Automatic Expansion of DBpedia Exploiting Wikipedia Cross-Language Information

Alessio Palmero Aprosio[1,2], Claudio Giuliano[1], and Alberto Lavelli[1]

[1] Fondazione Bruno Kessler, Via Sommarive 18, I-38123 Trento
{aprosio,giuliano,lavelli}@fbk.eu
[2] Università degli Studi di Milano, Via Comelico 39/41, I-20135 Milano

Abstract. DBpedia is a project aiming to represent Wikipedia content in RDF triples. It plays a central role in the Semantic Web, due to the large and growing number of resources linked to it. Nowadays, only 1.7M Wikipedia pages are deeply classified in the DBpedia ontology, although the English Wikipedia contains almost 4M pages, showing a clear problem of coverage. In other languages (like French and Spanish) this coverage is even lower. The objective of this paper is to define a methodology to increase the coverage of DBpedia in different languages. The major problems that we have to solve concern the high number of classes involved in the DBpedia ontology and the lack of coverage for some classes in certain languages. In order to deal with these problems, we first extend the population of the classes for the different languages by connecting the corresponding Wikipedia pages through cross-language links. Then, we train a supervised classifier using this extended set as training data. We evaluated our system using a manually annotated test set, demonstrating that our approach can add more than 1M new entities to DBpedia with high precision (90%) and recall (50%). The resulting resource is available through a SPARQL endpoint and a downloadable package.

1 Introduction

The need of structured information from the Web has led to the release of several large-scale knowledge bases (KB) in the last years. Most of them have been populated using Wikipedia as primary data source. The online encyclopedia represents a practical choice, as it is freely available, big enough to cover a large part of human knowledge, and populated by about 100,000 active contributors, therefore the information it contains represents a good approximation of what people need and wish to know. Some relevant examples include FreeBase,[3] DBpedia,[4] and Yago,[5] created using various techniques that range from crowd sourcing to handcrafted rules.

[3] http://www.freebase.com/
[4] http://dbpedia.org/About
[5] http://www.mpi-inf.mpg.de/yago-naga/yago/

We are particularly interested in DBpedia as it plays a central role in the development of the Semantic Web. The large and growing number of resources linked to it makes DBpedia one of the central interlinking hubs of the emerging Web of Data. First, the DBpedia project develops and mantains an ontology, available for download in OWL format. Then, this ontology is populated using a rule-based semi-automatic approach that relies on Wikipedia *infoboxes*, a set of *subject-attribute-value* triples that represents a summary of some unifying aspect that the Wikipedia articles share. For example, biographical articles typically have a specific infobox (`Persondata` in the English Wikipedia) containing information such as *name*, *date of birth*, *nationality*, *activity*, etc. Specifically, the DBpedia project releases an extraction framework used to extract the structured information contained in the infoboxes and to convert it in triples. Moreover, crowd sourcing is used to map infoboxes and infobox attributes to the classes and properties of the DBpedia ontology, respectively. Finally, if an infobox is mapped to a DBpedia class, all Wikipedia articles containing such infobox are added to the class. As the number of required mappings is extremely large, the whole process follows an approach based on the frequency of the infoboxes and infobox attributes. Most frequent items are mapped first. This guarantees a good coverage because infoboxes are distributed according the Zipf's law. Therefore, despite the number of mappings is small, a large number of articles have been added to the ontology. At the time of starting the experiments, there are 360 mappings available for the English DBpedia, covering around 1.7M entities, against almost 4M articles in Wikipedia. The remaining pages are automatically mapped to the trivial top-level class `owl:Thing`. Hereafter, when we speak about coverage, we will always refer to classes different from `owl:Thing`. The Italian chapter has only 50 mappings, but covering more than 600K pages (out of around 1M articles in the corresponding Wikipedia), because some infoboxes cover highy populated classes, like `Person` and `Place`. The French and Spanish chapters, differently, contain around 15K pages each, with 70 and 100 mappings respectively. Finally, the resulting KB is made available as Linked Data,[6] and via DBpedia's main SPARQL endpoint.[7]

Unfortunately, there is a lot of variability in the names used for infoboxes and infobox attributes. Thus, it often happens that two or more infoboxes might be mapped to the same class, but none of them is included in DBpedia because their individual frequency is too small. Moreover, the DBpedia ontology often has classes that do not have a corresponding Wikipedia infobox. For example, the class `Actor` does not have a generic infobox in the English Wikipedia. However, Wikipedia provides some very specific infoboxes mapped to subclasses of `Actor`, such as `Chinese-language_singer_and_actor`. In this way, Bruce Lee is present in the database as an `Actor`, while other very famous actors like Clint Eastwood and Brad Pitt are not, clearly an undesirable result. Finally, some articles do not have an infobox, even if Wikipedia provides one for the purpose. This may

[6] `http://wiki.dbpedia.org/Downloads`
[7] `http://dbpedia.org/sparql`

happen because the user who writes that article does not know how to specify it, or simply does not know that infoboxes exist.

At the early stages of the project, the construction of DBpedia was solely based on the English Wikipedia. More recently, other contributors around the world have joined the project to create localized and interconnected versions of the resource. The goal is to populate the same ontology used in the English project, using articles from editions of Wikipedia in different languages. At the time of writing, there are 16 different localized versions of DBpedia. The inclusion of more languages has widened the problem of coverage. As each edition of Wikipedia is managed by different groups of volunteers with different guidelines, the DBpedia leading idea to semi-automatically populate the ontology by mapping infoboxes to classes does not work properly in some cases. For example, in the Italian DBpedia, the `Cyclist` category is empty, simply because the Italian edition of Wikipedia has a more generic `Sportivo` (sportsman) infobox, evidently considered adequate by the Italian contributors. This is convenient because one can assign a lot of pages to a class with only a single mapping, but cannot identify a more specific class. Besides the `Cyclist` class, also `Actor`, `Writer` and `Scientist` are empty in the Italian DBpedia, for the same reason. Other languages have similar problems: there are no entities for `Politician` in French and German, for `Plant` in Spanish, and so on.

In this paper, we address the problem of populating the DBpedia ontology, that has 359 classes. We propose an automatic two-stage approach that exploits Wikipedia cross-language links to extend the DBpedia coverage in different languages. First, the cross-language links are used to add Wikipedia articles not present in the DBpedia for one language but present in others. In the above example, those cyclists in the Italian Wikipedia having a cross-language link to an English article already present in the English DBpedia can be automatically added to the Italian DBpedia. Thanks to this first step, we increased the DBpedia coverage on Wikipedia articles by around 60% on the six languages considered in our experiments (English, Italian, German, French, Spanish, and Portuguese). The relative error of cross-lingual links in Wikipedia is very small, so we asses that the precision of the first phase is almost 100% [11].

Second, we further boost the coverage by training a supervised kernel-based classifier using both the articles already present in DBpedia and the ones extracted in the first stage, and then classify those articles for which cross-language links do not exist. Experiments have been performed on a dataset of 400 articles manually annotated by the authors. Starting from 5.6M total entities extracted from Wikipedia in the six languages, around 2.2M are added using the first step. We show that our approach further increases the coverage of the DBpedia with high accuracy. Our algorithm can be tuned to have different tradeoffs between precision and recall. The resulting resource contains a total of nearly 4M entities, 1.7M of them not included in the original DBpedia for the six languages considered for the experiment.

2 Entity Representation

The goal of our research is to assign novel entities to DBpedia classes requiring no additional human supervision. Specifically, we consider those entities not already present in DBpedia for which there exists at least a Wikipedia article, no matter in which language. The ontology population task is cast as a machine-learning classification problem, where entities already present in DBpedia (again, no matter in which language the corresponding Wikipedia articles are available) are used to train a state-of-the-art classifier that assigns novel entities to the most specific class in the DBpedia ontology.

Our approach exploits the Wikipedia cross-language links to represent each entity with features extracted from the corresponding articles in different languages. This novel contribution is supported by the observation that different Wikipedia communities tend to structure the articles in a slightly different way. As already reported in Section 1, English and Italian Wikipedia have an infobox for biographies (`PersonData` and `Bio`, respectively), while Spanish and French do not. DBpedia offers the triple set of cross-language links, but the information stored in one language is not automatically trasferred on other ones.

Formally, we proceed as follows to automatically derive the set of entities $\mathcal{E}$, also used to build the training set. Let $\mathcal{L}$ be the set of languages available in Wikipedia, we first build a matrix E where the i-th row represents an entity $e_i \in \mathcal{E}$ and j-th column refers to the corresponding language $l_j \in \mathcal{L}$. The cross-language links are used to automatically align on the same row all Wikipedia articles that describe the same entity. The element $E_{i,j}$ of this matrix is *null* if a Wikipedia article describing the entity e_i does not exist in l_j. An instance in our machine learning problem is therefore represented as a row vector e_i where each j-th element is a Wikipedia article in language l_j. Figure 1 shows a portion of the entity matrix.

Fig. 1: A portion of the entity matrix

en	de	it	...
Xolile Yawa	Xolile Yawa	*null*	...
The Locket	*null*	Il segreto del medaglione	...
Barack Obama	Barack Obama	Barack Obama	...
null	*null*	Giorgio Dendi	...
...	...	...	...

3 Kernels for Entity Classification

The strategy adopted by kernel methods [20,19] consists of splitting the learning problem in two parts. They first embed the input data in a suitable feature

space, and then use a linear algorithm (e.g., the perceptron) to discover nonlinear patterns in the input space. Typically, the mapping is performed implicitly by a so-called *kernel function*. The kernel function is a similarity measure between the input data that depends exclusively on the specific data type and domain. A typical similarity function is the inner product between feature vectors. Characterizing the similarity of the inputs plays a crucial role in determining the success or failure of the learning algorithm, and it is one of the central questions in the field of machine learning.

Formally, the kernel is a function $k : X \times X \rightarrow \mathbb{R}$ that takes as input two data objects (e.g., vectors, texts, parse trees) and outputs a real number characterizing their similarity, with the property that the function is symmetric and positive semi-definite. That is, for all x_1, $x_2 \in X$, it satisfies

$$k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle,$$

where ϕ is an explicit mapping from X to an (inner product) feature space $\mathcal{F}$.

In the remainder of this section, we define and combine different kernel functions that calculate the pairwise similarity between entities using their corresponding Wikipedia articles as source of information. They are the only domain specific elements of our classification system, while the learning algorithm is a general purpose component. Many classifiers can be used with kernels, we use k-nearest neighbor (k-nn).

3.1 Bag-of-features Kernels

The simplest method to calculate the similarity between two entities is to compute the inner product of their vector representation in the vector space model (VSM). Formally, we define a space of dimensionality N in which each dimension is associated with one feature, and the entity e is represented in the language $l_j \in \mathcal{L}$ by a row vector

$$\phi_j(e_i) = (w(f_1, E_{i,j}), w(f_2, E_{i,j}), \ldots, w(f_N, E_{i,j})), \tag{1}$$

where the function $w(f_k, E_{i,j})$ records whether a particular feature f_k is active in the Wikipedia article $E_{i,j}$. Using this representation we define the *bag-of-features kernel* between entities as

$$K_F(e_1, e_2) = \sum_{j=1}^{|\mathcal{L}|} \langle \phi_j(e_1), \phi_j(e_2) \rangle, \tag{2}$$

Notice that this kernel computes the similarity between e_1 and e_2 as the sum of their similarities in those languages for which Wikipedia articles exist. Based on this general formulation, we define 4 basic kernel functions as follows.

Bag-of-templates Kernel. To define the similarity between pairs of entities, we count how many occurrences of templates their corresponding Wikipedia articles in a specific language share. Templates are commonly used for boilerplate

messages, standard warnings or notices, infoboxes, navigational boxes and similar purposes. In our experiments, we take into consideration solely the infoboxes (Section 4.1 describes the set of heuristics used to extract the infoboxes). The *Bag-of-templates kernel* (K_T) is defined as in Equation (2), where the function $w(f_k, E_{i,j})$ in Equation (1) is a binary function that records whether a particular infobox f_k is used in the Wikipedia article $E_{i,j}$.

Bag-of-categories Kernel. Wikipedia categories are intended to group together articles on similar subjects and have proven useful in text classification [22], ontology learning [15], and ontology population [21]. The *bag-of-categories kernel* (K_C) is defined as in Equation (2) where the function $w(f_k, E_{i,j})$ in Equation (1) is a binary function that records whether a particular category f_k is used in the Wikipedia article $E_{i,j}$.

Bag-of-sections Kernel. Wikipedia articles are structured in several sections that might provide relevant cues for classification. For example, biographical articles typically include sections like *Early life*, *Career*, and *Personal life*; while articles referring to cities usually include sections like *Places of interest*, *Demographic evolution*, and *Administration*. The *bag-of-sections kernel* (K_C) is defined as in Equation (2) where the function $w(f_k, E_{i,j})$ in Equation (1) is a binary function that records whether a particular section name f_k is used in the Wikipedia article $E_{i,j}$.

Bag-of-words Kernel. The use of infoboxes, categories, and sections ensures highly accurate classification, however it produces extremely sparse feature spaces that compromises the recall. To overcome this problem, we also exploit content words of the text article as additional sources of information. The *bag-of-words kernel* (K_W) is defined as in Equation (2) where the function $w(f_k, E_{i,j})$ in Equation (1) is the standard *term frequency–inverse document frequency* (tf $\times$ idf) of the word f_k in the Wikipedia article $E_{i,j}$.

3.2 Latent Semantic Kernel

Given that the bag-of-words representation does not deal well with lexical variability, in the following we introduce the latent semantic kernels and show how to define an effective semantic VSM using (unlabeled) external knowledge. It has been shown that semantic information is fundamental for improving the accuracy and reducing the amount of training data in many natural language tasks, including fine-grained classification of named entities [4,7], question classification [12], text categorization [9], word sense disambiguation [10].

In the context of kernel methods, semantic information can be integrated considering linear transformations of the type $\tilde{\phi}_j(c_t) = \phi_j(c_t)\mathbf{S}$, where $\mathbf{S}$ is a $N \times k$ matrix [20]. The matrix $\mathbf{S}$ can be rewritten as $\mathbf{S} = \mathbf{WP}$, where $\mathbf{W}$ is a diagonal matrix determining the word weights, while $\mathbf{P}$ is the *word proximity*

matrix capturing the semantic relations between words. The proximity matrix $\mathbf{P}$ can be defined by setting non-zero entries between those words whose semantic relation is inferred from an external source of domain knowledge. The *semantic kernel* takes the general form

$$\tilde{k_j}(e_1, e_2) = \phi_j(e_1)\mathbf{S}\mathbf{S}'\phi_j(e_2)' = \tilde{\phi_j}(e_1)\tilde{\phi_j}(e_2)'. \tag{3}$$

It follows directly from the explicit construction that Equation (3) defines a valid kernel.

To define the proximity matrix for the latent semantic kernel, we look at co-occurrence information in a (large) corpus. Two words are considered semantically related if they frequently co-occur in the same texts. We use singular valued decomposition (SVD) to automatically derive the proximity matrix $\mathbf{\Pi}$ from a corpus, represented by its term-by-document matrix $\mathbf{D}$, where the $\mathbf{D}_{i,j}$ entry gives the frequency of term p_i in document d_t.[8] SVD decomposes the term-by-document matrix $\mathbf{D}$ into three matrixes $\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}'$, where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices (i.e., $\mathbf{U}'\mathbf{U} = \mathbf{I}$ and $\mathbf{V}'\mathbf{V} = \mathbf{I}$) whose columns are the eigenvectors of $\mathbf{D}\mathbf{D}'$ and $\mathbf{D}'\mathbf{D}$ respectively, and $\mathbf{\Sigma}$ is the diagonal matrix containing the singular values of $\mathbf{D}$. Under this setting, we define the proximity matrix $\mathbf{\Pi}$ as follows:

$$\mathbf{\Pi} = \mathbf{U}_k\mathbf{\Sigma}_k,$$

where $\mathbf{U}_k$ is the matrix containing the first k columns of $\mathbf{U}$ and k is the dimensionality of the latent semantic space and can be fixed in advance. By using a small number of dimensions, we can define a very compact representation of the proximity matrix and, consequently, reduce the memory requirements while preserving most of the information.

The matrix $\mathbf{\Pi}$ is used to define a linear transformation $\pi_j : \mathbb{R}^N \to \mathbb{R}^k$, that maps the vector $\phi_j(e_t)$, represented in the standard VSM, into the vector $\tilde{\phi_j}(e_t)$ in the latent semantic space. Formally, π_j is defined as follows

$$\pi_j(\phi_j(e_t)) = \phi_j(e_t)(\mathbf{W}\mathbf{\Pi}) = \tilde{\phi_j}(e_t), \tag{4}$$

where $\phi_j(e_t)$ is a row vector, $\mathbf{W}$ is a $N \times N$ diagonal matrix determining the word weights such that $\mathbf{W}_{i,i} = \log(\text{idf}(w_i))$, where $\text{idf}(w_i)$ is the *inverse document frequency* of w_i.

Finally, the *latent semantic kernel* is explicitly defined as follows

$$K_L(e_1, e_2) = \sum_{j=1}^{|\mathcal{L}|} \langle \pi_j(\phi_j(e_1)), \pi_j(\phi_j(e_2)) \rangle,$$

where ϕ_j is the mapping defined in Equation (1) and π_j is the linear transformation defined in Equation (4) in language $l_j \in \mathcal{L}$. Note that we have used a series of successive mappings each of which adds some further improvement to the entity representation.

[8] SVD has been first applied to perform latent semantic analysis of terms and latent semantic indexing of documents in large corpora by [3].

Table 1: Versions of DBpedia and Wikipedia used for our tests

	English	Italian	German	French	Spanish	Portuguese
Wikipedia	2012-10-01	2012-09-21	2012-10-09	2012-10-07	2012-09-27	2012-10-06
DBpedia	2012-06-04	2012-10-12	2012-06-04	2012-06-04	2012-06-04	2012-06-04

3.3 Composite Kernel

Having defined all the basic kernels, representing different characteristics of entity descriptions, we finally define the composite kernel as

$$K_{\text{COMBO}}(e_1, e_2) = \sum_{n=1} \frac{K_n(e_1, e_2)}{\sqrt{K_n(e_1, e_2) K_n(e_1, e_2)}}, \tag{5}$$

where K_n is a valid basic kernel. The individual kernels are normalized. This plays an important role in allowing us to integrate information from heterogeneous feature spaces. It follows directly from the explicit construction of the feature space and from closure properties of kernels that the composite kernel is a valid kernel.

4 Experiments

In this section, we evaluate different setups on the task of DBpedia expansion for six languages (English, Italian, German, French, Spanish, and Portuguese). The evaluation only concerns the second stage of our approach, because the first stage has precision almost 100% (see Section 4.1).

4.1 Pre-processing Wikipedia and DBpedia

Our experiments and results refer to the versions of Wikipedia and DBpedia available when this work started in mid October 2012. Table 1 lists the dumps used.

Wikipedia. We parsed the dump files to extract information about each single article and we built the matrix E using cross-language links (see Section 2). We manually check the accuracy of these links on 100 random pages: all of them were correct, so we can assume that the precision of this step is 100%. The matrix E build upon six languages (English, Italian, German, French, Spanish, and Portuguese) contains 5,626,515 entities.

We use a particular strategy for the template extraction, as we only want infoboxes for our classification. As Wikipedia does not provide a simple way to select only such type of templates, we implemented a simple rule-based handcrafted classifier[9] to filter templates that (i) appear less than 50 times, (ii) appear

[9] Looking at the template name for keywords such as `Infobox` is not a good strategy, as there is plenty of infobox templates that do not follow this rule.

Table 2: Total number of pages in Wikipedia, in DBpedia, and in DBpedia after using Wikipedia cross-language links. Quantities in the last row represent, for each language, the number of pages not included in DBpedia in any language considered

	Matrix E	EN	IT	DE	FR	ES	PT
Wikipedia	5,626,515	3,932,148	924,544	1,325,792	1,251,585	953,848	740,585
DBpedia	-	1,716,555	607,842	205,903	15,463	15,987	226,497
DBpedia CL	2,193,742	1,902,585	652,395	482,747	518,874	419,168	430,603
Not classified	3,432,773	2,029,563	272,149	843,045	732,711	534,680	309,982

mostly more than once in a page, (iii) are not written in a key/value form, and (iv) are written on multiple lines. In this way, we filter more than 90% of the templates, obtaining an average of a couple of templates for each page.

DBpedia. Starting from DBpedia dumps, we created a mapping that combines the entities in E with the ontology class(es) they belong to. Using entities instead of Wikipedia pages allows us to automatically extend and improve the DBpedia coverage. For instance, *Michael Jackson* is classified as a `Person` in the Italian and German DBpedia, an `Artist` in the English DBpedia and a `MusicalArtist` in the Spanish DBpedia. The most specific class is the last one, so the *entity* Michael Jackson becomes `MusicalArtist` in every language. The final mapping contains 2,193,742 entities: comparing this figure with the size of the matrix E, this means that there are around 3,432,773 entities in Wikipedia that are not classified in DBpedia. In our experiments we always refer to this set for the classification part that makes use of kernel methods. Data concerning the enriched DBpedia is shown in Table 2.

4.2 Benchmark

Experiments are carried out on a benchmark extracted from the entity matrix introduced in Section 2. Specifically, the data set contains 400 randomly extracted entities not already present in DBpedia in any language. The data set is split in development set (100 entities) and test set (300 entities). All entities have been annotated with the most specific available class in the version 3.8 of the DBpedia ontology by one of the authors of this paper. 50 more entities have been annotated by three different annotators, resulting in an inter-agreement of 78% (Fleiss' kappa measure, see [5]). An additional `Unknown` class has been introduced to annotate those entities that cannot be assigned to any class in the ontology. When an entity is assigned to a class, it is also implicitly assigned to all its super-classes. For instance, classifying *Michael Jackson* as a `MusicalArtist` we implicitly classify him as `Artist`, `Person` and `Agent`.

The evaluation is performed as proposed by [13] for a similar hierarchical categorization task. In the example above, classifying *Michael Jackson* as an `Athlete`, we obtain a false positive for this wrong classified class, two false negatives for missing classes `MusicalArtist` and `Artist`, and two true positives for `Person` and `Agent`.

4.3 Latent Semantic Models

For each language, we derive the proximity matrix Π (Section 3) from the 200,000 most visited Wikipedia articles. After removing terms that occur less than 5 times, the resulting dictionaries contain about 300,000 terms. We use the SVDLIBC package[10] to compute the SVD, truncated to 100 dimensions.

4.4 Learning Algorithm

We use a k-nn classifier[11] to classify novel entities into the DBpedia ontology. The optimization of the parameter k is performed on the development set, and $k = 10$ results as the best choice, because it maximizes the F_1 value. Entities are classified by a majority vote of their neighbors. To change the tradeoff between precision and recall, we set the minimum number of votes z ($1 \leqslant z \leqslant k$) a class needs to obtain to be assigned. The algorithm has maximum precision with $z = k$, maximum recall with $z = 1$, and maximum F_1 with $z = 8$.

To train the classifier, we randomly select 100,000 entities from the matrix E included in DBpedia. Each entity is then labelled according to the corresponding DBpedia class.

4.5 Classification Schemas

We compare three alternative training and classification schemas.

Bottom-up. A single training and classification step is performed. k-nn is trained using entities annotated with the most specific classes in DBpedia. In classification, an entity is annotated with the finer-grained class c if c receives $v_c \geqslant z$ votes; `Unknown` otherwise.[12] Note that the algorithm also considers the super-classes of c: if a fine-grained class cannot be assigned with a given confidence level z, it could return a more generic one s ($s \subseteq c$) such that $v_s \geqslant z$. For instance, if $z = 10$ and the 10 votes are divided 5 to `Astronaut` and 5 to `Scientist`, our system answers `Unknown` because none of the classes obtains 10 votes. However, ascending the ontology, we find that the class `Person` receives 10 votes, as both `Astronaut` and `Scientist` belong to it. The system then classifies it as `Person`, instead of `Unknown`. In case this process does not find any class at any level with a sufficient number of votes, the `Unknown` answer is given.

[10] `http://tedlab.mit.edu/~dr/svdlibc/`

[11] During the first experiments, we used our algorithms with two test classes: `Person` and `Place`. In this phase, Support Vector Machine (SVM) produced very good results. When we applied our approach to the entire DBpedia ontology (359 classes), SVM performance dramatically dropped.

[12] Assigning the class `Unknown` is equivalent to abstention.

Fig. 2: Precision/recall curve of the system

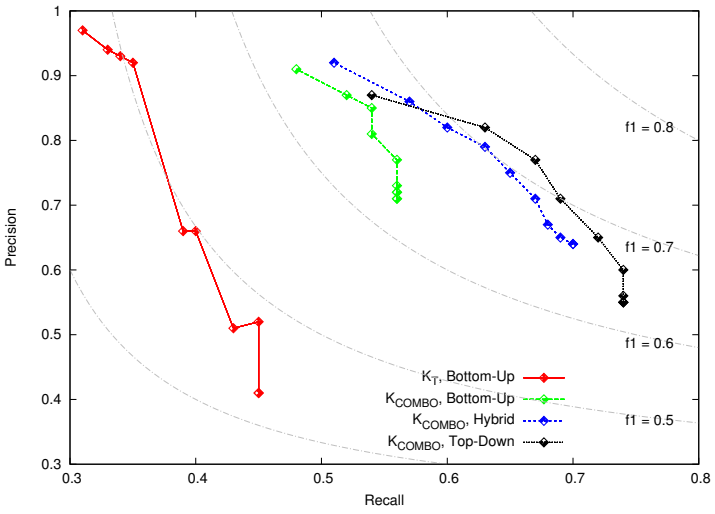


Table 3: Results of the most frequent class baseline (MF), the basic kernels (see Section 3.1) and the composite kernel K_{COMBO}, using $z = 10$

	MF	K_T	K_C	K_S	K_W	K_L	K_{COMBO}
Precision	0.35	0.97	0.90	0.94	0.81	0.84	0.91
Recall	0.38	0.31	0.40	0.16	0.22	0.41	0.48
F_1	0.31	0.47	0.55	0.27	0.34	0.55	0.63

Top-down. Multiple training and classification steps are performed. k-nn is trained using entities annotated with the most generic classes in DBpedia (ontology top-level). In classification, an entity is annotated with a generic class c if it receives $v_c \geqslant z$ votes; `Unknown` otherwise. The procedure is recursively repeated on all subtrees of the ontology using the previous classification to limit the number of classes to consider.

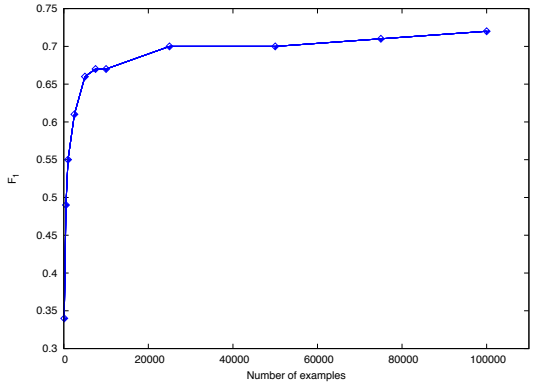
Hybrid. This variant consists in first training a k-nn as defined in the Bottom-up schema. Then, a set of specialized k-nns are trained for the most populated classes, such as, `Person`, `Organisation`, `Place`, `Work`, etc. In classification, let P be one of these classes, the Bottom-up schema is applied first. Then, if an entity is annotated with the class c such that $c \in P$, then a specialized k-nn is applied.

4.6 Results

First, we investigate the contribution of the kernel combination (Section 3) and then the one of the different training and classification schemas (Section 4.5).

Table 3 reports the results of the most frequent class baseline, the basic kernels (K_T, K_C, K_S, K_W, and K_L), and the composite kernel K_{COMBO}. The

Fig. 3: Learning curve of the system



experimental results show that the composite kernel K_{COMBO} significantly outperforms the basic kernels. We use approximate randomization [16] to assess the statistical significance between the obtained results (p-value $= 0.05$).

Figure 2 shows the precision/recall curves obtained by varying the parameter z. We also draw, in grey in the background, the contour lines joining points with the same F_1, so that one can quickly visualize this value. Four different setups are compared in order to determine the one that produces the best tradeoff between precision and recall.

$\mathbf{K_T}$, Bottom-up uses only the template information (as in the DBpedia framework) and the Bottom-up schema, obtaining the maximum precision of 97% at the expense of low recall of 31% ($z = 10$).

$\mathbf{K_{COMBO}}$, Bottom-up uses all the sources of information and the Bottom-up schema, obtaining a significant improvement in recall (48%) preserving a high precision of 91% ($z = 10$).

$\mathbf{K_{COMBO}}$, Hybrid uses all the sources of information and the Hybrid schema, obtaining a further improvement of precision (92%) and recall (51%).

$\mathbf{K_{COMBO}}$, Top-down uses all the sources of information and the Top-down schema, obtaining the maximum recall (54%), however the precision (87%) is significantly lower than the one obtained in the other experiments.

Figure 3 shows the learning curve of the system in term of F_1 in the configuration that maximizes the F_1 score (in our experiments, this happens in all configurations, when $z = 8$).

Finally, we perform some preliminary error analysis. Errors mostly depend on the following factors: (i) the Wikipedia article is too short; (ii) an appropriate class for the entity does not exist (this often happens with common nouns); (iii) some Wikipedia pages represent lists (for example, `Liste_des_conseillers...`) and our system often classifies them as the objects listed (in the example, `Person`); (iv) nesting of DBpedia classes is not optimal (for example, `Astronaut`

and `Scientist` are disjoint classes). The most common factor is (iii), as it is the cause of more than half of the errors in the experiments on the test set.

5 Related Work

The DBpedia project [1], started in 2007, manually creates an ontology starting from Wikipedia infobox templates. Nowadays, the English version covers around 1.7M Wikipedia pages, although the English Wikipedia contains almost 4M pages. Other languages suffer from an even lower coverage (see Table 2).

Differently, Yago [21], another similar project also started in 2007, aims to extract and map entities from Wikipedia using categories (for fine-grained classes) and WordNet (for upper-level classes). Its coverage is higher, but it is monolingual and its ontology contains thousands of hundreds of classes: it may be difficult to use it in practical applications.

There are also other projects aiming to extract Wikipedia entity types boostrapping information contained in the categories. For example, [17] uses extracted datatypes to train a name entity recogniser, while [15] investigates Wikipedia categories and automatically cleans them.

The tool presented in [6], *Tipalo*, identifies the most appropriate class of a Wikipedia article by interpreting its page abstract using natural language processing tools and resources. In this context, only English Wikipedia is considered, as this classifier cannot be easily adapted to other languages.

Similarly, [18] only considers the English DBpedia and therefore does not take advantages from inter-language links. In addition, there is some manual effort to classify biographies (using tokens from categories), that leads to very good results, but is not automatically portable to other languages; again linguistic tools are used to extract the definition from the first sentence.

The approach presented in [7] classifies people on an excerpt of the WordNet ontology, using kernel functions that implicitly map entities, represented by aggregating all contexts in which they occur, into a latent semantic space derived from Wikipedia. This approach queries online the name of the entity to collect contextual information. We specialize this approach to Wikipedia, that is easily to download and store locally.

[8] proposes an unsupervised approach based on lexical entailment, consisting in assigning an entity to the category whose lexicalization can be replaced with its occurrences in a corpus preserving the meaning.

6 Conclusions and Future Work

We have proposed a two-stage approach that automatically extends the coverage of DBpedia with respect to Wikipedia articles. We have first extended the population of DBpedia using cross-language links, and then used this extended population as training data to classify the remaining pages using a kernel-based supervised method. The experiments have been evaluated on a manually annotated test set containing 400 Wikipedia pages, resulting in high precision and

recall, with different tradeoffs of these values depending on the configuration of the algorithm. The resulting resource is available both as a download package and a SPARQL endpoint at `http://www.airpedia.org/`.

DBpedia also maps entity properties, such as `BirthDate` and `birthPlace` for `Person`, `director` for `Film`, and so on. We are currently working to deal with this problem, using natural language processing tools to find the correct relation in the article text. This can be seen as a relation extraction task, and one of the most reliable approaches to tackle this problem (starting from a large available knowledge base) is *distant supervision* [14]. This paradigm has been successfully used for pattern extraction [23] and question answering [2]. Moreover, we want to deal with entities belonging to more than one class. Some entities in DBpedia are correctly classified in multiple classes. For example, Madonna is a singer (`MusicalArtist`) and an actress (`Actor`).

Finally, we will investigate how to build a new ontology based on Wikipedia categories together with templates, using the results produced by our system.

References

1. Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: a nucleus for a web of open data. In *Proceedings of the 6th international Semantic Web Conference and 2nd Asian Semantic Web Conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.
2. Elena Cabrio, Julien Cojan, Alessio Palmero Aprosio, Bernardo Magnini, Alberto Lavelli, and Fabien Gandon. QAKiS: an open domain QA system based on relational patterns. In Birte Glimm and David Huynh, editors, *International Semantic Web Conference (Posters & Demos)*, volume 914 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2012.
3. Scott C. Deerwester, Susan T. Dumais, Thoms K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
4. Michael Fleischman and Eduard Hovy. Fine grained classification of named entities. In *Proceedings of the 19th International Conference on Computational Linguistics*, Taipei, Taiwan, 2002.
5. Joseph L. Fleiss. Measuring Nominal Scale Agreement Among Many Raters. *Psychological Bulletin*, 76(5):378–382, 1971.
6. Aldo Gangemi, Andrea Giovanni Nuzzolese, Valentina Presutti, Francesco Draicchio, Alberto Musetti, and Paolo Ciancarini. Automatic typing of DBpedia entities. In Philippe Cudré-Mauroux, Jeff Heflin, Evren Sirin, Tania Tudorache, Jérôme Euzenat, Manfred Hauswirth, Josiane Xavier Parreira, Jim Hendler, Guus Schreiber, Abraham Bernstein, and Eva Blomqvist, editors, *International Semantic Web Conference (1)*, volume 7649 of *Lecture Notes in Computer Science*, pages 65–81. Springer, 2012.
7. Claudio Giuliano. Fine-grained classification of named entities exploiting latent semantic kernels. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, pages 201–209, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

8. Claudio Giuliano and Alfio Gliozzo. Instance based lexical entailment for ontology population. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 248–256, Prague, Czech Republic, June 2007. Association for Computational Linguistics.
9. Alfio Gliozzo and Carlo Strapparava. Domain kernels for text categorization. In *Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 56–63, Ann Arbor, Michigan, June 2005.
10. Alfio Massimiliano Gliozzo, Claudio Giuliano, and Carlo Strapparava. Domain kernels for word sense disambiguation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 403–410, Ann Arbor, Michigan, June 2005.
11. Dimitris Kontokostas, Charalampos Bratsas, Sören Auer, Sebastian Hellmann, Ioannis Antoniou, and George Metakides. Internationalization of Linked Data: The case of the Greek DBpedia edition. *Web Semantics: Science, Services and Agents on the World Wide Web*, 15(0):51 – 61, 2012.
12. Xin Li and Dan Roth. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249, 2005.
13. I. Dan Melamed and Philip Resnik. Tagger evaluation given hierarchical tag sets. *Computers and the Humanities*, pages 79–84, 2000.
14. Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.
15. Vivi Nastase and Michael Strube. Decoding Wikipedia categories for knowledge acquisition. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*, AAAI'08, pages 1219–1224. AAAI Press, 2008.
16. Eric W. Noreen. *Computer-Intensive Methods for Testing Hypotheses : An Introduction*. Wiley-Interscience, 1989.
17. Joel Nothman, James R. Curran, and Tara Murphy. Transforming Wikipedia into named entity training data. In *Proceedings of the Australasian Language Technology Workshop*, Hobart, Australia, 2008.
18. A. Pohl. Classifying the Wikipedia Articles into the OpenCyc Taxonomy. In *Proceedings of the Web of Linked Entities Workshop in conjuction with the 11th International Semantic Web Conference*, 2012.
19. B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, Cambridge, Massachusetts, 2002.
20. J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
21. Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.
22. Pu Wang, Jian Hu, Hua-Jun Zeng, and Zheng Chen. Using wikipedia knowledge to improve text classification. *Knowledge and Information Systems*, 19:265–281, 2009. 10.1007/s10115-008-0152-4.
23. Fei Wu and Daniel S. Weld. Open information extraction using wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 118–127, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.