OWA). To be able to learn and exploit minimum cardinality constraints, e.g., (hasActivity min 3 Activity), instances are automatically expressed as disjoint (Unique Name Assumption) to avoid being linked by owl:sameAs. Moreover, we disable the negation operator (NOT), the operator of universal restriction (ONLY), and operator of the maximum cardinality restriction (MAX), so that the obtained definitions are applicable under OWA.

Some DL-Learner parameters have been set making a compromise between the expressiveness of the definition and the execution time. Hence, we allow statements with a cardinality value set maximum at 10 instead of 5 (default value) such as definitions like `hasObjectProperty min 10 class_name` can be learned and the maximum execution time is set at 200s. These parameters will be used whatever the application. An other important parameter is the noise percentage, i.e., the percentage of positive examples not covered by a definition. We have proceeded by trial and error to set it. Hence, we have developed a methodology from the conducted experiments, where 5 different values for the noise percentage (5–15–25–35–45 %) are tested, and tuned using test experiments. Moreover, in case of really complex and far from easy to learn definitions, we set up parameters to apply a search heuristic to get longer definitions. When these latter parameters are activated, we call it the complex configuration, when they are not, the basic one. This means 10 configurations are tested: the basic and complex configurations both with 5 different values for the noise percentage. For each test, the highest ranked solution which is the best one in terms of accuracy and length is automatically kept. For each target concept, the best definition from the 10 tests is chosen.

#### 4.4 Reasoning to Populate Target Classes and Annotate Documents

The second task of step 2 consists in applying the learned definitions to populate the target classes in the ontology. This task is done any time new descriptions have to be annotated. We chose to use FaCT++ [26], an OWL-DL reasoner,

which scales well with a large number of instances, unlike HerMiT [24] and Pellet [25] according to our experiments. Indeed, these two reasoners have never terminated the process when used on 10,000 documents. FaCT++ relies on the definitions of target concepts to identify the document entities that should be annotated with a target concept. For each target concept  $tc$ , if the entity described in a document  $d$  fits the definition of  $tc$ , then this entity becomes an instance of the class representing  $tc$ . In that way, the document  $d$  is annotated by  $tc$ . On the opposite, if the entity does not fit the definition of  $tc$ , then this entity does not become an instance of the  $tc$  class and  $d$  is annotated by *not tc*. Doing this sort of annotation relies on a closed world assumption (CWA), whereas the reasoning with OWL is generally based on an open world assumption (OWA). Nevertheless, our context is particular and allows us to simulate the CWA in each step. In step 1, for each entity, if a property assertion is not created, then we consider it does not exist. Indeed, the extraction of property assertions from documents operates under CWA since documents are supposed to be complete for all *document properties*. Moreover, the extraction of property assertions from external resources nearly operates under CWA too thanks to the model of acquisition presented in Sect. 4.2. Indeed, access paths providing approximate values are good ways to overcome incompleteness. For step 2a, as stated in Sect. 4.3, definitions respect the CWA.

## 5 Experimental Evaluation

### 5.1 Procedure

To evaluate the annotation process, we compare our approach with classification approaches. Classification approaches are able to annotate documents in the same way as SAUPODOC, i.e., with a positive annotation ( $tc$ ) or negative annotation (*not tc*) for each target concept  $tc$ . In this section, we assess the quality of the annotation. To do that, a set of annotated documents is used. This set is split. The training set (2/3) provides positive and negative examples given in inputs to learn definitions. The testing set (1/3) is used to compare the annotations obtained by each process with the correct annotations. The comparison is based on several metrics: precision, recall, accuracy and F-measure.

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN}$$

$$Acc. = \frac{TP + TN}{TP + FP + TN + FN} \quad F\text{-measure} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

To make a fair assessment between SAUPODOC and classification approaches, we consider the same domain terminology. Indeed, SAUPODOC is based on an ontology but classifiers are not. The input of classifiers consists of documents represented with a list of features and a label. The label is binary w.r.t. a target concept (the annotation of the document with this target concept is true or false). To make the list of features, we consider the domain terminology of the

ontology, i.e., the set of labels and key-expressions associated to each individual, as a domain dictionary. We call a *word* of the dictionary, a keyword or keyphrases from the domain dictionary. Two expressions referring to a same idea (same individual) are put into a same word of the dictionary, e.g., *scuba divers* and *diving*. All the words of the dictionary are lemmatized. Each document corresponds to a vector of features (Vector Space Model). A bag-of-words method is used. This means that each element of the vector (each feature) corresponds to a word of the dictionary. Documents are lemmatized. If a word of the dictionary is found, the value in its vector for its element is the TF-IDF value, otherwise 0. In summary, for a corpus, a list of vectors is made, one vector for each document representing its content w.r.t. the list of features, and a label w.r.t. a target concept. For each target concept, we launch the classifier with the same features but we change the binary label, depending on whether the annotation is true or false.

Two classifiers are tested: an SVM classifier and a decision tree classifier. We use several parameters. Indeed, several values are tested for kernel (PolyKernel with several exponents, RBF with several gammas) and complexity for SVM; and several confidence factors for decision trees. We only keep the best results, i.e., the results with the best accuracy on the test set.

## 5.2 Versions Used in the Evaluation

The experimental evaluation has been performed using the following versions of the SAUPODOC components: GATE 8.0, DBpedia 2014, DL-Learner 1.0, FaCT++ 1.6.2.

For classifiers, we used Stanford NLP 3.4.1 [17] to lemmatize and Weka 3.6.6 [13] to execute the classifiers.

## 5.3 The Two Tested Corpora

Experiments were conducted in two application domains, each one having different characteristics. The objective is to see whether the size of the corpus and the ontology richness affect the quality of results.

**The Destination Corpus.** The corpus of destinations contains 80 documents, which have been automatically extracted from the Thomas Cook catalog [2]. Each document describes a specific place (country, region, island or city). The documents are promotional, i.e., they express the qualities of destinations and have hardly any negative expressions. Geolocation and meteorological data are missing. This information will be extracted from DBpedia thanks to the model of acquisition (Sect. 4.2).

The main class of the ontology is *Destination*. There are 161 descriptive classes, which express the nature of the environment (46 classes), the activities that can be done (102 classes), the kind of family that should go there, e.g., people with kids, couples, etc. (6 classes) and the weather information (7 classes), e.g., the seasons. The individuals, which are instances of these descriptive classes,

have terminological forms via annotation properties (`label`, `isDefinedBy`). For example, the terms `archaeology`, `archaeological`, `acropolis`, `roman villa`, `excavation site`, `mosaic` are terminological forms of the individual `archaeology`.

In all, 39 target concepts are under study. Every destination of the corpus is annotated by the designer as a positive or negative example for each target concept.

**The Film Corpus.** The film corpus contains 10,000 documents. These documents have been created via an automatic extraction from DBpedia. Each document corresponds to a film described in a DBpedia resource. The document contains the DBpedia URI from which it was extracted and the abstract of the film. The abstract of film fits our context, i.e., there are hardly any negative expressions. Since the DBpedia URI is provided, we do not need to use DBpedia Spotlight to get it. Duration of the film as well as languages and countries are extracted from DBpedia. Indeed, duration is missing in the description. Languages and countries might be mentioned but a misinterpretation is possible, this is the reason why we prefer to trust DBpedia than documents for these properties. For instance, the presence of the term “French” may have different meanings. The film may be in French (language), or it may be a French film (country), or it can tell the story of a French person.

The film ontology is basic. It contains the main class `Film` and five descriptive classes expressing characteristics about films. The ontology only contains the classes needed for defining the target concepts in our experiments. It should be completed in case of new target concepts.

In total, 12 target concepts are taken into consideration. They correspond to some DBpedia categories. Annotated examples are automatically generated for films: a film  $f$  is a positive example for a target concept corresponding to a category  $c$  if  $\langle f \text{ dcterms:subject } c \rangle$  according to DBpedia, otherwise it is a negative example.

Let us note that, for our approach, the domain terminology may be partial, for which external resources are used. For example, there are no terms about either languages or countries in our film ontology but the DBpedia step can add instances of them. To be fair for classifiers, terms from these external resources are also added into the domain dictionary of classifiers.

## 5.4 Validation of the Approach

Table 1 shows the results for the testing set, i.e., the part not used in the process of construction of annotations. We can see that all the three approaches give good results in terms of accuracy, even if SAUPODOC has slightly better ones. However, accuracy is not the only metric to take into account. Most of the target concepts have many negative examples and few positive examples. Hence, if a classifier predicts negative for each document given in input, then the accuracy is high, e.g., 91.76 % on average for target concepts of the film corpus. Moreover, no instances of each target classes is found, which means all documents are annotated with negative annotations. As already said, our approach is currently used

for a Business to Consumer application whose goal is to provide users with entities matching their needs in a particular domain. This means it is important to obtain positive annotations. This is why alternative metrics like precision, recall and f-measure are needed. They allow us to evaluate the positive annotations, which are central in this settings. From Table 1, we can observe that SAUPODOC significantly outperforms classifiers for these three metrics.

**Table 1.** Average results for target concepts

Metric (%)		Accuracy			F-measure		
Approach		SAU-PODOC	SVM	Decision Tree	SAU-PODOC	SVM	Decision Tree
Corpus							
Destination (39 target concepts)		95.89	84.52	86.23	72.23	54.14	63.22
Film (12 target concepts)		95.46	94.41	94.32	75.65	61.74	61.40

Metric (%)		Precision			Recall		
Approach		SAU-PODOC	SVM	Decision Tree	SAU-PODOC	SVM	Decision Tree
Corpus							
Destination (39 target concepts)		73.95	58.10	64.23	71.58	55.32	65.89
Film (12 target concepts)		76.27	69.90	67.72	77.76	57.59	58.99

Moreover, for all the target concepts ( $39 + 12 = 51$ ) from the two corpora, SAUPODOC generates 8 target concept definitions that assign every input of the test set as a negative example. To avoid this, as definitions are intelligible, the designer can easily refine them in order to have some positive examples. This refinement is mandatory in contexts such as Business to Consumer applications, for which positive examples are needed for all target concepts.

Classifiers encounter the same problem. However, a refinement is impossible, since there is no explicit definitions. Indeed, SVM classifiers create a model, which is not comprehensive for human. Decision tree classifiers are more comprehensive since trees can be seen as sets of rules. But here, these rules deal with the TF-IDF number associated with a dictionary word, which is hard to be interpreted by humans. For example, the decision tree for the target concept *coastal destinations* is given in Fig. 5 (accuracy of 96.30 %). It means positive annotations are made when the TF-IDF value of the dictionary word *urban* is less than 0.18893 and *beach* is less than 0 and *sea* is more than 0.005502, or if *urban* is less than 0.18893 and *beach* is more than 0. Such a tree is difficult to be adjusted by the designer in case a refinement is needed.

```

_urban <= 0.018893
|
|  _beach <= 0
|  |
|  |  _sea <= 0.005502: 0
|  |  |
|  |  |  _sea > 0.005502: 1
|  |  |
|  |  |  _beach > 0: 1
|  |  |
|  |  |  _urban > 0.018893: 0

```

**Fig. 5.** The decision tree for coastal destinations

## 6 Conclusion and Future Work

We proposed an ontology-driven approach dealing with semantic annotations of documents describing entities of a same domain. Annotation is performed with

a list of concepts, which are not explicitly mentioned into the documents. The approach, called SAUPODOC, is based on an ontology, and combines population and enrichment steps. It makes tasks cooperate both at individual level and at concept level. Innovative mechanisms have been implemented to exploit the LOD. Property complex correspondences between the ontology and a data set can be defined and alternatives to missing data are provided. Results clearly show the benefit over well-known classifiers and the relevance of an ontology-based approach relying on a particular combination of various techniques to semantically annotate documents.

This work has been validated by the Wepingo company. It is a part of a wider approach to answer user's queries whose keywords are specific concepts by delivering documents related to instances of these concepts. Future work will be devoted to the integration of this semantic annotation process, extended with automatic generation of SPARQL queries to easily access LOD data, into a framework supporting the overall approach.

**Acknowledgements.** This work has been funded by the PORASO project, in the setting of a collaboration with the Wepingo company.

## References

1. <http://www.opencalais.com/>
2. <http://www.thomascook.com/>
3. Alec, C., Reynaud-Delaître, C., Safar, B., Sellami, Z., Berdugo, U.: Automatic ontology population from product catalogs. In: Janowicz, K., Schlobach, S., Lambrix, P., Hyvönen, E. (eds.) EKAW 2014. LNCS, vol. 8876, pp. 1–12. Springer, Heidelberg (2014)
4. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.G.: DBpedia: a nucleus for a web of open data. In: Aberer, K., Choi, K.-S., Noy, N., Allemang, D., Lee, K.-I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P. (eds.) ASWC 2007 and ISWC 2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007)
5. Béchet, N., Aufaure, M.A., Lechevallier, Y.: Construction et peuplement de structures hiérarchiques de concepts dans le domaine du e-tourisme. In: IC, pp. 475–490 (2011)
6. Bontcheva, K., Tablan, V., Maynard, D., Cunningham, H.: Evolving GATE to meet new challenges in language engineering. *Nat. Lang. Eng.* **10**(3/4), 349–373 (2004)
7. Cambria, E., Fu, J., Bisio, F., Poria, S.: AffectiveSpace 2: enabling affective intuition for concept-level sentiment analysis. In: AAAI, pp. 508–514 (2015)
8. Cambria, E., White, B.: Jumping NLP curves: a review of natural language processing research [review article]. *IEEE Comp. Int. Mag.* **9**(2), 48–57 (2014)
9. Cheng, X., Roth, D.: Relational inference for wikification. In: EMNLP (2013)
10. Cunningham, H., et al.: Text Processing with GATE. University of Sheffield Department of Computer Science, Sheffield (2011)

11. Esposito, F., Fanizzi, N., Iannone, L., Palmisano, I., Semeraro, G.: Knowledge-intensive induction of terminologies from metadata. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 441–455. Springer, Heidelberg (2004)
12. Fanizzi, N., d’Amato, C., Esposito, F.: DL-FOIL concept learning in description logics. In: Železný, F., Lavrač, N. (eds.) ILP 2008. LNCS (LNAI), vol. 5194, pp. 107–121. Springer, Heidelberg (2008)
13. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. *SIGKDD Explor.* **11**(1), 10–18 (2009)
14. Kessler, R., Béchet, N., Roche, M., Moreno, J.M.T., El-Bèze, M.: A hybrid approach to managing job offers and candidates. *Inf. Process. Manage.* **48**(6), 1124–1135 (2012)
15. Lehmann, J.: DL-Learner: learning concepts in description logics. *J. Mach. Learn. Res.* **10**, 2639–2642 (2009)
16. Lehmann, J., Auer, S., Bühmann, L., Tramp, S.: Class expression learning for ontology engineering. *J. Web Seman.* **9**, 71–81 (2011)
17. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The stanford CoreNLP natural language processing toolkit. In: 52nd ACL: System Demonstrations, pp. 55–60 (2014)
18. Mendes, P.N., Jakob, M., García-Silva, A., Bizer, C.: DBpedia spotlight: shedding light on the web of documents. In: *I-Semantics*, pp. 1–8. ACM, New York (2011)
19. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. *Linguisticae Investigationes* **30**, 3–26 (2007)
20. Oren, E., Möller, K., Scerri, S., Handschuh, S., Sintek, M.: What are Semantic Annotations? Technical report, DERI Galway (2006)
21. Petasis, G., Möller, R., Karkaletsis, V.: BOEMIE: reasoning-based information extraction. In: *LPNMR*, pp. 60–75. A Corunna, Spain (2013). [CEUR-WS.org](http://www.ceur-ws.org)
22. Ratinov, L., Roth, D., Downey, D., Anderson, M.: Local and global algorithms for disambiguation to wikipedia. In: *ACL* (2011)
23. Reeve, L.: Survey of semantic annotation platforms. In: *ACM Symposium on Applied Computing*, pp. 1634–1638. ACM Press (2005)
24. Shearer, R., Motik, B., Horrocks, I.: HermiT: a highly-efficient OWL reasoner. In: *OWLED*, vol. 432 (2008). [CEUR-WS.org](http://www.ceur-ws.org)
25. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: a practical OWL-DL reasoner. *J. Web Seman.* **5**(2), 51–53 (2007)
26. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: system description. In: Furbach, U., Shankar, N. (eds.) *IJCAR 2006*. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006)
27. Yelagina, N., Panteleyev, M.: Deriving of thematic facts from unstructured texts and background knowledge. In: Klinov, P., Mourontsev, D. (eds.) *KESW 2014*. CCIS, vol. 468, pp. 208–218. Springer, Heidelberg (2014)
28. Yosef, M.A., Hoffart, J., Bordino, I., Spaniol, M., Weikum, G.: AIDA: an online tool for accurate disambiguation of named entities in text and tables. *PVLDB* **4**(12), 1450–1453 (2011)