UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

FORMALIZING THE GET-SPECIFIC DOCUMENT CLASSIFICATION ALGORITHM

Fausto Giunchiglia, Ilya Zaihrayeu, and Uladzimir Kharkevich

March 2007

Technical Report # DIT-07-013

Formalizing the Get-Specific Document Classification Algorithm

Fausto Giunchiglia, Ilya Zaihrayeu, and Uladzimir Kharkevich

Department of Information and Communication Technology
University of Trento, Italy
{fausto,ilya,kharkevi}@dit.unitn.it

Abstract. The paper represents a first attempt to formalize the get-specific document classification algorithm and to fully automate it through reasoning in a propositional concept language without requiring user involvement or a training dataset. We follow a knowledge-centric approach and convert a natural language hierarchical classification into a formal classification, where the labels are defined in the concept language. This allows us to encode the get-specific algorithm as a problem in the concept language. The reported experimental results provide evidence of practical applicability of the proposed approach.

1 Introduction

Classification hierarchies have always been a natural and effective way for humans to organize their knowledge about the world. These hierarchies are rooted trees where each node defines a topic category. Child nodes' categories define aspects or facets of the parent node's category, thus creating a multifaceted description of the objects which can be classified in these categories. Classification hierarchies are used pervasively: in conventional libraries (e.g., the Dewey Decimal Classification system (DDC) [10]), in web directories (e.g., DMoz [3]), in e-commerce standardized catalogues (e.g., UNSPSC [4]), in personal email and file system folders, and so on.

Standard classification methodologies amount to manually organizing objects into classification categories following a predefined system of rules. The rules may differ widely in different approaches, but there is one classification pattern which is commonly followed. The pattern is called the *get-specific principle*, and it requires that an object is classified in a category (or in a set of categories), which most specifically describes the object. In practice, it usually means that this category lies as deep in the classification tree as possible, while still being more general than the topic of the object. Following the get-specific principle is not easy and is constrained by a number of limitations, discussed below:

- the meaning of a given category is implicitly codified in a natural language label, which may be ambiguous and may therefore be interpreted differently by different classifiers;

- a link, connecting two nodes, may also be ambiguous in the sense that it may specify the meaning of the child node, of the parent node, or of both. For instance, a link connecting the parent node *"programming"* with its child node *"Java"* may, or may not mean that (a) the parent node means "computer programming" (and not, for example, "events scheduling"); (b) that the child node means "Java, the programming language" (and not "Java, the island"); or (c) that the parent node's meaning excludes the meaning of the child node, *i.e.*, it is "programming and *not* Java";
- as a consequence of the previous two items, the classification task also becomes ambiguous in the sense that different classifiers may classify the same objects differently, based on their subjective opinion.

In the present paper we propose an approach to converting classifications into *formal classifications*, whose labels are encoded in a propositional concept language. Apart from this, we present a classification model and show how the get-specific algorithm can be described in this model. We then show how the model and the algorithm can be encoded in the concept language, which allows us to fully automate document population in formal classifications through propositional reasoning. Note that by doing this, we eliminate the three ambiguities discussed above. In order to evaluate our approach, we have re-classified documents from several branches of the DMoz directory without any human involvement or an a priori created training dataset. The results show the viability of the proposed approach, which makes it a realistic alternative to the standard classification approaches used in Information Science.

The remainder of the paper is organized as follows. In Section 2 we introduce the classification model, we show how the get-specific algorithm can be described in this model, and we identify the main problems peculiar to the algorithm. In Section 3 we show how classifications can be translated into formal classifications, how the get-specific algorithm can be encoded in the concept language, and how its peculiar problems can be dealt with in the concept language. In Section 4 we present and discuss evaluation results of our approach. In Section 5 we discuss the related work and, in particular, we compare our approach to that used in Information Science. Section 6 summarizes the results and concludes the paper.

2 The Get-Specific Classification Algorithm

Classifications are hierarchical structures used for positioning objects in such a way, that a person, who navigates the classifications, will be facilitated in finding objects related to a given topic. To attain such organization of objects, in standard classification approaches, objects are manually classified by human classifiers which follow a predefined system of rules. The actual system of rules may differ widely in different classification approaches, but there are some generic principles which are commonly followed. These principles make the ground of the *get-specific* algorithm, described in the rest of this section.

2.1 Classifications and a Classification Model

To avoid ambiguity of interpretation, in Definition 1 we formally define the notion of classification; and in Figure 1 we give an example of a classification, extracted from the DMoz web directory and adjusted for sake of presentation.

Definition 1. *A classification is a rooted tree $C = \langle N, E, L \rangle$ where N is a set of nodes, E is a set of edges on N, and L is a set of labels expressed in a natural language, such that for any node $n_i \in N$, there is one and only one label $l_i \in L$.*

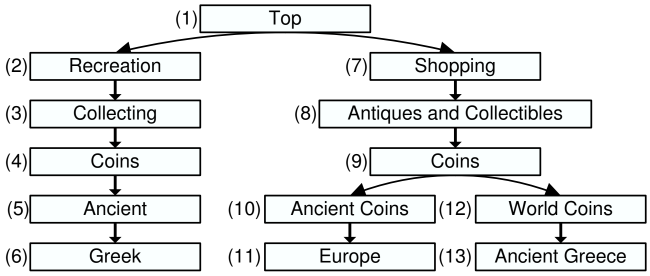


Fig. 1. A part of the DMoz web directory

We see the process of classification as a decision making procedure in which the classification tree is divided into a set of minimal decision making blocks. Each block consists of a node (called the root node of the block) and its child nodes (see Figure 2). While classifying an object, the classifier considers these blocks in a top-down fashion, starting from the block at the classification root node and then continuing to blocks rooted at those child nodes, which were selected for *further consideration*. These nodes are selected following decisions which are made at each block along two dimensions: *vertical* and *horizontal*. In the vertical dimension, the classifier decides which of the child nodes are selected as candidates for further consideration. In the horizontal dimension, the classifier decides which of the candidates are *actually* selected for further consideration. If none of the child nodes are appropriate or if there are no child nodes, then the root node of the block becomes a *classification alternative* for the given object. The process reiterates and continues recursively until no more nodes are left for further consideration. At this point, all the classification alternatives are computed. The classifier then decides which of them are most appropriate for the classification of the object and makes *the final classification choice*.

2.2 Modelling the Get-Specific Classification Algorithm

In this subsection we discuss the general principles lying behind the get-specific algorithm and we show how these principles can be implemented within the

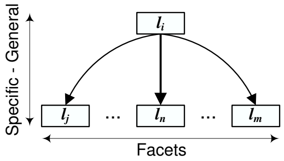


Fig. 2. The decision making block

model introduced in the previous subsection. Particularly, we discuss how vertical and horizontal choices, as well as the final classification choice are made.

- **Vertical choice.** Classification hierarchies are organized such that upper level categories represent more abstract or more general concepts, whereas lower level categories represent more concrete or more specific concepts. When the classifier searches for an appropriate category for the classification of an object, she looks for the ones which most specifically describe the object and, therefore, when making a vertical choice, she selects a child node as a candidate if it describes the object more specifically than the parent does. For example, if a document about ancient Greek coins is classified in the classification from Figure 1, then node n_6 is more appropriate for the classification than node n_5. When this principle is applied recursively, it leads to the selection of the category which lies as deep in the classification hierarchy as possible. The principle described above is commonly called the *get-specific principle*. Let us consider, for instance, how Yahoo! describes it:

  > *"When you suggest your site, get as specific as possible. Dig deep into the directory, looking for the appropriate sub-category."* [6]

  In practice, application of the get-specific principle usually means that a document is placed in only one node in a path from the root. However, in some cases, a document is placed in several most specific categories on the path. Examples of these approaches are Amazon [1] and Wikipedia [5].
- **Horizontal choice.** Child nodes may describe different aspects or *facets* of the parent node and, therefore, more than one child node may be selected in the vertical choice if a multifaceted document is being classified. As a consequence of this, the classifier needs to decide which of the several sibling nodes are appropriate for further consideration. When one sibling node represents a more specific concept than another, then the former is usually preferred over the latter. For example, node n_{10} is more appropriate for the classification of ancient Greek coins than node n_{12}. As a rule of thumb, the horizontal choice is made in favor of as few nodes as possible and, preferably, in favor of one node only. We call the principle described above, the *get-minimal principle.* Consider, for instance, how DMoz describes it.

  > *"Most sites will fit perfectly into one category. ODP categories are specialized enough so that in most cases you should not list a site more than once."* [2]

In conventional library systems, such as the DDC [10], the horizontal choice is always made in favor of only one node. This restriction is conditioned by the fact that a book, as a physical object, can be put only in one place on the shelf. Electronic directories, such as DMoz, do not have this constraint and, therefore, they can classify documents in multiple categories. Noteworthy, even if DMoz enforces the rule of single classification choice, in some cases, it finds it useful to classify documents in multiple places to improve navigability of and effectiveness of search in the electronic directory. In fact, there are about 10% of site listings which are classified in more than one category [12].

- **Tradeoff between vertical and horizontal choices.** The two principles described above cannot always be fulfilled at the same time. Namely, if the vertical choice results in too many candidates, then it becomes hard to fulfill the principle of minimality in the horizontal choice. In order to address this problem, a tradeoff needs to be introduced between the two requirements, which usually means trading specificity in favor of minimality. The following is an example of a tradeoff rule used in DMoz:

  > *"If a site offers many different things, it should be listed in a more general category as opposed to listing it in many specialized subcategories. [. . .] For example, if you have a site on US History covering the American Revolution, the Civil War, and the Presidents, you should not list it in each one of those subcategories, but rather list it in a general category such as Society: History: By_Region: North_America: United_States."* [2]

  As a mean to fulfill both principles and avoid the tradeoff, in each minimal decision making block, the child nodes must represent values from only one homogeneous facet. In this case, since such values are usually disjoint, vertical choice results into one candidate node only, if any at all.
- **The final classification choice.** When all classification alternatives are determined, the classifier confronts all of them in order to make her final classification choice. Note that now the choice is made not at the level of a minimal decision making block, but at the level of the whole classification. However, the classifier uses the same selection criteria as those used in the horizontal choice. For example, nodes n_6 and n_{13} are more appropriate for the classification of documents about ancient Greek coins than node n_{11}.

Note that the get-specific algorithm described above allows the classifier to position an object in the classification hierarchy without considering all the nodes of the hierarchy, thus it allows it to reduce information load on the classifier.

2.3 Problems of the Get-Specific Classification Algorithm

As discussed in [12], there are several problems which are common to document classification algorithms. The problems are caused by the potentially large size of classifications, by ambiguity in natural language labels and in document descriptions, by different interpretations of the meaning of parent-child links, by different views of different classifiers on the classification of the same document, and so on. All these problems lead to nonuniform, duplicate, and error-prone classification, especially when the classification is populated by multiple classifiers. In addition to the problems discussed in [12], the get-specific algorithm has two peculiar problems, related to the two decision dimensions. We discuss these problems below on the example of a document titled *"Gold Staters in the Numismatic Marketplace"*, being classified in the classification from Figure 1.

- **Vertical choice: the "I don't know" problem.** The classifier may make a mistake because she does not (fully) understand the meaning of a child node or the relation of the document to that node, whereas the node is a valid candidate. For example, the classifier may not know that "Gold Stater" is a coin of ancient Greece and, therefore, will erroneously classify the document into node n_5, whereas a more appropriate node is n_6.
- **Horizontal choice: the "Polarity change" problem.** The classifier may make a mistake when one of the sibling candidate nodes is more appropriate for further consideration than another, but a descendent of the latter is more appropriate for the classification of the object than a descendant of the former node. For instance, the label of node n_{10} more specifically describes the document than the label of node n_{12}. Therefore, the classifier will choose node n_{10} only as a candidate for further consideration and will finally classify the document in node n_{11}, whereas a more appropriate node for the classification is node n_{13}, a descendant of n_{12}.

3 Formalizing the Get-Specific Classification Algorithm

In this section we formalize the get-specific classification algorithm by encoding it as a problem expressed in propositional Description Logic language [7], referred to as L^C. First, we discuss how natural language node labels and document descriptions are converted into formulas in L^C. Second, we discuss how we reduce the problems of vertical, horizontal, and final classification choices to fully automated propositional reasoning. Finally, we show how the problems discussed in Section 2.3 can be dealt with in a formal way.

3.1 From Natural Language to Formal Language

Classification labels are expressed in a natural language, which is ambiguous and, therefore, is very hard to reason about. In order to address this problem, we encode classification labels into formulas in L^C following the approach proposed in [12]. This allows us to convert the classification into a new structure, which we call *Formal Classification* (FC):

Definition 2. *A Formal Classification is a rooted tree $FC = \langle N, E, L^F \rangle$ where N is a set of nodes, E is a set of edges on N, and L^F is a set of labels expressed in L^C, such that for any node $n_i \in N$, there is one and only one label $l_i^F \in L^F$.*

Note that even if L^C is propositional in nature, it has a set-theoretic semantics. As proposed in [12], the interpretation of a concept is the set of documents, which are *about* this concept. For instance, the interpretation of concept `Capital` (defined as "a seat of government") is the set of documents about capitals, and *not* the set of capitals which exist in the world.

Below we briefly describe how we convert natural language labels into formulas in L^C. Interested readers are referred to [12] for a complete account. Figure 3 shows the result of conversion of the classification from Figure 1 into a FC.

1. **Build atomic concepts.** Senses of nouns and adjectives become atomic concepts, whose interpretation is the set of documents about the entities or individual objects, denoted by the nouns, or which possess the qualities, denoted by the adjectives. We enumerate word senses using WordNet [19], and we refer to them as follows: `pos-lemma#i`, where `pos` is the part of speech, `lemma` is the word lemma, and `i` is the sense number in WordNet.
2. **Disambiguate word senses.** Irrelevant word senses are identified and corresponding to them atomic concepts are discarded. As proposed in [17], if there exists a relation (e.g., synonymy, hypernymy, or holonymy) in WordNet between any two senses of two words in a label (or in different labels on a path to the root), then corresponding to them concepts are retained and other concepts are discarded. If no relation is found, then we check if a relation exists between two WordNet senses by comparing their glosses [15]. If no relation is found after these two steps, then we keep all the concepts.
3. **Build complex concepts.** Complex concepts are built as follows: first, words' formulas are built as the logical disjunction ($\sqcup$) of atomic concepts corresponding to their senses (remaining after step 2). Second, syntactic relations between words are translated into logical connectives of L^C. For example, a set of adjectives followed by a noun group is translated into the logical conjunction ($\sqcap$) of the formulas corresponding to the adjectives and to the nouns; prepositions like "of" and "in" are translated into the conjunction; coordinating conjunctions "and" and "or" are translated into the logical disjunction ($\sqcup$); words and phrases denoting exclusions, such as "except" and "but not", are translated into the logical negation ($\neg$).

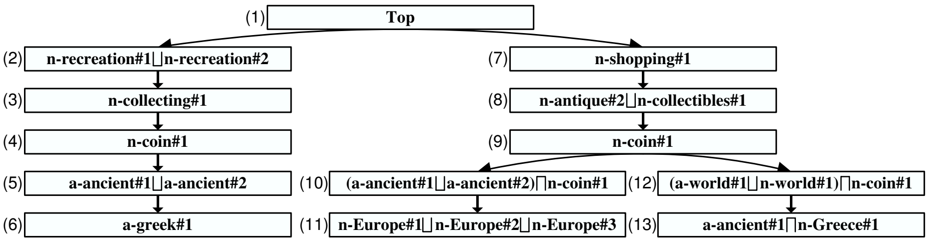

(1) Top
(2) n-recreation#1⊔n-recreation#2
(3) n-collecting#1
(4) n-coin#1
(5) a-ancient#1⊔a-ancient#2
(6) a-greek#1
(7) n-shopping#1
(8) n-antique#2⊔n-collectibles#1
(9) n-coin#1
(10) (a-ancient#1⊔a-ancient#2)⊓n-coin#1
(11) n-Europe#1⊔n-Europe#2⊔n-Europe#3
(12) (a-world#1⊔n-world#1)⊓n-coin#1
(13) a-ancient#1⊓n-Greece#1

Fig. 3. Formal Classification

Before a document can be automatically classified, it has to be assigned an expression in L^C, which we call the document concept, written C^d. The assignment of a concept to a document is done in two steps: first, a set of n keyphrases is retrieved from the document using text mining techniques (see, for example, [24]); the keyphrases are converted to formulas in L^C, and the document concept is then computed as the conjunction of the formulas.

3.2 The Algorithm

In the following we describe how we make vertical and horizontal choices, compute the tradeoff, and make the final classification choice in FCs. Note that the choice making problems are encoded as problems expressed in L^C.

- **Vertical choice.** A child node n_i is a candidate, given that a document with concept C^d is being classified, if the label of the node, l_i^F, subsumes C^d, i.e., if the following holds: $C^d \sqsubseteq l_i^F$. In formulas, if N_c is the set of child nodes in the block, then we compute the vertical choice $V(C^d)$ as:

  $$V(C^d) = \{n_i \in N_c | C^d \sqsubseteq l_i^F\} \qquad (1)$$

  If the vertical choice results in no candidates, then root node n_r of the current block is added to the set of classification alternatives $A(C^d)$:

  $$\textbf{if } |V(C^d)| = 0 \textbf{ then } A(C^d) \leftarrow A(C^d) \cup \{n_r\} \qquad (2)$$

  In Figure 4a we show an example of a situation when two child nodes n_2 and n_4 are selected for further consideration, and in Figure 4b we show an example of a situation when no child node can be selected.

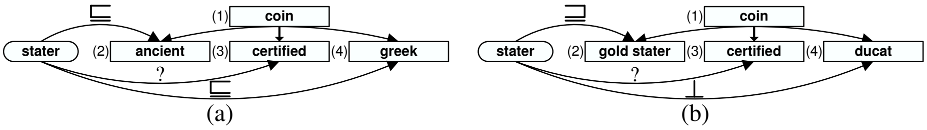


Fig. 4. Vertical choice ("?" means no relation is found)

- **Horizontal choice.** Given the set of candidates $V(C^d)$, we exclude those nodes from the set, whose label is more general than the label of another node in the set. In formulas, we compute the horizontal choice $H(C^d)$ as:

  $$H(C^d) = \{n_i \in V(C^d) | \nexists n_j \in V(C^d),\ s.t.\ j \neq i, l_j^F \sqsubseteq l_i^F, and\ l_j^F \not\sqsupseteq l_i^F\} \quad (3)$$

  We introduce the last condition (i.e., $l_j^F \not\sqsupseteq l_i^F$) to avoid mutual exclusion of nodes, whose labels in the FC are equivalent concepts. For instance, two syntactically different labels “seacoast” and “seashore” are translated into two equivalent concepts. When such situation arises, all the nodes, whose labels are equivalent, are retained in $H(C^d)$.

- **The tradeoff.** Whenever the size of $H(C^d)$ exceeds some threshold k, the nodes of $H(C^d)$ are discarded as candidates and root node n_r of the block is added to the set of classification alternatives $A(C^d)$. In formulas:

$$\textbf{if} |H(C^d)| > k \textbf{ then } H(C^d) \leftarrow \emptyset \textbf{ and } A(C^d) \leftarrow A(C^d) \cup \{n_r\} \tag{4}$$

- **The final classification choice.** When no more nodes are left for further consideration, set $A(C^d)$ includes all the classification alternatives. We compare them to make the final classification choice, but, differently from vertical and horizontal choices, we compare the *meanings* of nodes given their path to the root, and not their labels. We encode the meaning of node n_i into a concept in L^C, called *concept of node* [13], written C_i, and computed as:

$$C_i = \begin{cases} l_i^F & \text{if } n_i \text{ is the root of the } FC \\ l_i^F \sqcap C_j & \text{if } n_i \text{ is not the root, where } n_j \text{ is the parent of } n_i \end{cases} \tag{5}$$

Similar to how the horizontal choice is made, we exclude those nodes from $A(C^d)$, whose concept is more general than the concept of another node in the set. In formulas, we compute the final classification choice $C(A)$ as:

$$C(A) = \{n_i \in A(C^d) | \nexists n_j \in A(C^d),\ s.t.\ j \neq i,\ C_j \sqsubseteq C_i,\ and\ C_j \not\sqsupseteq C_i\} \tag{6}$$

The last condition (i.e., $C_j \not\sqsupseteq C_i$) is introduced to avoid mutual exclusion of nodes with the same meaning in the classification hierarchy. For instance, two paths *top/computers/games/soccer* and *top/sport/soccer/computer_games* lead to two semantically equivalent concepts. When such situation arises, all the nodes with the same meaning are retained in $C(A)$.

Computing Equations 1, 3, and 6 requires verifying whether the subsumption relation holds between two formulas in L^C. As shown in [12], a problem expressed in L^C can be rewritten as an equivalent problem expressed in propositional logic. Namely, if we need to check whether a certain relation *rel* (which can be $\sqsubseteq$, $\sqsupseteq$, $\equiv$, or $\perp$) holds between two concepts A and B, given some knowledge base $\mathcal{KB}$ (which represents our a priori knowledge), we construct a propositional formula according to the pattern shown in Equation 7 and check it for validity:

$$\mathcal{KB} \rightarrow rel(A, B) \tag{7}$$

The intuition is that $\mathcal{KB}$ encodes what we know about concepts A and B, and $rel(A, B)$ holds only if it follows from what we know. In our approach $\mathcal{KB}$ is built as the conjunction of a set of axioms which encode the relations that hold between *atomic* concepts in A and B. As discussed in Section 3.1, atomic concepts in L^C are mapped to the corresponding natural language words’ senses. These senses may be lexically related through the synonymy, antonymy, hypernymy, or holonymy relations. The idea, therefore, is to find the lexical relations using WordNet and to translate synonymy into the logical equivalence, antonymy into the disjointness, hypernymy and holonymy into the subsumption relation in L^C.

3.3 Dealing with Problems

Encoding a classification algorithm into a problem in L^C allows it to avoid many problems, which are common to classification algorithms [12]. Particularly, since the problem is encoded in a formal language, there is no ambiguity in interpretation of classification labels, of edges, and document contents. Apart from this, since computation is performed by a machine, the problem of classification size becomes largely irrelevant. Finally, since the formal algorithm is deterministic, the classification is always performed in a uniform way.

In Section 2.3 we discussed two problems, peculiar to the get-specific algorithm. Below we discuss what they mean in L^C and how they can be dealt with.

- **Vertical choice: the "I don't know" problem.** This problem arises when the specificity relation in Equation 1 cannot be computed while a human observes that it exists. The problem is caused by lack of background knowledge and it can be dealt with by adding missing axioms to the underlying knowledge base [14]. For instance, if we add a missing axiom which states that concept `Stater` (defined as "any of the various silver or gold coins of ancient Greece") is more specific than concept `Greek` (defined as "of or relating to or characteristic of Greece ..."), then the algorithm will correctly classify document *"Gold Staters in the Numismatic Marketplace"* into node n_6 in the classification shown in Figure 1.
- **Horizontal choice: the "Polarity change" problem.** The problem arises when the label of node n_i is more specific than the label of its sibling node n_j (i.e., $l_i^F \sqsubseteq l_j^F$), but the concept of a n_i's descendant node n_k is more general than the concept of a n_j's descendant node n_m (i.e., $C_k \sqsupseteq C_m$). In the simplest case, this problem can be dealt with by not performing the horizontal choice. In this case, both n_k and n_m will be in the classification alternative set for some document, and n_k will then be discarded when the final classification choice is made.

4 Evaluation

In order to evaluate our approach, we selected four subtrees from the DMoz web directory, converted them to FCs, extracted concepts from the populated documents, and automatically (re)classified the documents into the FCs. We extracted document concepts by computing the conjunction of the formulas corresponding to the first 10 most frequent words appearing in the documents (excluding stop words). The number of words was determined purely experimentally: smaller number of words did not make document concepts specific enough, which led to lower precision and recall; and larger number of words led to more computations without affecting significantly the final result. We used WordNet 2.0 [19] for finding word senses and their relations, and we used S-Match [13] for computing Equation 7. Parameter k for tradeoff computation was set to 2.

In the evaluation we employ standard information retrieval measures such as micro- and macro-averaged precision, recall, and F1 [21]. In Table 1 we report dataset statistics and evaluation results for each of the four datasets. We performed a detailed analysis of the "Languages" dataset results (see Figure 5). In Figure 5a we show how precision and recall are distributed among nodes. Figure 5b shows how far (in terms of the number of edges) an automatically classified document is from the node where it was actually classified in DMoz.

Dataset	Nodes	Docs	Max subtree depth	Mi-Pr	Mi-Re	Mi-F1	Ma-Pr	Ma-Re	Ma-F1
Photography[a]	27	871	4	0.2218	0.1871	0.2029	0.2046	0.1165	0.1485
Beverages[b]	38	1456	5	0.4037	0.4938	0.4442	0.3848	0.3551	0.3693
Mammals[c]	88	574	5	0.3145	0.3014	0.3078	0.3854	0.2677	0.3159
Languages[d]	157	1217	6	0.4117	0.4503	0.4301	0.4366	0.4187	0.4275

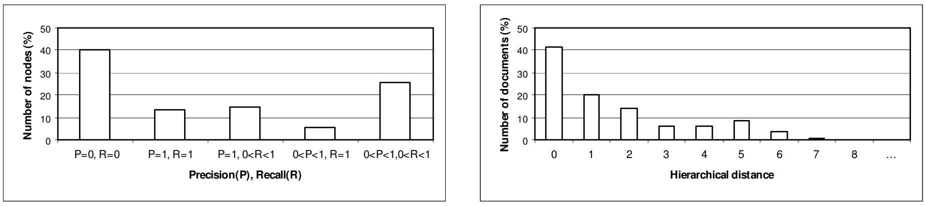
[a] http://dmoz.org/Shopping/Photography/.

[b] http://dmoz.org/Shopping/Food/Beverages/.

[c] http://dmoz.org/Health/Animal/Mammals/.

[d] http://dmoz.org/Science/Social_Sciences/Linguistics/Languages/Natural/Indo-European/.

Table 1. Dataset statistics and evaluation results



(a)

(b)

Fig. 5. Analysis of the "Languages" dataset results

From Figure 5a we observe that about 40% of nodes in the "Languages" dataset have precision and recall equal to 0[1]. After manual inspection of the results, we concluded that this problem is caused by lack of background knowledge. For instance, 8 documents about Slovenian language were misclassified because

[1] Precision for nodes with no documents was counted as 0.

there was no WordNet synset "Slovenian" defined as "the Slavic language spoken in Slovenia" and a hypernym relation of it with synset "Slavic language".

Figure 5b shows that about 20% of documents are classified in one edge distance from the node where they were originally populated, whereas 89% of them were classified one node higher on the path to the root. Note that this still allows it to find a document of interest by browsing the classification hierarchy. As we wrote in Section 2.2, in some approaches, documents are classified in several nodes on a path to the root to improve the navigability of the classification.

5 Related Work

The idea of that the get-specific classification algorithm can be encoded in a formal language and the first formal specification of the algorithm were reported in [12]. The current paper extends [12] in several respects. First, it proposes a classification model and shows how the algorithm can be implemented in this model. Second, it discusses how the model can be described and implemented in L^C. Third, it identifies the main problems peculiar to the get-specific algorithm and shows how they can be dealt with in L^C. Finally, for the first time, the current paper presents experimental results, which demonstrate that document classification can be fully automated using a knowledge-centric approach.

The idea of that natural language labels in classifications can be translated in a formal language was first introduced in [9], and, in [17], the authors provided a detailed account of the translation process using Description Logic as the target formal language. The current paper uses the translation rules described in [12], which originates from [17], but which uses the less computationally expensive propositional subset of Description Logics. In [12], the authors define a set-theoretic semantics for the translation rules and show that a propositional concept language is enough to capture the semantics of a large amount of labels.

In Information Science, document classification usually refers to supervised or unsupervised text categorization [21]. The principal difference between the two is that, in the former case, a small set of documents (called *the training set*) needs to be pre-classified in the categories to allow for a larger set of documents to be automatically classified. Category names do not need to be meaningful terms or phrases, and they can only be logical identifiers of the pre-classified sets. Unsupervised approaches do not provision document pre-classification, and are mainly based on the comparison of terms appearing in the documents and terms associated with each category [18]. Below we compare our approach to text categorization approaches in which categories are organized in a hierarchy.

Differently from the supervised case (e.g., see [11, 16, 23]), in our approach we do not need to have a pre-classified set of documents. In fact, classification choices depend on the *meaning* of classification labels and not on the documents already classified in nodes. This makes our approach very dynamic w.r.t. supervised classification since it does not require manual re-creation of the training set for the classification of the same set of documents if the category set changes. Noteworthy, some classification approaches rely on an underlying knowledge base (e.g., WordNet [19]) in order to find relations among words to optimize the construction of the feature space (e.g., see [8, 20]). However, these approaches still require a training dataset to operate.

Differently from the unsupervised approach (e.g., [18, 25]), we do not need to annotate classification nodes with a relatively large (w.r.t. the label size) set of keywords to classify documents. Apart from this, in formal classification labels, the terms are connected through logical connectives, which increases the expressiveness of the category description. However, unsupervised classification is the approach closest to ours from the text categorization domain in that it takes a classification and a set of documents as input and classifies the documents into the classification categories without a training dataset. The results, reached by the two approaches, are comparable. For instance, in [25], the authors report to reach max 42.70% in micro-F1 measure on different web directory datasets.

6 Conclusions and Future Work

The current paper makes a contribution at the turn of several disciplines. First, it takes the notion of classification from Library Science and shows how it can be converted in a form of ontology – the fundamental notion on the Semantic Web. Interestingly, the two notions are often used interchangeably in the two communities [22]. Second, we provide a classification model and show how the get-specific algorithm, commonly used in hierarchical document classification systems, can be described in this model. Third, it shows how document classification can be fully automated using a knowledge-centric approach, an approach which is conceptually different from the one used in Information Science. Finally, evaluation results reported in this paper demonstrate the proof of concept of our approach, which makes it a viable alternative to the conventional way of automated document classification.

Our future work includes: (a) development of more accurate document concept extraction algorithms; (b) evaluation of our approach in specific domains using domain ontology as the underlying knowledge base; (c) development of knowledge base enrichment algorithms which take into account the classification semantics (which, for example, will define concept `Stater` as more specific than concept `Greek`); and (d) automatic document re-classification when the structure of the classification hierarchy changes.

References

1. Amazon: See http://www.amazon.com/.
2. DMoz guidelines: See http://dmoz.org/guidelines/site-specific.html.
3. DMoz: See http://dmoz.org/.
4. UNSPSC: See http://www.unspsc.org/.
5. Wikipedia: See http://en.wikipedia.org/.
6. Yahoo! guidelines: See http://docs.yahoo.com/info/suggest/appropriate.html.

7. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
8. S. Bloehdorn and A. Hotho. Text classification by boosting weak learners based on terms and concepts. In *Proc. of IEEE International Conference on Data Mining (ICDM 04)*, pages 331–334. IEEE Computer Society Press, 2004.
9. P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: a new approach and an application. In *Proc. of the 2nd International Semantic Web Conference (ISWO'03). Sanibel Islands, Florida, USA*, October 2003.
10. Lois Mai Chan and J.S. Mitchell. *Dewey Decimal Classification: A Practical Guide.* Forest P.,U.S., December 1996.
11. S. T. Dumais and H. Chen. Hierarchical classification of web content. In *Proc. of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, Athens, GR, 2000. ACM Press.
12. F. Giunchiglia, M.Marchese, and I. Zaihrayeu. Encoding classifications into lightweight ontologies. *JoDS VIII*, Winter 2006.
13. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. Semantic schema matching. In *Proc. of CoopIS*, pages 347–365, 2005.
14. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. Discovering missing background knowledge in ontology matching. In *Proc. of ECAI*, 2006.
15. F. Giunchiglia and M. Yatskevich. Element level semantic matching. In *Proc. of Meaning Coordination and Negotiation workshop, ISWC*, 2004.
16. D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proc. of ICML-97, 14th International Conference on Machine Learning*, pages 170–178, Nashville, US, 1997. Morgan Kaufmann Publishers.
17. B. Magnini, L. Serafini, and M. Speranza. Making explicit the semantics hidden in schema models. In *Proc. of the Workshop on Human Language Technology for the Semantic Web and Web Services, held at ISWC-2003*, October 2003.
18. A. McCallum and K. Nigam. Text classification by bootstrapping with keywords, em and shrinkage. In *Proc. of ACL99 - Workshop for Unsupervised Learning in Natural Language Processing*, 1999.
19. G. Miller. *WordNet: An electronic Lexical Database.* MIT Press, 1998.
20. X. Peng and B. Choi. Document classifications based on word semantic hierarchies. In *Proc. of International Conference on Artificial Intelligence and Applications*, pages 362–367, 2005.
21. F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
22. D. Soergel. The rise of ontologies or the reinvention of classification. *Journal of the American Society for Information Science*, 50(12):1119–1120, 1999.
23. A. Sun and E. P. Lim. Hierarchical text classification and evaluation. In *Proc. of ICDM*, pages 521–528, 2001.
24. Peter D. Turney. Learning algorithms for keyphrase extraction. *Information Retrieval*, 2(4):303–336, 2000.
25. S. Veeramachaneni, D. Sona, and P. Avesani. Hierarchical dirichlet model for document classification. In *Proc. of ICML*, 2005.