Unsupervised Web-based Automatic Annotation¹

Miquel Millán², David Sánchez, Antonio Moreno
{josemiguel.millan, david.sanchez, antonio.moreno}@urv.cat
Intelligent Technologies for Advanced Knowledge Acquisition (ITAKA) Research Group
Universitat Rovira i Virgili, Tarragona, Spain

Abstract. The success of the Semantic Web depends both on the definition of ontologies used to represent the knowledge as on the annotations performed of the web contents. As manual approaches have low scalability, there is a need of tools capable to generate all this knowledge in an automatic and reliable way. In this paper is presented a complete algorithm to annotate web contents in an automatic and unsupervised manner. It is structured in a three-stepped procedure, based on the usage of several concept similarity measures and linguistic patterns. It is able to detect the entities to annotate, the candidate classes of these entities and, finally, associate them with the classes of an ontology. Some prospective results are presented.

Keywords. Semantic web, automatic annotation, ontologies

1. Introduction

Since the creation of the World Wide Web (WWW), presented by Tim Berners-Lee in 1989, its structure and architecture have been in constant growth and development. Nowadays the Web is involved in what we familiarly know as the Social Web, where all its users are able to add and modify its contents. This has brought lots of new information to the Web and its size has grown up to $4 \times 10^9$ static pages [1] (the surface web) plus the so-called deep web, which consists in the dynamically created web pages. Although this increase of information could seem a very interesting feature, the lack of structure brought some problems: it complicates its accessing, as it cannot be interpreted semantically by IT applications [2], both manually and in an automatic way. So, in order to solve these inconveniences a new global initiative has been proposed [3]: the Semantic Web.

The Semantic Web relies on a set of domain ontologies where the knowledge is structured and, using them, proposes a semantically annotated Web in which search engines could interpret the Web information. This will result in an increase of the quality of the results presented by the search engines to the user. However, to achieve

¹ This work has been partially supported by the K4Care European research project (IST-2004-026968)
² Corresponding Author, Intelligent Technologies for Advanced Knowledge Acquisition (ITAKA) Research Group, Universitat Rovira i Virgili, Tarragona. Avda. Països Catalans, 26, 43007 (Tarragona); E-mail: josemiguel.millan@urv.cat
the goals of the Semantic Web there are some issues to be solved. On the one hand it is assumed to have the knowledge represented in ontologies. On the other hand, it is expected to have the entire web resources annotated with the different concepts represented in ontologies. However, due to the manual knowledge representation bottleneck, there is a lack of these annotations nowadays. To aid to this situation some solutions to the annotation problem have been proposed in the past [6]. However, as it will be described in Section 2, manual and supervised solutions are does not scale well and there is a little effort on the automatic side.

In this paper, an Unsupervised Web-based Automatic Annotation system for text documents is proposed. A new methodology to detect Named Entities, which are candidates to be annotated in a document, is described. Next the possible classes to which each named entity may belong are extracted using the linguistic patterns applied over Web information. In a third step the most appropriate class of the input ontology is chosen as the tag for each Named Entity, using dictionaries and the information collected in the second step. For illustrative purposes, some preliminary results of our Web content annotator are shown.

The rest of the paper is organized as follows; in Section 2 related work is presented, in Section 3 the procedure is described in detail, introducing the basic learning techniques employed. In Section 4 some brief results are presented. Finally, in Section 5 future work and conclusions are discussed.

2. Related work

Nowadays there are several systems and algorithms to annotate documents. As described in [6] these solutions can be classified in different ways: considering their level of automatism or their architecture, their need of previous knowledge to generate new annotations or the nature of the contents to be annotated.

Three levels of automatism can be distinguished. In the first one, no automatic annotation is done, and only the tools to annotate are given to the user [7]. The second one is the support systems: they do not annotate automatically, but they suggest possible annotations; some examples can be found in [8][9][10]. These systems are great for annotate documents created from scratch; however, considering the amount of already available document in the Web, they are not so useful. Lastly, there are a few completely automatic solutions capable to automatically annotate contents like [11][12], which are more suitable to annotate the Web considering its magnitude.

From the point of view of its internal architecture, some of these systems are designed as a framework [13], which could be used by other bigger tools to annotate contents, or used as stand-alone tools. Even though it is possible to find different solutions, like plug-ins to give annotation functionalities to tools that are not designed to annotate contents [14].

From a technological point of view, some of these systems use the available standards. Most of them are capable to read ontologies in different formats (OWL, DAML+OIL, etc) and to annotate the contents using different standards like XPointer, RDF, etc.

In addition, they can have different orientations depending on the contents they are supposed to annotate. Most of them are focused on annotate textual documents, both raw text documents and semi-structured (XML, HTML, SGML, etc) ones. Besides
these documents, it is possible to find other solutions [16] oriented towards the
annotation of multimedia documents like photos, music or videos.

There are supervised systems that need previously annotated contents to learn the
annotation rules [15]. Others that are more powerful do not depend on this previous
information [11]. This is a critical issue, as the need of previous knowledge hampers
their performance and applicability.

As there is a dearth of automatic and unsupervised solutions, our contribution
consists in an annotation tool that does not need previous knowledge and is capable of
autonomously annotate contents (without interacting with the user), which makes it
very suitable for deal with already available web resources. It is also capable to
annotate documents using present-day standards and providing annotated documents
understandable by current tools.

3. Automatic Annotation Methodology

The procedure is structured in three basic steps and described in Figure 1. The first one
is the detection of the Named Entities [5] (considered as ontological instances) in the
document. The second one is the detection of the classes to which these entities may
belong by means of linguistic analysis (using text patterns based on Hearst Patterns
[17]) applied over Web documents. Finally, in the third step, the class candidates are
matched with the ones in the given ontology, in order to find which is the most
adequate annotation label; in other words, which is the most appropriate class to which
the Named Entity is an instance.

\[
\text{Annotate\_Document(} \text{Document d)} \\
\{ \\
\quad \text{tagged\_document} = \text{tag\_document}(d) \\
\quad \text{ne} = \text{extract\_named\_entities}(\text{tagged\_document}) \\
\quad \text{for entity in ne} \\
\quad \quad \{ \\
\quad \quad \quad \text{for pattern in text\_patterns} \\
\quad \quad \quad \quad \{ \\
\quad \quad \quad \quad \quad \text{abstracts} = \text{download\_SearchEng\_Abstracts(build\_pattern(entity, pattern))} \\
\quad \quad \quad \quad \quad \text{entity.class\_candidates} += \text{extract\_class\_candidates}(\text{abstracts}) \\
\quad \quad \quad \quad \\} \\
\quad \quad \quad \text{entity.class} = \text{Search\_class\_candidate}(\text{entity}) \\
\quad \quad \} \\
\quad \text{return generate\_annotated\_document(d, ne)} \\
\}\]

**Figure 1. Algorithm’s pseudo code**

3.1. Detection of Named Entities

In our work we have approximated the detection of textual entities to the discovery of
Named Entities [5]. From the unsupervised learning point of view, this problem can be
solved using several techniques like searching capitalized words [18], analyzing the
Noun Phrases in the text [11], using dictionaries (like WordNet), etc. Our Named
Entities detection procedure uses all these techniques incrementally.
Firstly the Noun Phrases are detected using a combination of text taggers\(^3\). After that the extracted set is refined using capitalized words filtering, statistical analysis using queries to Web search engines and checks against a dictionary. Regarding this last point, we concretely use WordNet.

WordNet is an important resource for automatic learning procedures, as it is the most commonly used online and offline lexical and semantic repository for the English language. In addition it can also be used locally making it very interesting in terms of efficiency and very helpful to reduce the usage of the network resources. Many authors have contributed to it \([19][20]\) and used it to do many different knowledge acquisition tasks \([21]\). It offers a lexicon, a thesaurus and semantic linkage between the major parts of the English words. It has the words organized in synonyms sets \((\textit{synsets})\), sets of words that have the same, or a very similar meaning, and thus they could be interchanged in some context, as they share a commonly agreed meaning, with little or no variation. At a higher level, it also has lexical and semantic pointers, which simply are directed edges in Wordnet whose nodes are synsets. These pointers described relationships between different synsets like, \textit{hyponymy}, \textit{meronymy}, \textit{attribute relationship}, \textit{“instance of” relationship}, etc. Based on its functionalities, it is possible to distinguish a common word from a Named Entity using WordNet, as Named Entities usually are not present in WordNet, or in case they are, they have a semantic pointer with an \textit{“instance of”}.

More in detail, the Named Entity detection process begins when the document’s HTML markup is cleaned to prepare the text to be annotated. Over this text, a four-step procedure is applied to detect the Named Entities. The first step consists on the detection of Noun Phrases that may contain Named Entities. This procedure is based in the composition of three taggers. This first tagging procedure (see \textbf{Table 1}) is done in order to prioritize the mark of capitalized words as Proper Nouns (thus, Named Entities candidates). Once the capitalized words are marked as possible Proper Nouns, the text is passed over two \textit{n-gram taggers}, which are trained with the Brown Corpus\(^4\), in order to refine the tagging of the rest of the words. First, a Unigram tagger, which annotates the words assigning the tag that is most likely for that particular word. After that, a Bigram tagger (which assigns tags depending on the preceding word) is passed over the resultant text. Once this combination of taggers is trained, it has a tagging precision of 93.4\% over the Brown Corpus.

After the text has been tagged, in the second step, a grammar (based on the tags presented in \textbf{Figure 2}), is used in order to detect the Noun Phrases (which may contain a Named Entity). This grammar describes the structure of a Noun Phrase which usually is composed by a central particle composed by one or more Proper Nouns, \(<\text{NNP}>^+\), followed or leaded by zero or more Nouns (both in singular as in plural), \(<\text{NN}\mid\text{NNS}>^*\) (e.g.- \textit{“Paris”}). Usually, this central particle is leaded by some optional determinants or/and adjectives forming a Noun Phrase (e.g.- \textit{“the city”}, \textit{“of lights”}). Eventually, a completely well formed Noun Phrase is composed by one or more single Noun Phrases (e.g.- \textit{“Paris the city of lights”}). As a result, a set of Named Entity candidates is retrieved.

\(^3\) To tag the text the NLTK taggers are used (\url{http://nltk.sourceforge.net}, last accessed on March 28, 2008) and the tagged Brown corpus is used to train the \textit{n-gram taggers}. The tag-set used is the Brown tag-set that can be found in \url{http://www.comp.leeds.ac.uk/amalgam/tagsets/brown.html} (last accessed on March 28, 2008).

\(^4\) See \url{http://icame.uib.no/brown/bcm.html} (Last accessed on March 28, 2008)
## Table 1. Regular expressions used to detect morphologically the text tags

<table>
<thead>
<tr>
<th>Regular Expression</th>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>[A-Z].*$</td>
<td>NNP</td>
<td>Proper Noun</td>
<td>Madrid</td>
</tr>
<tr>
<td>*.ing$</td>
<td>VBG</td>
<td>Gerund verb tense</td>
<td>distinguishing</td>
</tr>
<tr>
<td>*.ed$</td>
<td>VBD</td>
<td>Regular verb in past tense</td>
<td>distinguished</td>
</tr>
<tr>
<td>*.es$</td>
<td>VBZ</td>
<td>Verb in 3rd singular person, present tense</td>
<td>Distinguishes</td>
</tr>
<tr>
<td>*.ould$</td>
<td>MD</td>
<td>Modal verb</td>
<td>would</td>
</tr>
<tr>
<td>*.s</td>
<td>NN$</td>
<td>Singular common noun genitive</td>
<td>season’s</td>
</tr>
<tr>
<td>*.s$</td>
<td>NNS</td>
<td>Plural common noun</td>
<td>stadiums</td>
</tr>
<tr>
<td>*.al$</td>
<td>JJ</td>
<td>Adjective</td>
<td>global</td>
</tr>
<tr>
<td>^:[0-9]+([0-9]+)$</td>
<td>CD</td>
<td>Cardinal Number</td>
<td>125,000</td>
</tr>
<tr>
<td>.*</td>
<td>NN</td>
<td>Singular common noun</td>
<td>word</td>
</tr>
</tbody>
</table>

### NOUNP:
```
{<NN|NNS>*<NNP>*<NN|NNS>*}
```

### UNINP:
```
{<DT|DTI|DTS|DTX|FP|GJ|JJ-JT|JJ|JJS|JJS>*<NOUNP>}
```

### NP:
```
{<UNINP>?<NOUNP><UNINP>?}
```

### Figure 2. Noun Phrases detection grammar

Considering the possibility of using WordNet as a thesaurus, in the third step WordNet is used to distinguish common words from the proper nouns that compose the Named Entity candidate. In case all the NNPs in the Noun Phrase are found in WordNet, the candidate is immediately discarded, as it is a commonly used word. However, if one or more of them are not found there, or they are found as WordNet Instances, they are considered as valid candidates and will be evaluated in the fourth step.

The fourth step is based on the statistical observation of the candidates over the Web. It is introduced to detect and discard misspellings and to confirm that the candidate is typically presented in a Named Entity form. So, the remaining candidates from the previous stage are evaluated in front of their writing in the Web. Each one is queried in a publicly available web search engine. The abstracts obtained are joined in one piece of text and, the candidate is searched into this snippet set. The probability to find the text written as it is in the original form (which confirms that the candidate is typically presented in its Named Entity form) is evaluated using the following formula:

\[
\text{Score(Named Entity)} = \frac{\# \text{Case Sensitive Matchings}}{\# \text{Case Sensitive Matchings} + \# \text{Case Unsensitive Matchings}}
\]

It compares the number of matches written equally (same uppercase letters position and same letters) with the total matches (same letters) found. If it is higher than a certain threshold, the candidate is considered as a Named Entity. A minimum number of hits is also required in order to avoid misspelled terms (which are quite typical in the case of proper nouns).
3.2. Retrieval of class candidates

Once the Named Entities to annotate have been selected, it is imperative to find the domain class to which they should be annotated. A first approximation may consist in trying to match all the ontology classes with these Named Entities using a statistical analysis using web co-occurrence measures [4].

As shown in [22] several web based collocation scores have been designed. These measures rely on the fact that the World Wide Web is the largest database in the world and, because of its vastness and diversity, its word distribution may be taken as an estimation of the current use of the terms in society [23]. Taking this into consideration, a specialization of the Pointwise Mutual Information (PMI) measure (the PMI - Information Retrieval) can be computed from the web hit count presented by a search engine [22] when specific queries are constructed. The PMI-IR gives us a ratio of how related is a set of words with another set of words, estimated from the web information distribution.

Although these measures are robust and can be suitable to estimate the degree of relatedness between a Named Entity and each ontological class, they have a problem: the large amount of queries to a web search engine derived from their usage and the cost of these queries (approximately one second per query). Considering the possibility of using WordNet’s semantic interlinkage between words as a substitute of those measures, a solution to this problem would be to map directly these Named Entities and the classes in the ontology using WordNet similarity measures. Nevertheless, this is not possible, because as a stated in the previous section, very few Named Entities modeled in WordNet.

In order to solve those difficulties, we have introduced of an intermediate step in which class candidates for each Named Entity are automatically retrieved from the analysis of additional web resources, as proposed in other works like [11]. With these class candidates we would be able to use WordNet to relate them with the ontology classes (as it is likely that both are contained on it). At the end, we will employ some web queries to retrieve the needed web resources but the total number will be smaller than in the first approximation.

In order to discover Named Entity-class relationships, a pattern based taxonomical learning approach can be employed. Marti Hearst studied the use of text patterns to extract knowledge from text [22] in 1991. She described a set of text patterns and a method to acquire the hyponymy lexical relation from unrestricted text. Nonetheless, this technique has also been used to discover instance/concept relations [22]. Thus, we propose to find the class candidates for each Named Entity based on her patterns (see Table 2, where CONCEPT is the last Noun in the last Noun Phrase before the mark -in the first three patterns- or in the first Noun Phrase after the text mark -in the last two-).

On the other hand, two new patterns have been added to this list, as after some experimental results, we found that they provide good contextualization. They are formally described in Table 3.

These patterns are used in conjunction with the candidates extracted in the previous step (see Subsection 3.1), to construct queries for a web search engine replacing the INSTANCE part by each Named Entity. After that, the queries are applied, and the snippets obtained are analyzed to extract the CONCEPT part from them. Each CONCEPT found is added to the corresponding Named Entity compiling a class candidates list.
### Table 2. Hearst Patterns used

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>Pattern structure</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>HEARST 1</td>
<td>CONCEPT such as (INSTANCE)+ ((and</td>
<td>or) INSTANCE)?</td>
</tr>
<tr>
<td>HEARST 2</td>
<td>CONCEPT (?)+ especially (INSTANCE)+ ((and</td>
<td>or) INSTANCE)?</td>
</tr>
<tr>
<td>HEARST 3</td>
<td>CONCEPT (?)+ including (INSTANCE)+ ((and</td>
<td>or) INSTANCE)?</td>
</tr>
<tr>
<td>HEARST 4</td>
<td>INSTANCE (?)+ and other CONCEPT</td>
<td>Eifel Tower and other monuments</td>
</tr>
<tr>
<td>HEARST 5</td>
<td>INSTANCE (?)+ or other CONCEPT</td>
<td>Coliseum or other historical places</td>
</tr>
</tbody>
</table>

### Table 3. Additional Text Patterns

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>Pattern structure</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTRA PATTERN 1</td>
<td>INSTANCE (?)+ is a</td>
<td>Paris is a beautiful city</td>
</tr>
<tr>
<td>EXTRA PATTERN 2</td>
<td>INSTANCE (?)+ like other CONCEPT</td>
<td>Taj Mahal like other mausoleums</td>
</tr>
</tbody>
</table>

### 3.3. Class candidates matching with the ontology classes

In this third stage we associate the most appropriate ontology class with each of the Named Entities found in the first step by means of the class candidates retrieved in the second step. The ontology classes are compared with the class candidates of each Named Entity using a measure of their semantic similarity similarity. Without relying of web scale statistics which may result in poor performance, there exist similarity measures based on the semantic interlinkage between words in WordNet. Some of the available measures are:

- **Path finding**: This measure gives the path length between two concepts in the is-a hierarchy of WordNet. The path length is scaled by the depth of the hierarchy in which they reside to obtain the relatedness of the two concepts.
- **Information content**: Measures the specificity of a concept. The measure of relatedness between two concepts is the information content of the most specific concept that both concepts have in common.
- **Context vector**: This measure does not depend on the interlinkage between words that, in some situations, has a poor coverage in the WordNet semantic network. In more detail, it incorporates information from WordNet glosses as a unique representation for the underlying concept, creating a co-occurrence matrix from a corpus made up of the WordNet glosses. Each content word and each gloss have their associated context vectors representing a unique content or the average of all the context vectors of the words in a gloss, respectively. The relatedness between concepts is measured calculating the cosine between a pair of gloss vectors.

Concretely, we have used the *Path finding* measure because it is mainly based on taxonomical links and offers a low computation time. However, despite using these similarity measures, it is possible to have coverage problems using only WordNet as
not all the possible relations are modeled. So, this stage is used to filter the less similar ontological classes. At the end, the reduced set of the most similar ones are verified using collocation measures against the original Named Entity [22]. The most similar one is selected as the final annotation tag. Even some extra web search engine queries have to be done, the amount of them is in the order of dozens. Even added to the number of queries generated in the second step, it still remains much lower than in other approximations like [4].

This third stage is decomposed in several steps described in Figure 3:

```plaintext
Search_class_candidate (Named_Entity entity)
{
    similarities = []
    relevantClasses = []
    entity.class = find_direct_match(entity.class_candidates, ontology)
    if entity.class == ""
    {
        similarities = compute_WN_similarities (entity.class_candidates, ontology)
        for value in similarities
        {
            if value.similarity > SIM_THRESHOLD
            {
                relevantClasses.append(value)
            }
        }
        entity.class = class_of_Max_PMIIR(entity, relevantClasses)
    }
    return entity.class
}
```

Figure 3. Ontology class selection algorithm

So, for each Named Entity, the algorithm takes all the class candidates found in the previous stage. First it compares all the class candidates with all the ontology classes. It syntactically tags the class candidates and compares the main NN | NNP contained in each class candidate with each ontology class. In case that one of the ontology classes is the same than the class candidate, it is assigned to be the annotating label.

If a direct matching is not found, it is possible that the ontology contains the same semantic concept but expressed with different words, so we should look for its most similar one. It is here where the WordNet-based similarity measure is used to assess which of the ontology classes is more similar to one of the class candidates. A threshold is set in order to demand a minimum degree of similarity. If there is not any class similar enough, we suppose that the input ontology does not have any concept related with the concrete Named Entity.

As a result of the described filtering process, the set of ontology classes to evaluate is reduced (in experiments it is usually goes from dozens to 4-8). A final selection step is done over the remaining classes. In this final step we use a collocation measure, concretely the previously mentioned PMI-IR, because it is robust and independent of the WordNet’s semantic coverage. We calculate the PMI-IR between the Named Entity and each of the filtered class from the ontology; we choose the one with the highest
value, which is selected as the final annotation label. The concrete PMI-IR score used is calculated as follows:

$$\text{Score(ClassCandidate)} = \frac{\text{hits(NamedEntity AND OntologyClass)}}{\text{hits(OntologyClass)}}$$ (2)

At this point, one could think that some problems of semantic ambiguity can appear (e.g. “Barcelona” could be “a geographical place” or “a sports team”), however, as the ontology is set a priori, these problems are implicitly solved by the ontology definition.

3.4. Annotation

Once we have obtained the different Named Entities from the document and discovered to which class in the ontology they belong, we annotate the document with an annotation standard. Our objective is that these annotations should be readable by the available web browsers. Several standards such as XMLPointer5, RDF6 and HTML MicroFormats7 have been evaluated; as our priority was to use a standard readable nowadays, we decided to use HTML MicroFormats. They are an extension of the basic HTML, which let enrich it with semantic information. Even though it has semantic limitations, it gives the possibility to describe the class to which a concrete entity belongs. So, each Named Entity is annotated as follows:

```html
<a href="Ontology Class URL" class="Ontology Class">Named Entity</a>
```

Figure 4. HTML MicroFormats usage

Using this notation we are able to include the required semantic information with a low increase in the size of the document.

3.5. Runtime Complexity and Query Size

The runtime complexity of this algorithm for one document is $O(|Q|)$ where $|Q|$ is the number of queries done to the web search engine.

$|Q|$ can be split in $|N|+|N|\cdot |P|+|N|\cdot |O|$ where $|O|$ is the maximum number of classes in the ontology. In our case, $|P| = 7$ as we use seven patterns, so, for one document the cost is $O(|N|+7\cdot |N|+|O|\cdot |N|)$. From this, we can conclude that the algorithm has a linear cost depending on the number of Named Entities found $O(|N|)$.

4. Results and early Evaluation

As we are currently refining and tuning the presented methodology, the evaluation of the results is under development, but some preliminary evaluation results have been

---

5 http://www.w3.org/TR/WD-xptr (Last accessed on March 28, 2008)
6 http://www.w3.org/RDF/ (Last accessed on March 28, 2008)
obtained comparing the results of the algorithm with the ones extracted by a human expert. In this section, we present a set of preliminary results and these early evaluation values. We have applied the algorithm over various geographical Wikipedia articles\(^8\). The contents of these articles have been annotated with a modified version (adding more city concepts) of a geographical locations ontology\(^9\). We have summarized some of the results generated annotating one of these articles in Table 4.

Table 4. Results extract from Tarragona’s article annotation process

<table>
<thead>
<tr>
<th>Named Entity</th>
<th>Class candidates</th>
<th>Ontology classes related with class candidates</th>
<th>Assigned ontology class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hispania Tarracronensis</td>
<td>Spanish provinces</td>
<td>Province</td>
<td>Province</td>
</tr>
<tr>
<td>city Tarragona travel guide</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Antoni Gaudi</td>
<td>Incomparable man, Place, Master architect, Iconic</td>
<td>Buildings, City, Medical Structure, Office</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>structures, WH Testserver:</td>
<td>buildings, Office buildings, Public Place, etc.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Barcelona, Important buildings, etc.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tarragona Amphitheatre</td>
<td>Roman amphitheatre</td>
<td>Monument</td>
<td>Monument</td>
</tr>
<tr>
<td>Compostela</td>
<td>Long street, City, Town, Cultural center, Shrines,</td>
<td>City, Business centre, Shrine, Country, Place,</td>
<td>City</td>
</tr>
<tr>
<td></td>
<td>Davao, PDF file, etc.</td>
<td>Travel point, etc.</td>
<td></td>
</tr>
</tbody>
</table>

From these results it is possible to extract the following observations:

- In the Named Entity detection step (first column) the algorithm usually detects well-formed Named Entities related to the main topic of the web content, like Compostela, Tarragona Amphitheatre, Hispania Tarracronensis. Other Named Entities, which do not pertain to geographical places, but are strongly related with the main content of the web page like Antoni Gaudí are also detected. Some other chains like city Tarragona travel guide are also considered Named Entities although they are not; however as the main goal of this first stage is to extract as many candidates as possible to improve the recall, the fact of having non-annotable Named Entities is not a big problem, as they will be implicitly filtered in further stages. On the other hand, there is a little proportion of candidates to be annotated that are not detected by this step, but it is not very significant in comparison with the detected ones. The precision and the recall obtained in this step (if the results are compared with the expert’s ones) are 81.21\% and 40.76\% respectively, where the precision is high (good to obtain further reliable results).

---


\(^9\) [http://212.119.9.180/Ontologies/0.3/space.owl](http://212.119.9.180/Ontologies/0.3/space.owl)
• The class candidates detection step (second column) is quite effective (over 60-70% of the candidates are strongly related with one ontology class). It collects an enough number of class candidates for each Named Entity. For a Named Entity like Compostela we obtain class candidates like city, town, cultural center, etc., which are good class candidates, but we also obtain others like PDF file, which are not good class candidates.

• In the last step, the ontology class assignment (third and fourth columns) is able to select the corresponding class, as some of the classes obtained in the previous step are generally highly related with the ones in the ontology, like Roman amphitheatre with Monument. On the other hand, in the cases that the class candidates are not directly expressed in the ontology it obtains quite correct relationships, mainly thanks to the robustness of web-scale statistics. At this point, it is also interesting to note that the incorrect Named Entities (they are not Named Entities or are not related with the ontology) are filtered by the fact that no class candidates are found, like the case of city Tarragona travel guide or the case of Antoni Gaudi that is a Named Entity but it is not related with the ontology used, as it is an artist. In this class assignment step, the algorithm has obtained a precision of 69.93% and a recall of 69.28%.

5. Conclusions and further work

Up to this point, our algorithm gives promising results in terms of annotation quality without spending a big amount of resources. The combination of well-known techniques in the Named Entity detection phase gives well-defined Named Entities, thus it is possible to extract better class candidates. The class candidates obtained are, in fact, well related with the Named Entities, on the one hand because the Named Entities are good, and on the other hand because the text patterns are a powerful technique to obtain this knowledge. However, in the experimental results we have observed that some of the patterns (like the “is a”, “and other”, “or other” ones) are more precise and useful than others. The use of dictionaries, like WordNet, is also a good solution to reduce the total number of queries to web search engines, which is the slowest part of the algorithm.

As the algorithm is currently under development, as future work, it will be interesting to evaluate the system’s precision and performance using different WordNet measures, and comparing the results obtained. Another point is to study how to reduce the number of queries to web search engines, as they are slow and give to the system a dependency on external resources. Derived from the quality of the class candidates obtained with the different patterns it will be interesting to have a larger set of text patterns to obtain the class candidates and, depending on the contents, use the most appropriate ones. Reducing this set has some advantages, as on the one hand we will decrease drastically the number of queries and the annotation time and, on the other hand, the data sparseness problems derived from the reduction of this set are solved due to the size of the Web. Finally, we also would like to subdue our algorithm under an intense evaluation process. This will give us a deep knowledge of the algorithm’s performance in terms of precision and recall.
References