Bootstrapping Domain Ontologies from Wikipedia: A Uniform Approach

Daniil Mirylenka and **Andrea Passerini**
University of Trento
Via Sommarive 9, 38123, Trento, Italy
{dmirylenka, passerini}@disi.unitn.it

Luciano Serafini
Fondazione Bruno Kessler
Via Sommarive 18, 38123, Trento, Italy
serafini@fbk.eu

Abstract

Building ontologies is a difficult task requiring skills in logics and ontological analysis. Domain experts usually reach as far as organizing a set of concepts into a hierarchy in which the semantics of the relations is under-specified. The categorization of Wikipedia is a huge concept hierarchy of this form, covering a broad range of areas. We propose an automatic method for bootstrapping domain ontologies from the categories of Wikipedia. The method first selects a subset of concepts that are relevant for a given domain. The relevant concepts are subsequently split into classes and individuals, and, finally, the relations between the concepts are classified into subclass_of, instance_of, part_of, and generic related_to. We evaluate our method by generating ontology skeletons for the domains of Computing and Music. The quality of the generated ontologies has been measured against manually built ground truth datasets of several hundred nodes.

1 Introduction

Building ontologies is a difficult task that requires expertise in the domain that is being modeled, as well as in logic and ontological analysis. Firstly, domain knowledge is necessary to decide "the scope and the boundaries of the ontology" [Iqbal *et al.*, 2013], that is, to separate the entities relevant to the modeled domain from the accessory elements that should not be included into the ontology. In the subsequent phase, domain expertise is needed to express the relations between the selected entities. The most important kind of relations are hierarchical relations, such as *meronomy*, and the relation between an entity and its type. A clear distinction between relation types requires additional competences in logics and ontological analysis. Domain experts tend to merge these relations into a generic `broader/narrower` relation, which results in the partially formalized knowledge resources such as classification schemes, lexicons, and thesauri.

These types of partially structured descriptions of the domain are widely used in the Semantic Web, spanning from global categorizations, such as that of Wikipedia, to domain-specific schemes, such as the ACM Computing Classification System, and providing great support for structural access to web resources. An example of a system that exploits such a resource is ScienScan [Mirylenka and Passerini, 2013], which provides structured access to computer science literature by navigating the Wikipedia category network. A fully developed formal representation of this structure would enhance these types of applications with the possibility of a more flexible semantic search and navigation [Osborne *et al.*, 2013]. For this purpose, we argue, it is worth facilitating the transformation of the informally structured knowledge representations into fully formalized ontologies.

In this paper we propose an automatic method based on the machine learning techniques for extracting domain *ontology skeletons* from the Wikipedia category hierarchy. We use the term *ontology skeleton* to indicate a basic version of an ontology that contains all the primitive concepts, the essential hierarchical relations between them, and some track of the remaining generic relations of unspecified type. An ontology skeleton is meant to be further refined in two ways: first, by providing corrections to the solutions proposed by the automatic algorithm, and second, by assigning more specific relation types to generic relations. The method first selects a subset of the categories that are relevant for the given domain. The relevant categories are then split into classes and individuals, and, finally, the relations between the categories are classified as either `subclass_of`, `instance_of`, `part_of`, or generic `related_to`.

We evaluate our method by generating ontology skeletons for the domains of `Computing` and `Music`. The quality of the generated ontologies has been measured against manually built ground truth datasets. The results suggest high quality in selecting the relevant nodes, discriminating classes from individuals, and identifying the `subclass_of` relation. The accuracy of identifying the `instance_of` relation is on par with the state of the art. The more difficult `part_of` relation between Wikipedia categories was never addressed in the literature, and our initial results need further improvement. The code, data, and experiments are available[1] online.

2 Problem statement

According to [Suárez-Figueroa *et al.*, 2012], our problem fits into the scenario of reusing non-ontological resources for building ontologies. Our resource, the categorization of

[1] https://github.com/anonymous-ijcai/dsw-ont-ijcai

Wikipedia[2], is represented by a hierarchy of labels used for organizing Wikipedia articles. With some exceptions, the categories follow the established naming conventions[3], such as:

- the names of the *topic categories* should be singular, normally corresponding to the name of a Wikipedia article;
- the names of the *set categories* should be plural;
- the meaning of a name should be independent of the way the category is connected to other categories.

As suggested above, there are two main kinds of categories: *topic categories*, named after a topic, and *set categories*, which are named after a class. An example of a topic category is `France`, which contains articles speaking about the whole France; an example of a set category is `Cities_in_France`.

With the exception of occasional cycles, Wikipedia categories are organized into a lattice. There is a top-level category, and all other categories have at least one parent. The semantics of the hierarchical relation is not specified, and individual relations may be of different ontological nature. The main types of relations are:

1. subset relation between the set categories, e.g. `Countries_in_Europe` ← `Baltic_countries`,
2. membership relation between a set category and a topic category, e.g. `Countries_in_Europe` ← `Albania`,
3. part-of relation, usually, between two topic categories, e.g. `Scandinavia` ← `Sweden`,
4. sub-topic relation between two topic categories, e.g. `European_culture` ← `European_folklore`,
5. other relations, whose nature may be specified by the category labels, e.g. `Europe` ← `Languages_of_Europe`.

Formalizing these relations in description logics requires:

The definition of the signature: For each set category, such as `Country_in_Europe`, one should introduce a class. For each topic category, such as `Sweden`, one should introduce an individual. Then, the relations between the entities should be declared, such as `part_of` and `subtopic_of`. New relational symbols should be introduced to formalize the relations implied by the category names, such as the relation `spoken_in` implied by the category `Language_of_Europe`.

The definition of the axioms: Finally, the sub-category relations between Wikipedia categories should be transformed into axioms of description logic, such as:

- `Baltic_countries` $\sqsubseteq$ `Country_in_Europe`,
- `part_of(Sweden, Scandinavia)`, etc..

In the rest of the paper we propose an automatic method for extracting domain ontology skeletons from the category network of Wikipedia. At the high level, the method consists of the following three steps: 1) selecting the relevant subset of categories, 2) transforming each category into a class or an individual, and 3) establishing the semantic relations between the categories. Our solution relies on the following simplifying assumptions: 1) relations `subtopic_of` and `part_of` are merged into a unique relation `part_of`, and 2) relations other than `part_of`, `instance_of`, and `subclass_of` are codified as a single generic relation `related_to`.

[2] http://en.wikipedia.org/wiki/Wikipedia:Topic_category

[3] http://en.wikipedia.org/wiki/Wikipedia:Category_names

3 Solution

Each of the three steps of our method—selecting the relevant categories, splitting them into classes and individuals, and classifying the relations—is cast into a binary classification problem. More precisely, the problem at the first step reduces to discriminating between relevant and irrelevant nodes, the next one—to discriminating between classes and individuals, and that at the last step—to discriminating between a specific relation (such as `subclass_of`) and the generic `related_to`. Solving the problem at each step requires providing manually annotated examples: for instance, those of relevant and irrelevant categories. The rest of the work is done by a machine learning algorithm. In the following sections we provide the detailed description of the three steps of the method.

3.1 Selecting the subgraph of relevant categories

In order to identify the relevant categories, we first select the most general category representing the domain of interest (henceforth referred to as "the root"). For the computer science domain, for instance, we choose the category `Computing`. The category selection algorithm is based on the following two observations. First, all relevant categories are descendants of the root with respect to the sub-category relations. Second, the categories further from the root are more likely to be irrelevant (following sub-category links, one can arrive from `Computing` to `Buddhism_in_China` in just 9 hops).

The algorithm (Algorithm 1) performs a breadth-first traversal of the category graph, starting from the root. For each category being visited, the decision is made (line 7), whether the category is relevant, and should be scheduled for recursive traversal. The decision is made by a trained classifier, based on the features of the category (discussed further). To ensure the termination of the algorithm, we set a limit `max_depth` on the maximum allowed traversal depth. Based on our experience, we claim that using `max_depth`=20 is safe, in the sense that any category that is further than 20 hops from the root is irrelevant. In practice much smaller value can be chosen, depending on the domain. For `Computing` we empirically discovered that `max_depth`=7 with high confidence.

Algorithm 1: Selection of the relevant categories.

```
   input  : root; max_depth
   output: relevant: the set of relevant categories
1  queue ← empty FIFO queue
2  visited ← {root}
3  relevant ← ∅
4  queue.push(root)
5  while queue is not empty :
6      category ← queue.pop()
7      if isRelevant(category) :
8          relevant ← relevant ∪ {category }
9          if getDepth(category) < max_depth :
10             for subcategory in subcat(category) :
11                 if subcategory not in visited :
12                     queue.push(subcategory)
13                     visited ← visited ∪ {subcategory }
```

Classifying the categories into relevant and irrelevant. We train a binary classifier to predict if a category is relevant based on its features. In this and other tasks we use the standard L_2-regularized linear SVM [Fan *et al.*, 2008]. We chose the SVM classifier because of its popularity, availability in the machine learning toolkits, and reported performance in various domains.

Training the classifier requires providing examples of both relevant and irrelevant categories. In order to collect the training examples, we first run the simple breath-first traversal of the category graph starting from the root, and limiting the depth to `max_depth`. This results in the initial set of categories that are under the root category and within the distance of `max_depth` from it. From this initial set we randomly sample paths going down from the root and add the categories along these paths into the training set.

The features of a category that we use in this task are based on the depth (the distance from the root), the title, and the parent categories. The most important feature is the depth itself. Another feature is the maximum similarity of the category's title to any of its parents' titles, measured as the Jaccard index between the sets of the words stems.

There is a dependency between the relevance status of a category and those of its adjacent categories: relevant categories tend to have relevant children and parents and vice versa. The relevance status of the parents is, however, not known during classification. As a proxy for this information, we compute the fraction of parents that have been visited by the selection algorithm (and have thus already been identified as descendants of the root category), as well as their minimum, maximum and average depth.

3.2 Discriminating between classes and individuals

Having selected the set of relevant categories, we need to split them into classes and individuals. Following our general approach, we view this task as a classification problem. We collect a sample of relevant categories, annotate them manually as either classes or individuals, and use the sample to train a binary classifier. The categories can be classified independently from each other, which makes this task easier than the one described in the previous section.

We tried to minimize the effort and resources required for computing the features in order to make our approach lightweight and easily reusable. For this task we limited the set of features to those describing the title of the category. The property most indicative of the category type is the grammatical number (singular or plural) of the head word of the title. We encoded the grammatical number approximately with the word suffixes of lengths 1, 2, and 3. Computing suffices is simpler than deriving the grammatical number with a part-of-speech tagger: it is not prone to tagging errors, and can capture more subtle cases, such as learning that suffix *-are* in words like "software" and "hardware" is indicative of class. Despite simplicity, this feature alone proved sufficient for achieving decent performance in this task.

3.3 Classifying the relations between the nodes

At this step we need to establish the semantic relations between the entities in the ontology. We introduce a relation between two nodes of the ontology whenever there is a subcategory relation between the corresponding Wikipedia categories. By default, the introduced relation has a generic type, which we denote as `related_to`. More specific relation types can be defined for certain combinations of node types. Specifically, two classes may be linked with `subclass_of` relation, two individuals with `part_of`, and an individual and a class with the `instance_of` relation. In these settings, predicting the type of relation between the given two nodes is equivalent to predicting whether the relation is specific or generic, which is a binary classification problem.

For each specific relation type we train a separate classifier on a dedicated training set. The training sets are collected by sampling pairs of relevant categories linked with sub-category relations. The node types predicted at the previous step are used to filter out the category pairs that are inconsistent with relation types.

As previously, we only use features that describe the titles, or, more precisely, the relation between the titles of the parent and the child categories, namely:

- whether there is a verbatim match between the stems: 1) of the head words, 2) of the first words;
- the Jaccard similarity between the sets of stems,
- similarity between the *head words* (various measures),
- *average pairwise* similarity between the words,
- whether there is a hyponym relation between the words in the titles, and whether it is between: 1) head words, 2) a head and a non-head word, 3) non-head words;
- whether the titles are related with a certain pattern, e.g one is a substring of the other, or the titles end with the same phrase, etc..

For similarity between the words we took the maximum similarity between the possible senses of the words, according to WordNet. We used a number of similarity measures based on WordNet, as implemented in the NLTK [Bird *et al.*, 2009] package. These included the similarity measures due to Leacock and Chodorow [1998], Wu and Palmer [1994], Resnik [1995], Lin [1998], and Jiang and Conrath [1997].

4 Evaluation

First, we applied our approach to building an ontology skeleton for the domain of computing. We executed and evaluated the 3 steps of our method: 1) selecting the subgraph of relevant categories, 2) classifying categories into classes and individuals, 3) classifying relations (`subclass_of` vs. `related_to`; `instance_of` vs. `related_to`; `part_of` vs. `related_to`). For each of the tasks 1)–3) we manually annotated a sample of "ground truth" data for both training and evaluation. The performance of the tasks was evaluated in cross-validation, and the parameters of the classifiers were tuned with a nested cross-validation on the training parts of each fold. As performance measures we used the accuracy of the prediction, the F1 scores with respect to both classes (referred to as the positive and the negative class), and the sum of the F1 scores weighted according to the class sizes.

To put the results into perspective, we implemented two standard baselines for each of the three tasks. The **majority rule** baseline always predicts the most frequent class in

the training set; it has zero recall and zero F1 measure on the smaller (minority) class. The **stratified random** baseline predicts randomly, but respecting the distribution of the classes in the training set. The standard baselines show their best when the classes are imbalanced, with the accuracy approaching 100% in the extreme case. The weighted F1 measure of the random baseline is equal (in expectation) to the accuracy, while for the majority rule it is always smaller than the accuracy. We use the weighted F1 score as our main performance measure, as it allows for a more meaningful comparison to the majority rule.

4.1 Evaluation of the relevant subgraph selection

We started from the Wikipedia category `Computing` as the root, and executed the breadth-first selection procedure described in Section 3.1. The 7-level deep selection produced 28 264 categories, from which a random sample of 1 000 categories was selected for labeling. Two authors of this paper independently annotated the categories in the set as 'relevant' or 'irrelevant', omitting the dubious cases. The 642 categories for which the two annotators gave the same label were selected as the ground truth, and contained 264 relevant and 378 irrelevant categories according to the labeling.

We evaluated the category selection using 5-fold cross-validation. For each fold, the category classifier was trained on the training part of the fold and plugged into Algorithm 1. The output of the algorithm was then evaluated on the test part of the fold. Table 1 shows the performance on this task. The learning curve (omitted for space reasons) shows that as few as 30 categories suffice to train the selection procedure.

Table 1: Performance of category selection for `Computing`. The scores (mean and standard deviation) are given in per cents.

method	accuracy	F1 pos.	F1 neg.	**weighted F1**
Ours	92 (0.6)	91 (0.8)	94 (0.5)	92 (0.6)
Depth(4)	91 (0.8)	89 (1.3)	92 (0.6)	91 (0.9)
Majority	59 (0.4)	0 (0)	74 (0.3)	44 (0.5)
Random	53 (7.6)	38 (8.4)	60 (4.8)	55 (6.0)

In addition to the standard baselines, we implemented a more informed **depth-based baseline**. The depth-based method marks all nodes within a certain distance from the root as relevant, leaving out all the rest. Figure 1 shows the accuracy of this baseline depending on the selection depth. Table 1 includes the best performance (at depth 4), which almost reaches that of our method.

After evaluating the category selection using cross-validation, we trained the classifier on the whole ground truth dataset, and executed Algorithm 1 from scratch with the retrained classifier. This final run of the algorithm produced a set of 7159 categories, which we then used in the subsequent stages of the overall procedure.

Evaluating the coverage of the selected categories. ACM Computing Classification System[4] (ACM CCS) is a standard and widely-used categorization of computer science. Its current version contains over two thousand concepts, and can be seen as a gold standard for topics relevant to computing.

[4] https://www.acm.org/about/class/2012

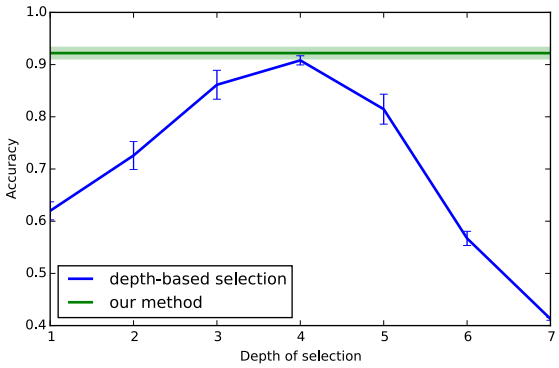

Figure 1: Accuracy of the depth-based baseline depending on the depth. The horizontal line represents our method. The green band and the vertical blue bars show the standard deviation as per cross-validation.

We evaluated the results of our selection procedure by estimating how well they cover the concepts of ACM. For the purpose of this experiment, we established a partial mapping between the concepts of ACM and the categories of Wikipedia using an automatic ontology matching technique [Jiménez-Ruiz *et al.*, 2012], resulting in 398 pairs of the corresponding topics. Assuming that ACM CCS contains only relevant topics, all mapped Wikipedia categories were considered relevant. Of the 398 categories relevant according to the mapping, 327 were also marked as relevant by our algorithm, corresponding to the coverage (or recall) of **0.82**.

The question of which topics are relevant to a given domain is somewhat subjective. Consequently, there was some disagreement between ACM CCS and the ground truth annotations on which our algorithm was trained. Some of the ACM CCS concepts irrelevant according to our labeling include: `Reference`, `Measurement`, `Cartography`, `Reference_works`, and `Documents`. Thus, even our ground truth labeling had a non-perfect recall, estimated at **0.90**.

The precision could not be estimated in the same way, firstly because ACM CCS does not cover all of the Wikipedia categories relevant to `Computing`, and secondly because our mapping between ACM CCS and Wikipedia was only partial.

4.2 Evaluation of the node type classification

Having selected the subgraph of categories relevant to `Computing`, the next step was to group the nodes into classes and individuals. From the 7 159 categories obtained in the previous step, we randomly selected a sample of 270 categories for manual annotation, and used the sample for training and evaluation. Table 2 summarizes the performance scores of our method, as well as of the two standard classifiers.

Comparison with WikiTaxonomy.
WikiTaxonomy [Ponzetto and Strube, 2007] is a large-scale taxonomy derived from the Wikipedia category network, and containing over a hundred thousand categories. Similarly to

Table 2: Performance of the category type classification for Computing. The scores (mean and standard deviation) are given in per cents.

method	accuracy	F1 pos.	F1 neg.	**weighted F1**
Ours	95 (4.1)	96 (3.3)	92 (5.4)	95 (4.0)
Majority	66 (1.0)	80 (0.7)	0 (0)	52 (1.2)
Random	54 (8.9)	65 (6.1)	40 (13.4)	61 (6.6)

our work, WikiTaxonomy discriminates between classes and individuals, and establishes subclass_of and instance_of relations between them. We looked at WikiTaxonomy as an alternative classification of categories, and measured its performance on our ground truth dataset. As our dataset was collected from a recent version of Wikipedia, it included some new categories that were not present in WikiTaxonomy. For the purpose of comparison, we limited the test set to the 75 categories that WikiTaxonomy did contain. The weighted F1 scores of our method and WikiTaxonomy on this dataset were **0.92** and **0.54** respectively.

We should point out the dramatic difference in the proportion of classes in WikiTaxonomy (8% classes, 92% individuals) versus in our labeling (60% classes, 30% individuals).

This suggests that our notions of classes and individuals and the criteria for assigning categories to these groups are likely different from those used in WikiTaxonomy.

Most of the errors of our method in this task were related to difficulties in distinguishing between the plural and the singular forms in the category titles. For instance, Robotics, Bioinformatics, and Computer graphics were mistakenly labeled as classes. With few exception, the mistakes of WikiTaxonomy, according to our ground truth labeling, were related to predicting categories like Data management, Graph coloring, and World Wide Web as classes.

4.3 Evaluation of the relation type classification

For each of the three specific relation types (subclass_of, instance_of, and part_of) we evaluated the corresponding binary classification task of discriminating between that specific relation and the generic related_to. For each of the three subtasks we collected and manually labeled a "ground truth" dataset of parent-child category pairs.

For the subtasks of discovering subclass_of and instance_of relations, we used WikiTaxonomy as a natural alternative method to compare with. For the sake of comparison, we measured the performance of our method on the fraction of pairs from the test data, for which both parent and child categories were contained in WikiTaxonomy. In order to have enough such pairs, we specifically sampled a number of them and included into the dataset for manual annotation. The performance of WikiTaxonomy on the subclass_of relation was measured in the following way: if for a certain parent-child pair of categories from the test set either subclass_of or instance_of relation was found in WikiTaxonomy, we considered that WikiTaxonomy predicted the subclass_of relation between these categories; if no relation was found, we considered that WikiTaxonomy predicted the generic related_to relation. The performance on the instance_of task was measured in the same way.

Table 3: Performance of the relation type classification for Computing. The scores (mean and standard deviation) are given in per cents.

method	accuracy	F1 pos.	F1 neg.	**weighted F1**
subclass_of				
Ours	80 (9.0)	87 (6.0)	48 (29.4)	83 (7.6)
WikiTx	66 (10.8)	74 (10.9)	39 (24.5)	71 (9.4)
Majority	85 (9.8)	92 (6.0)	0 (0)	79 (13.6)
Random	79 (8.6)	87 (5.8)	15 (24.0)	78 (12.2)
instance_of				
Ours	67 (11.9)	71 (16.8)	55 (14.5)	67 (10.9)
WikiTx	68 (13.8)	66 (13.6)	66 (15.4)	68 (14.4)
Majority	59 (15.5)	73 (15.2)	0 (0)	45 (16.0)
Random	53 (13.2)	61 (15.4)	37 (14.6)	52 (13.2)
part_of				
Ours	59 (13.4)	63 (23.0)	32 (32.9)	62 (15.4)
Majority	69 (26.2)	79 (19.5)	0 (0)	59 (33.1)
Random	62 (17.0)	70 (12.9)	33 (34.8)	66 (17.0)

Table 3 summarizes the performance scores of both WikiTaxonomy and our method, as well as the standard baselines. The relatively high score of the baselines on the subclass_of relation is due to the class imbalance (about 85% of subclass_of vs. 15% of related_to). On this relation our method outperformed the baselines, though insignificantly. On the instance_of relation the performance of our method is comparable to WikiTaxonomy. On the part_of relation, due to the difficulty of the task our method performs comparably to the random baseline.

4.4 Evaluation on a different domain

In order to evaluate the generalization of the proposed method, we executed it on the domain of Music. Table 4 summarizes the results of the experiments. The best performance was achieved when the classifiers were trained on the annotated data from the new domain. Interestingly, the method performed reasonably well on the first two tasks even with the classifiers trained on Computing.

As we can see from the table, our methods performed well on the tasks of selecting the relevant categories, classifying the category types and classifying the subclass_of relation, and poorly on classifying the other two relation types. As in the case of other domain, the baseline depth-based selection performed well, when the depth was selected appropriately. In case of Music the optimal depth turned out to be equal 6. Note that our machine learning–based method was not informed about this optimal depth and selected the nodes through the automatic procedure of Algorithm 1.

We can also see that in case of the Music domain, the performance of our method on the instance_of relation is significantly worse, and also worse than that of the baselines. The main reason for this is that in this domain we could hardly find negative examples for this type of relation (4 negative examples in a sample of 54). It is clear that in this case it is "safe" to always predict the positive class (the majority rule). The results on this relation could be improved by selecting a larger sample containing more negative examples. The part_of re-

Table 4: Performance of the various tasks on the domain `Music`. The scores (mean and standard deviation) are given in per cents.

method	accuracy	F1 pos.	F1 neg.	**weighted F1**
category selection				
Ours	89 (0.9)	92 (0.7)	79 (1.4)	88 (0.9)
Ours*	86	90	78	89
Depth(6)	89 (2.9)	92 (2.0)	79 (5.8)	88 (3.1)
category type classification				
Ours	98 (2.0)	99 (1.2)	92 (6.6)	98 (2.0)
Ours*	98 (2.0)	99 (1.2)	92 (6.6)	98 (2.0)
Majority	85 (0.6)	92 (0.3)	0 (0)	79 (0.8)
Random	76 (4.7)	83 (0.3)	17 (8.3)	73 (2.9)
relation `subclass_of`				
Ours	95 (5.5)	97 (3.3)	71 (83.0)	96 (4.6)
WikiTx	60 (10.8)	71 (11.0)	31 (14.5)	68 (10.8)
Majority	90 (5.5)	95 (3.1)	0 (0)	86 (8.0)
Random	84 (6.3)	91 (3.9)	4 (12.0)	82 (7.9)
relation `instance_of`				
Ours	71 (29.7)	73 (37.1)	12 (21.3)	72 (35.1)
WikiTx	41 (23.2)	50 (29.0)	13 (20.4)	49 (28.1)
Majority	93 (8.6)	96 (4.7)	0 (0)	90 (12.5)
Random	84 (15.8)	91 (10.2)	0 (0)	85 (14.0)
relation `part_of`				
Ours	45 (21.0)	47 (33.9)	17 (22.3)	44 (25.9)
Majority	80 (12.3)	88 (7.9)	0 (0)	71 (16.7)
Random	67 (16.1)	77 (15.1)	12 (18.4)	66 (15.8)

* Trained on `Computing`

lation, on the other hand, is intrinsically more difficult, and likely requires more sophisticated features based on semantics and background knowledge.

5 Related Work

5.1 WikiTaxonomy

WikiTaxonomy is a large-scale taxonomy derived from the categories of Wikipedia. In [Ponzetto and Strube, 2007] the authors described a method for identifying the `is_a` relation between categories (a union of our `subclass_of` and `instance_of`). A successive work [Zirn *et al.*, 2008] concerned discriminating between the individuals and classes. The methods described in these works consisted of multiple steps, each of which refined the result based on a number heuristic rules. The rules examined the titles of the categories, the connections between categories, and the relations between the categories and the articles.

Similarly to these works, our method uses the information in the category titles to classify both categories and relations between them. The syntactic features of the titles implemented in our method are largely inspired by the rules used in WikiTaxonomy. A major distinction of our work is that we aim to provide domain-specific ontologies, and thus need to extract a portion of categories relevant for a given domain. Secondly, we rely on machine learning to combine various pieces of evidence, avoiding ad-hoc rule-based pipelines and rule combination schemes. Another distinction of our work is that we target more relation types.

5.2 Wikipedia-based ontologies

The following works present large-scale ontologies extracted from the various kinds of knowledge contained in Wikipedia.

DBpedia [Lehmann *et al.*, 2014] extracts knowledge available in Wikipedia in semi-structured format, such as infoboxes, between-language links and redirects. The central entities in DBpedia correspond to Wikipedia articles. The Wikipedia categories are not used as classes to group the entities. Instead, DBpedia relies on a manually curated ontology that was created from the most commonly used infoboxes, and contains (as of February 2015) several hundred classes and over two thousand properties. The category network is exported in SKOS format, preserving the hierarchical structure, but not enriched semantically in any way.

YAGO [Suchanek *et al.*, 2008; Hoffart *et al.*, 2013] similarly extracts structured information from Wikipedia in the form of facts about entities that represent Wikipedia articles. In contrast to DBpedia, it integrates WordNet [Miller, 1995], and relies on its taxonomical structure to form the class subsumption hierarchy. From Wikipedia categories, only the leaf ones are taken into consideration, and a simple heuristic is used to identify the leaf categories that represent classes. In addition to the relations extracted from infoboxes, some relations are derived from the so-called relational categories.

Catriple [Liu *et al.*, 2008] also uses Wikipedia category names to generate facts about the articles, with new property types being extracted automatically. In contrast to other systems, Catriple observed the patterns in the titles of the parent-child category pairs. For instance, the two categories, `Songs_by_artist` and `The_Beatles_songs`, suggest that "artist" is a property, and "The Beatles" is a value.

The main distinction of the described works is that they focus on Wikipedia articles as entities and mainly derive non-taxonomical relations describing the articles. For the taxonomical organization of knowledge these sources use external high-level hierarchies, such as WordNet, or the manually created DBpedia ontology. Our work is specifically focused on the *fine-grained* and *hierarchical* structure of the domains, which we derive from the Wikipedia categories. The information about the fine-grained structure 1) is not available in general-purpose ontologies and classification schemes, 2) cannot be reasonably provided manually due to the large scale (thousands of nodes and relations).

6 Discussion and future work

We proposed an automated method for bootstrapping domain ontologies from the category network of Wikipedia. In all the three steps of the process—extracting the relevant concepts, identifying classes and individuals, and classifying relations—the method relies on supervised classification to combine various pieces of evidence. This uniform approach is flexible, and can easily incorporate new features. The flexibility comes at the cost of annotating training examples, which can be facilitated with interactive tools. Furthermore,

the annotation has to be performed at most once for a given domain. Our experiments suggested that the method performs reasonably well on the first two tasks even without retraining on a new domain. We plan to further investigate the questions of domain dependence in the future work.

One property of our method is that it relies only on the information present in the categories, namely the titles and (only in the first of the three tasks) the subcategory relations. Despite the limited information, the method has performed well in selecting the relevant categories, identifying the classes and individuals, and identifying the `subclass_of` relation. The performance on the other two relations (`instance_of` and `part_of`) suggests that these types are more difficult to capture. In particular, it is clear that the category titles often contain insufficient information, even for a human, to determine the `part_of` relation: consider, for instance, `part_of`(`Social_media`, `World_Wide_Web`) or `part_of`(`Data_warehousing`, `Business_intelligence`). Improving the accuracy on these relations requires additional knowledge provided to the classifier. Potential sources of additional knowledge include the texts of categories and articles of Wikipedia, and the structure of the extended category-article network. We also plan to investigate into the learning algorithms for joint (rather than independent) classification of categories and relations, which would allow incorporating more powerful relational features [Getoor and Taskar, 2007]. Finally, the utility of our method for facilitating the construction of domain ontologies should be more comprehensively assessed in a user study with ontology engineers.

References

[Bird *et al.*, 2009] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python*. O'Reilly Media, Inc., 2009.

[Fan *et al.*, 2008] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

[Getoor and Taskar, 2007] Lise Getoor and Ben Taskar. *Introduction to statistical relational learning*. MIT press, 2007.

[Hoffart *et al.*, 2013] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence*, 194:28–61, 2013.

[Iqbal *et al.*, 2013] Rizwan Iqbal, Masrah Azrifah Azmi Murad, Aida Mustapha, Sharef, and Nurfadhlina Mohd. An analysis of ontology engineering methodologies: A literature review. *Research Journal of Applied Sciences, Engineering and Technology*, 6(16):2993–3000, 2013.

[Jiang and Conrath, 1997] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[Jiménez-Ruiz *et al.*, 2012] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, and Yujiao Zhou. Logmap 2.0: Towards logic-based, scalable and interactive ontology matching. SWAT4LS '11, pages 45–46, New York, NY, USA, 2012. ACM.

[Leacock *et al.*, 1998] Claudia Leacock, George A Miller, and Martin Chodorow. Using corpus statistics and wordnet relations for sense identification. *Computational Linguistics*, 24(1):147–165, 1998.

[Lehmann *et al.*, 2014] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. DBpedia – a large-scale, multilingual knowledge base extracted from Wikipedia. *Semantic Web Journal*, 2014.

[Lin, 1998] Dekang Lin. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304, 1998.

[Liu *et al.*, 2008] Qiaoling Liu, Kaifeng Xu, Lei Zhang, Haofen Wang, Yong Yu, and Yue Pan. Catriple: Extracting triples from Wikipedia categories. In *The Semantic Web*, Lecture Notes in Computer Science. 2008.

[Miller, 1995] George A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[Mirylenka and Passerini, 2013] Daniil Mirylenka and Andrea Passerini. Navigating the topical structure of academic search results via the Wikipedia category network. In *CIKM'13*, pages 891–896. ACM, 2013.

[Osborne *et al.*, 2013] Francesco Osborne, Enrico Motta, and Paul Mulholland. Exploring scholarly data with Rexplore. In *The Semantic Web ISWC 2013*, volume 8218 of *Lecture Notes in Computer Science*, pages 460–477. 2013.

[Ponzetto and Strube, 2007] Simone Paolo Ponzetto and Michael Strube. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, pages 1440–1445. AAAI Press, 2007.

[Resnik, 1995] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

[Suárez-Figueroa *et al.*, 2012] Mari Carmen Suárez-Figueroa, Asunción Gómez-Prez, Enrico Motta, Aldo Gangemi, MariCarmen Suárez-Figueroa, and Mariano Fernández-López. *The NeOn Methodology for Ontology Engineering*. Springer Berlin Heidelberg, 2012.

[Suchanek *et al.*, 2008] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A Large Ontology from Wikipedia and WordNet. *Elsevier Journal of Web Semantics*, 2008.

[Wu and Palmer, 1994] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[Zirn *et al.*, 2008] Cäcilia Zirn, Vivi Nastase, and Michael Strube. Distinguishing between instances and classes in the Wikipedia Taxonomy. ESWC'08, pages 376–387, Berlin, Heidelberg, 2008. Springer-Verlag.