Encoding Classifications into Lightweight Ontologies [⋆]

Fausto Giunchiglia, Maurizio Marchese, Ilya Zaihrayeu
{fausto, marchese, ilya}@dit.unitn.it

Department of Information and Communication Technology
University of Trento, Italy

Abstract. Classifications have been used for centuries with the goal of cataloguing and searching large sets of objects. In the early days it was mainly books; lately it has also become Web pages, pictures and any kind of digital resources. Classifications describe their contents using natural language labels, an approach which has proved very effective in manual classification. However natural language labels show their limitations when one tries to automate the process, as they make it very hard to reason about classifications and their contents. In this paper we introduce the novel notion of *Formal Classification*, as a graph structure where labels are written in a propositional concept language. Formal Classifications turn out to be some form of lightweight ontologies. This, in turn, allows us to reason about them, to associate to each node a normal form formula which univocally describes its contents, and to reduce document classification and query answering to reasoning about subsumption.

1 Introduction

In today's information society, as the amount of information grows larger, it becomes essential to develop efficient ways to summarize and navigate information from large, multivariate data sets. The field of classification supports these tasks, as it investigates how sets of "objects" can be summarized into a small number of classes, and it also provides methods to assist the search of such "objects" [11]. In the past centuries, classification has been the domain of librarians and archivists. Lately a lot of interest has focused also on the management of the information present in the web: see for instance the WWW Virtual Library project[1], or the web directories of search engines like Google, or Yahoo!.

Standard classification methodologies amount to manually organizing topics into hierarchies. Hierarchical library classification systems (such as the Dewey

[⋆] This paper is an integrated and extended version of two papers: the first with title "Towards a Theory of Formal Classification" was presented at the 2005 International Workshop on Context and Ontologies; the second with title "Encoding Classifications into Lightweight Ontologies" was presented at the 2006 European Semantic Web Conference.

[1] The WWW Virtual Library project, see http://vlib.org/.

Decimal Classification System (DDC) [3] or the Library of Congress classification system (LCC)[2]) are attempts to develop static, hierarchical classification structures into which all of human knowledge can be classified.

More recently, many search engines like Google, Yahoo as well as many eCommerce vendors, like Amazon, offer classification hierarchies (*i.e.*, web directories) to search for relevant items. Such web directories are sometimes referred to as *lightweight ontologies* [29]. However, as such, they lack at least one important property that ontologies must have: ontologies must be represented in a *formal language*, which can then be used for *automating reasoning* [21]. None of the existing human crafted classifications possesses this property. Because classification hierarchies are written in natural language, it is very hard to automate the classification task, and, as a consequence, standard classification approaches amount to *manually* classifying objects into classes. Examples include DMoz, a human edited web directory, which "*powers the core directory services for the most popular portals and search engines on the Web, including AOL Search, Netscape Search, Google, Lycos, DirectHit, and HotBot, and hundreds of others*" [28].

Although all the above mentioned classifications are based on well-founded classification methodologies, they have a number of limitations:

- the semantics 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 be considered to 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. This observation has an impact in particular on the fact that both current tasks of classification and search by means of browsing do not scale to large amounts of information.

In the present paper we propose an approach to converting classifications into Formal Classifications, or lightweight ontologies, thus eliminating the three ambiguities discussed above. This in turn allows us to automate, through propositional reasoning, the essential tasks of document classification and query answering. Concretely, we propose a three step approach:

- first, we convert a classification into a new structure, which we call *Formal Classification* (*FC*), where all the labels are expressed in a propositional Description Logic (DL) language (*i.e.*, a DL language without roles) [1];

[2] The Library of Congress Classification system, see http://www.loc.gov/catdir/cpso/lcco/lcco.html/.

- second, we convert a FC into a *Normalized Formal Classification* (*NFC*). In NFCs each node's label is a propositional DL formula, which univocally codifies the meaning of the node in the classification, taking into account both the label of the node and its position within the classification;
- third, we encode document classification and query answering in NFCs as a propositional satisfiability (SAT) problem, and solve it using a sound and complete SAT engine.

NFCs are *full-fledged* lightweight ontologies, and have many nice properties. Among them:

- NFC node labels univocally codify the set of documents which can be classified in these nodes;
- NFCs are *taxonomies* in the sense that, from the root down to the leaves, labels of child nodes are subsumed by the labels of their parent nodes;
- as nodes' labels codify the position of the nodes in the hierarchy, document classification and query answering can be done simply by analyzing the set of labels. There is no need to inspect the edge structure of the NFC.

The remainder of the paper is organized as follows. In Section 2 we introduce classifications and discuss how they are currently used in real use cases. In Section 3 we motivate a formal approach to dealing with classifications. In Section 4 we introduce the notion of FC as a way to disambiguate labels in classifications. In Section 5 we discuss how we disambiguate links in classifications by introducing the notion of NFC. In Section 6 and Section 7 we show how the two main operations performed on classifications, namely classification and search, can be fully automated in NFCs by means of propositional reasoning. In Section 8 we discuss related work. Section 9 summarizes the results and concludes the paper.

2 Classifications

Classifications are hierarchical structures used to organize large amounts of objects [17]. These objects can be of many different types, depending on the characteristics and uses of the classification itself. In a library, they are mainly books or journals; in a file system, they can be any kind of file (*e.g.*, text files, images, applications); in the directories of Web portals, the objects are pointers to Web pages; in market places, catalogs organize either product data or service titles. Classifications are useful for both object classification and retrieval. Users browse the hierarchies and catalogue or access the objects associated with different concepts, which are described by natural languages labels. Noteworthy, many widely used classifications impose a simple structure of a rooted tree. Examples of tree-like classifications are DMoz, DDC, Amazon, directories of Google and Yahoo, file system directories, and many others[3].

We define the notion of Classification as follows:

[3] While making this statement we excluded from consideration secondary classification links normally used for improving navigability and which make the classification a DAG. An example of such links is the "related" links of DMoz.

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

Classifications, as tree-like structures with natural language node labels, are used in a variety of domains such as standardization (*e.g.*, eCl@ss [4]), controlled thesauri (*e.g.*, MeSH [27]), and many others. Depending on their target application, classifications and their elements are given a specific interpretation and can therefore be treated differently. In this paper, we see classifications as objects whose primary purpose is the classification of and search for documents. We therefore define the *classification semantics* as strongly related to documents, and we define it at three levels:

- **Label**: labels, as such, describe real world entities or individual objects, and the meaning of a label in a classification is the set of documents which are *about* the entities or individual objects described by the label. Note, that a label can denote a document, *e.g.*, a book; and, in this case, the classification semantics of this label is the set of documents which are about the book, *e.g.*, book reviews. Note that the label semantics is fully captured by the label itself and nothing else;
- **Node**: nodes represent complex concepts formed as a combined faceted view of the real world entities and/or individual objects described by the nodes' labels and by the labels of all their ascendant nodes. The meaning of a node in a classification is the set of documents which are about the complex concept represented by the node. The classification semantics of a given node is defined by the labels of the nodes on the path from the root to the node;
- **Classification**: the classification semantics of nodes defines the basis for how they are used for the classification of documents in a specific classification algorithm. The set of documents which are populated in a node defines the semantics of this node at the classification level. In the most general case, the classification semantics is defined by the nodes' labels, by the structure of the classification, and by the employed classification algorithm.

In the rest of this section we briefly describe and discuss two different Classifications: a relatively old librarian classification hierarchy, the Dewey Decimal Classification system (DDC), and an example from a modern web catalogue, namely the DMoz human-edited web directory.

Example 1 (DDC). Since the 19^{th} century, librarians have used DDC to organize vast amounts of books. DDC divides knowledge into ten different broad subject areas, called classes, numbered 000 - 999. Materials which are too general to belong to a specific group (encyclopedias, newspapers, magazines, etc.) are placed in the 000's. The ten main classes are divided up into smaller classes by several sets of subclasses. Smaller divisions (to subdivide the topic even further) are created by expanding each subclass and adding decimals if necessary. A small part of the DDC system is shown on Figure 1.

- 500 Natural Science and Mathematics
- 520 Astronomy and allied sciences
  - 523 Specific celestial bodies and phenomena
    - 523.1 The universe
    - 523.2 Solar system
    - 523.3 The Earth
    - 523.4 The moon
    - 523.5 Planets
      - 523.51 Mercury
      - 523.52 Venus
      - 523.53 Mars → 523.53HAN
      - . . .

Fig. 1. A part of the DDC system with an example of book classification

In DDC, the notation (*i.e.*, the system of symbols used to represent the classes in a classification system) provides a universal language to identify the class and related classes.

Before a book is placed on the shelves it is:

- classified according to the discipline matter it covers (given the Dewey number);
- some letters (usually three) are added to this number (usually they represent the author's last name);
- the number is used to identify the book and to indicate where the book will be shelved in the library. Books can be assigned a Dewey number corresponding to both leaf and non-leaf nodes of the classification hierarchy.

Since parts of DDC are arranged by discipline, not subject, a subject may appear in more than one class. For example, the subject "clothing" has aspects that fall under several disciplines. The psychological influence of clothing belongs to 155.95 as part of the discipline of psychology; customs associated with clothing belong to 391 as part of the discipline of customs; and clothing in the sense of fashion design belongs to 746.92 as part of the discipline of the arts. However, the final Dewey number associated to a book has to be unique and, therefore, the classifier needs to impose a classification choice among all the possible alternatives.

As an example, let's consider the Dewey number for the following book: Michael Hanlon, "*Pictures of Planet Mars*". A possible classification is Dewey number: 523.53 HAN and the classification choice for the book is shown in Figure 1.

The main properties of DDC are:

- the classification algorithm relies on the *get-specific* rule[4]: when you add a new object, get as specific as possible: dig deep into the classification schema,

[4] Look at http://docs.yahoo.com/info/suggest/appropriate.html to see how Yahoo! implements this rule.

looking for the appropriate sub-category; it is a bad practice to submit an object to a top level category, if one more specific exists. At present, the enforcement of such rule is left to the experience of the classifier;

– each object is placed in exactly one place in the hierarchy. As a result of this restriction, a classifier often has to choose arbitrarily among several reasonable categories to assign the classification code for a new document (see the above example for "clothing"). Despite the use of documents called "subject authorities", which attempt to impose some control on terminology and classification criteria, there is no guarantee that two classifiers make the same decision. Thus, a user, searching for information, has to guess the classifier's choice in order to decide where to look for, and will typically have to look in a number of places;
– each non-root node in the hierarchy has only one parent node. This enforces a tree structure on the hierarchy. □

Example 2 (DMoz). The Open Directory Project (ODP), also known as DMoz (for Directory.Mozilla.org, the domain name of ODP), is a multilingual open content directory of World Wide Web links owned by America Online that is constructed and maintained by a community of volunteer editors. ODP uses a hierarchical ontology scheme for organizing site listings. Data is made available through an RDF-like dump that is published on a dedicated download server. Listings on a similar topic are grouped into categories, which can then include smaller categories. As of March 10, 2006, the RDF held 5,272,517 listings and over 590,000 multilingual categories. DMoz powers the core directory services for the most popular portals and search engines on the Web, including AOL Search, Netscape Search, Google, Lycos, DirectHit, and HotBot, and hundreds of others [28].

In DMoz, as in DDC, objects (here mainly web links) are classified by a human classifier following the get-specific rule. In this classification hierarchy, an object can be often reached from different paths of the hierarchy, thus providing an efficient way for finding items of interest following different perspectives. This normally means that the object is classified in two (or more) nodes or that the object is classified in one node and there are "related" links from other nodes to the node where the object is classified.

In the following we present an example of classification for a software programming document in the DMoz web directory. The document title is "*Java Enterprise in a Nutshell, Second Edition*". In the DMoz web directory, reduced for sake of presentation, the example title can be found through two different search paths (see Figure 2), namely:

`Top/Business/Publishing and Printing/Books/Computers/`

`Top/Computers/Programming Languages/Java/` □

From the two specific examples we can see that Web catalogues are more flexible than classifications like Dewey. In fact, their aim is not to position a resource in a unique position, but rather to position it in such a way, that the

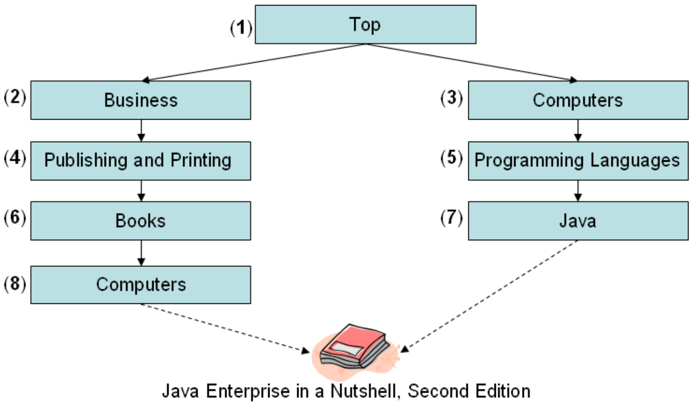

(1) Top
(2) Business
(3) Computers
(4) Publishing and Printing
(5) Programming Languages
(6) Books
(7) Java
(8) Computers
Java Enterprise in a Nutshell, Second Edition

Fig. 2. A part of the DMoz web directory

user, who navigates the catalogue, will be facilitated in finding appropriate or similar resources related to a given topic.

3 Why Formal Classifications?

There are many methodologies for how to classify objects into classification hierarchies. These methodologies range from the many rigorous rules "polished" by librarians during hundreds of years; to less strict, but still powerful rules of classification in a modern web directory[5]. What is relevant here, is that in all these different cases, a human classifier needs to follow a common pattern, which we summarize in four main steps. These steps are also followed when one searches for an object by means of classification browsing. The only difference is in that now the categories are inspected for where to find an object, and not where to put it. We discuss the four steps below, and we elucidate them on the example of the part of the DMoz web directory presented in Example 2.

1. Disambiguating labels. The challenge here is to disambiguate natural language words and labels. For example, the classifier has to understand that in the label of node n_7 (see Figure 2) the word "*Java*" has at least three senses, which are: an island in Indonesia; a coffee beverage; and an object-oriented programming language. Moreover, words in a label are combined to build complex concepts. Consider, for example, the labels at node n_4, *publishing and printing*, and n_5, *programming languages*. The combination of natural language atomic elements is used by the classifier to aggregate (like in *publishing and printing*) or disambiguate (like in *programming languages*) atomic concepts;

2. Disambiguating links. At this step the classifier has to interpret links between nodes. Namely, the classifier needs to consider the fact that each non-root node is "viewed" in the *context* of its parent node; and then specify the

[5] See, for instance, the DMoz classification rules at http://dmoz.org/guidelines/.

meanings of the nodes' labels. For instance, the meaning of the label of node n_8, *computers*, is bounded by the meaning of node n_6, *business books' publishing*;

3. Understanding classification alternatives. Given an object, the classifier has to understand what classification alternatives for this object are. For instance, the book "Java Enterprise in a Nutshell, Second Edition" might potentially be put in all the nodes of the hierarchy shown in Figure 2. The reason for this is that the book is related to both business and technology branches;

4. Making classification choices. Given the set of classification alternatives, the classifier has to decide, based on a predefined system of rules, where to put the given object. The system of rules may differ from classification to classification, but the get-specific rule is commonly followed. Note, that there may be more than one category for the classification. For instance, if the get-specific rule was used, then one would classify the above mentioned book into nodes n_7 and n_8, as they most specifically characterize the book.

Humans have proven to be very effective at performing steps 1 and 2, as described above. However, there are still some challenges to be addressed. The main challenge in step 1 is dealing with the ambiguities introduced by multiple possibilities in meaning. One source of this is in that labels contain many conjunctions "and"'s and "or"'s, whereas they actually mean inclusive disjunction, *i.e.*, either the first conjunct, or the second, or both. For instance, the phrase "publishing and printing" means either publishing, or printing, or both. Apart from the conjunctions, multiple possibilities are introduced also by punctuation marks denoting enumeration (*e.g.*, the comma), and by words' senses (recall the various senses of the word "Java"). It has been shown, that cognitive reasoning with the presence of multiple possibilities (distinctions) is an error-prone task for humans [14]. For instance, even if DMoz labels are short phrases, consisting, on average, of 1.81 tokens, they contain 0.23 conjunctions per label; and average polysemy for nouns and adjectives is 3.72 per word[6]. Conjunctions, punctuation, and words' senses count together to 3.79 possibilities in meaning per label.

Statistics category	**Value**
Total English labels	477,786
Tokens per label, avg.	1.81
Total links classified in English labels	3,047,643
Duplicate links, % from the total	10.70%
Nouns and adjectives polysemy, avg.	3.72
"and"'s and *"or"*'s per label, avg.	0.23
Total disjunctions per label, avg.	3.79
Root-to-leaf path length, avg.	7.09
Branching factor, avg.	4.00

Table 1. DMoz statistics

[6] A summary of the statistical analysis we performed on DMoz is reported in Table 1. In our analysis we excluded branches leading to non-English labels, such as `Top/World/` or `Top/Kids and Teens/International/`.

The challenge of step 2 is that the classifier may need to follow a long path of nodes in order to figure out a node's meaning. It has two consequences: first, the classifier needs to deal with the growing complexity in ambiguity introduced by each new label in the path; and, second, the classifier has to consider each new label in the context of the labels of the ancestor nodes, and, thus, partly resolve the ambiguity. Note, that, for instance, the average length of a path from the root to a leaf node in DMoz is 7.09.

Steps 3 and 4 is where the real problems for humans begin. Even with classifications of average size, it is not easy to find all the classification alternatives. Consider, for instance, how many times you did not find an email in your own mail directory. With large classifications this task becomes practically impossible. For instance, think about possible classification alternatives in DMoz, which has 477,786 English categories. Thus, at step 3, a human classifier may not be able to enumerate all the possible classification alternatives for an object.

Step 4 requires abundant expertise and profound methodological skills on the side of the classifier. However, even an expert makes subjective decisions, what leads, when a classification is populated by several classifiers, to nonuniform, duplicate, and error-prone classification. If the get-specific rule is used, then the classifier has to parse the classification tree in a top-down fashion, considering at each parent node, which of its child nodes is appropriate for the classification or for further consideration. The higher the average branching factor in the classification tree, the higher the probability of that two different classifiers will find appropriate two different sibling nodes at some level in the tree. This is because the difference in meaning of the two nodes may be vague or, vice versa, because the two nodes have distinct meanings and they represent different facets of the object being classified. In this latter case the classifiers may simply follow different perspectives (facets) when classifying the object (recall the example with "clothing" from Example 1). Note, that even if DMoz encourages the classification of a Web page in a single category, among 3,047,643 links (classified in English labels), about 10.70% are classified in more than one node[7]. And, about 91.36% of these are classified in two different nodes. This is not surprising given that DMoz is populated by more than 70,000 classifiers, and that it has average branching factor of 4.00.

Given all the above described complexity, humans still outperform machines in natural language understanding tasks [25], which are the core of steps 1 and 2. Still, the availability of electronic repositories that encode world knowledge (*e.g.*, [16, 19]), and powerful natural language processing tools (*e.g.*, [22, 16]) allows the machines to perform these steps reasonably well. Moreover, machines can be much more efficient and effective at steps 3 and 4, if the problem is encoded in a formal language, which is what we propose to do in our approach.

[7] We identified duplicate links by exact equivalence of their URLs.

4 Disambiguating labels

To support an automatic classifier in step 1 of the classification task (as described in the previous section), we propose to convert classifications into a new structure, which we call *Formal Classification* (FC), more amenable to automated processing.

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

FCs and classifications are related in the sense that a FC is a *formalized* copy of a classification. In other words, a FC has the same structure as the classification, but it encodes the classification's labels in a formal language (*i.e.*, L^C), capable of encapsulating, at the best possible level of approximation, their classification semantics. In the following we will call L^C, the *concept language.* We use a *Propositional* Description Logic language for several reasons:

- since natural language labels are meant to describe real world entities, and *not* actions, performed on or by entities, or relations between entities, the natural language labels are mainly constituted of noun phrases; and, therefore, there are very few words which are verbs. This makes it very suitable to use a Description Logic (DL) language as the formal language, as DLs are a precise notation for representing noun phrases [1];
- The set-theoretic semantics of DL allows us to translate syntactic relations between words in a label into the logical operators of DL, preserving, at the best possible level of approximation, the classification semantics of the label. Below in this section we provide concrete examples and argumentation of the translation process and its principles;
- a formula in L^C can be converted into an equivalent formula in a propositional logic language with the boolean semantics. Thus, a problem expressed in L^C can be converted into a *propositional satisfiability problem*[8].

Converting classifications into FCs automates step 1, as described in Section 3. In our approach we build on the work of Magnini et. al. [17]. We translate a natural language label into an expression in L^C by means of mapping different parts of speech (POSs), their mutual syntactic relation, and punctuation to the classification semantics of labels. We proceed in three steps, as discussed below:

1. Build atomic concepts. Senses of common nouns and adjectives become atomic concepts of L^C, whose interpretation is the set of documents about the entities, which are denoted by the nouns, or which possess the qualities denoted by the adjectives. We enumerate word senses using WordNet [19], and we write

[8] For translation rules from a Propositional Description Logic to a Propositional Logic, see [6].

`x#i` to denote an atomic concept corresponding to the i^{th} sense of the word `x` in WordNet. For instance, `programming#2` is an atomic concept, whose interpretation is the set of documents which are about computer programming; and the atomic concept `red#1` denotes the set of documents which are about red entities, *e.g.*, red cats or red cars. Proper nouns become atomic concepts of L^C, whose interpretation is the set of documents about the individual objects, denoted by the proper nouns. They may be long expressions, denoting names of people, movies, music bands, and so on. Some examples are the movie *"Gone with the Wind"*, and the music band *"The Rolling Stones"*. Apart from proper nouns, multi-words are recognized, and each their distinct sense becomes an atomic concept in the concept language[9]. Words which are not found in WordNet are assigned distinct atomic concepts, which are uniquely identified by the string representation of the words (case ignored). Put it differently, words which are not found in WordNet and whose string representations are equal, are assigned the same atomic concept. As the output of this step, each word (multi-word, or proper noun) is associated with one or more atomic concepts, whereas many of them are associated with the corresponding senses from WordNet.

2. Word sense disambiguation. At this step we discard irrelevant word senses and corresponding atomic concepts. In part, we follow the approach proposed in [17]. Namely, if there is a relation found in WordNet between any two senses of two words in a label, then these senses are retained and other unrelated senses are discarded. The relation looked for in WordNet is synonymy, hypernymy (*i.e.*, the "kind-of" relation, *e.g.*, *car* is a kind of *vehicle*), or holonymy (*i.e.*, the "part-of" relation, *e.g.*, *room* is a part of a *building*). If no relation found, then we check if a relation exists between two WordNet senses by comparing their glosses as proposed in [9]. In this respect we go beyond what is suggested in [17].

3. Build complex concepts. Complex concepts are built from atomic concepts as follows: first, we build words' formulas as the logical disjunction ($\sqcup$) of atomic concepts corresponding to their senses (remaining after step 2), and we write `x`$*$ to denote the disjunction of the (remaining) senses of word `x`. For instance, the noun *"Programming"* becomes the concept (`programming#1` $\sqcup$ `programming#2`), whose interpretation is the set of documents which are about event scheduling and/or about computer programming. Second, labels are chunked, *i.e.*, divided into sequences of syntactically correlated parts of words. We then translate syntactic relations to the logical connectives of L^C following a precise pattern. Let us consider a few examples.

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. The interpretation of the resulting concept is the set of documents which are

[9] Because of their negligibly small presence, we do not consider verbs. We neither consider articles, numerals, pronouns and adverbs. However, their share in the labels of real classifications is reasonably small. When such words are found, they are just omitted from the label.

about the real world entities denoted by all the nouns, and which possess qualities, denoted by all the adjectives. For instance, the phrase *"long cold winter blizzard"* is translated into the concept $\texttt{long}* \sqcap \texttt{cold}* \sqcap \texttt{winter}* \sqcap \texttt{blizzard}*$.

Prepositions are also translated into the conjunction. The intuition is that prepositions denote some commonality between the two objects they relate; and, in terms of the classification semantics, this "commonality" can be approximated to the set of documents which are about both objects. For instance, the following phrases: *"books of magic"*, *"science in society"*, and *"software for engineering"*, they all denote what the two words, connected by the prepositions, have in common.

Coordinating conjunctions "and" and "or" are translated into the logical disjunction. For instance, *"flights or trains"* and *"animals and plants"* become $\texttt{flight}* \sqcup \texttt{train}*$ and $\texttt{animal}* \sqcup \texttt{plant}*$ respectively. Punctuation marks such as the period (.), the coma (,) and the semicolon (;) are also translated into the logical disjunction. For instance, the phrase *"metro, bus, and trolley"* is converted into the concept $\texttt{metro}* \sqcup \texttt{bus}* \sqcup \texttt{trolley}*$.

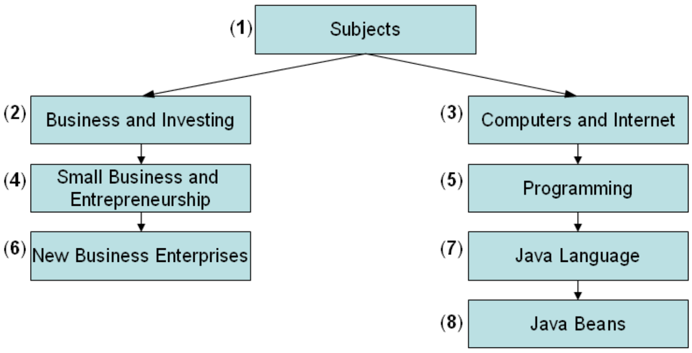
Words and phrases denoting exclusions, such as "excluding", "except", "but not", are translated into the logical negation ($\neg$). For instance, label *"runners excluding sprinters"* becomes the concept $\texttt{runner}* \sqcap \neg \texttt{sprinter}*$. However, since they are meant to describe what "there is" in the world, and not what "there isn't", labels contain very few such phrases.

The use of logical connectives, as described above but with the exception of prepositions, allows it to *explicitly* encode the classification semantics of labels. In other words, the interpretation of the resulting formulas explicitly represents the set of documents which are about the corresponding natural language labels. The translation of prepositions is an approximation, as they may encode meaning, which only partly can be captured by means of the logical conjunction. For example, *"life in war"* and *"life after war"* will collapse into the same logical formula, whereas the classification semantics of the two labels is different.

Example 3 (Disambiguating labels in a web directory). Let us consider how the label of node n_2 in the part of the Amazon book directory shown in Figure 3 can be disambiguated. The label consists of three tokens: "business", "and", and "investing", whereas the first and the last tokens are recognized as nouns, and the second token is recognized as a coordinating conjunction. The noun "business" has nine, and the noun "investing" has one sense in WordNet. Therefore, after step 1 we have two words with associated atomic concepts as shown below:

business (`business#1`, `business#2`, ..., `business#9`), and
investing (`investing#1`)

At step 2, the senses of the two words are compared, and it is found that `investing#1` (defined as "the act of investing; laying out money or capital in an enterprise with the expectation of profit") is a second level hyponym of `business#2` (defined as "the activity of providing goods and services involving financial and commercial and industrial aspects"). Therefore, the second sense (and the atomic concept associated with it) of the word "business" is retained and all the others are discarded.



(1) Subjects
(2) Business and Investing
(3) Computers and Internet
(4) Small Business and Entrepreneurship
(5) Programming
(6) New Business Enterprises
(7) Java Language
(8) Java Beans

Fig. 3. Amazon Book Directory

At step 3 we build a complex concept by considering the fact that the coordinating conjunction "and" is translated into the logical disjunction. We have therefore:

$l_2^F = \texttt{business\#2} \sqcup \texttt{investing\#1}$ □

In order to estimate how much of the information encoded into the labels of a real classification can be captured using our approach, we have conducted a grammatical analysis of the DMoz classification. For doing this, we have used the OpenNLP Tools tokenization and POS-tagging library [22], which reports to achieve more than 96% accuracy on unseen data[10]. In Table 2 we show the POS statistics of the DMoz tokens. Note, that about 77.59% of the tokens (nouns and adjectives) become concepts, and about 14.69% (conjunctions and prepositions) become logical connectives of L^C. WordNet coverage for common nouns and adjectives found in DMoz labels is quite high, and constitutes 93.12% and 95.01% respectively. Detailed analysis of conjunctions and prepositions shows that about 85.26% of them are conjunctions "and", and about 0.10% are conjunctions "or". In our analysis we found no words or phrases which would result into the logical negation. Only about 4.56% of tokens are verbs and adverbs in all their forms.

Note, that the propositional nature of L^C allows us to *explicitly* encode about 90.13% of the label data in DMoz (*i.e.*, nouns, adjectives, conjunctions "and" and "or"). Still, this is a rough understated estimation, as we did not take into account multi-word common and proper nouns. In fact, a manual analysis of the longest labels, as well as of the ones with verbs, shows that the majority of these labels represents proper names of movies, games, institutions, music bands, etc.

[10] The tool may not function at its expected performance on special data as short labels because it has been trained on well-formed natural language sentences.

POS	**Share**
Common nouns	71.22%
Proper nouns	0.18%
Adjectives	6.19%
Conjunctions and prepositions	14.69%
Verbs, adverbs	4.56%
Other POSs	3.16%

Table 2. DMoz token statistics

5 Disambiguating Edges

As discussed in Section 2, the classification semantics of nodes codifies the fact that child nodes are always considered in the context of their parent nodes. This means that the meaning of a non-root node is the set of documents, which are about its label, and which are *also* about its parent node. We encode the classification semantics of nodes into their property which we call *concept at a node* [6]. We write C_i to refer to the concept at node n_i, and we define this notion as:

$$C_i = \begin{cases} l_i^F & \text{if } n_i \text{ is the root of } FC \\ l_i^F \sqcap C_j & \text{if } n_i \text{ is not the root of } FC\text{, where } n_j \text{ is the parent of } n_i \end{cases} \tag{1}$$

There may be two meaningful relations between the concept at a parent node, and the label of its child node, as represented in Figure 4:

- in case (a) the label of the child node is about the parent node, but it is also about something else. In this case the parent node *specializes* the meaning of the child node by bounding the interpretation of the child node's label with the interpretation of the concept at the parent node. For instance, think about a classification where the root node is labeled “Italy” and its sole child node is labeled “Pictures” (see Figure 4a). A human can understand that the meaning of the child node is “pictures of Italy” and not “pictures of Germany”, for example. In the corresponding FC this knowledge is encoded into the concept at node $C_2 = \texttt{italy} * \sqcap \texttt{picture}*$;
- in case (b) the child node represents a *specification* of the parent node, and their relation can be, for instance, the “is-a” or the “part-of” relation. Note, that in this case, differently from case (a), the parent node does not influence the meaning of the child node. Suppose that in the previous example the child node's label is “Liguria” (see Figure 4b). A human can understand that the meaning of this node is the same as of its label. In the corresponding FC this knowledge is encoded into the concept at node $C_2 = \texttt{italy} * \sqcap \texttt{liguria}*$, which can be simplified to $C_2 = \texttt{liguria}\#1$, taking into account the fact that both words “Italy” and “Liguria” have only one sense in WordNet, and given that the corresponding axiom ($\texttt{liguria}\#1 \sqsubseteq \texttt{italy}\#1$) is memorized in some background knowledge base.

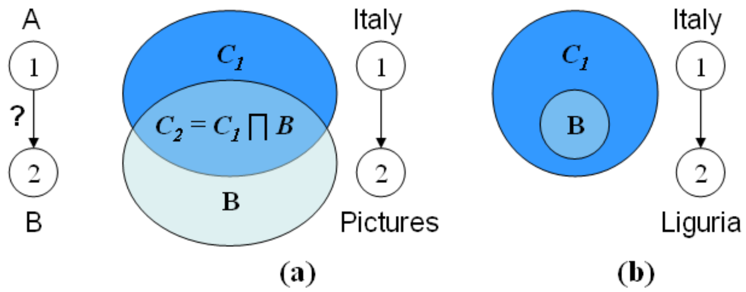


Fig. 4. Edge semantics in FCs

Note, that applying Equation 1 recursively, we can compute the concept at any non-root node n_i as the conjunction of the labels of all the nodes on the path from the root of the FC, n_1, to n_i. This corresponds to how the notion of concept at a node is defined in [7], namely:

$$C_i = l_1^F \sqcap l_2^F \sqcap \ldots \sqcap l_i^F \tag{2}$$

The concept at a node encodes, but only to a certain extent, the path from the root to the node. In fact, there may be more than one way to reconstruct a path from a concept. Atomic concepts in a concept at a node may be "distributed" differently among different number of nodes, which, in turn, may have a different order in the path. The number of nodes may range from one, when the concept at the node is equivalent to the node's label, to the number of clauses in the CNF equivalent of the concept. However, all the possible paths converge to the same semantically equivalent concept. Consider, for instance, node n_8 in the classification shown in Figure 2. The two paths below will converge to the same concept for the node[11]:

```
top/Publishing and Printing/Business Books/Computers/
top/Business/Publishing and Printing/Computer Books/
```

We use the notion of concept at a node to define another structure which we call *Normalized Formal Classification* (NFC).

Definition 3 (Normalized Formal Classification) *A Normalized Formal Classification is a rooted tree $NFC = \langle N, E, L^N \rangle$ where N is a finite set of nodes, E is a set of edges on N, and L^N is is a finite set of labels expressed in L^C, such that for any node $n_i \in N$, there is one and only one label $l_i^N \in L^N$ and $l_i^N \equiv C_i$.*

Note, that the main characteristic of NFCs, that distinguishes them from FCs, is the fact that labels of child nodes are always more specific than the labels of their parent nodes. Interestingly, if a taxonomic classification, *i.e.*, a

[11] For sake of presentation we give these examples in natural language.

classification with only "is-a" and "part-of" links, is converted into a FC, then the latter is also a NFC.

Apart from this, NFCs have a number of important properties relevant to classifications, discussed below:

- the interpretation of nodes' labels is the set of documents which *can* be classified in these nodes. We underline the "can" since, as we discuss in the next section, documents which *are* actually classified in the nodes are often a subset of the interpretation of the labels in NFCs;
- two nodes, representing in a classification the same real world entities, will have semantically equivalent labels in the NFC. This fact can be exploited for automatic location and/or prevention of adding of such "duplicate" nodes. As an example, consider the different paths that lead to the same concept as described earlier in this section;
- NFCs are full-fledged *lightweight ontologies*, suitable for the automation of the core classification tasks, such as document classification and query answering.

The consideration of the path from the root to any given node allows us not only to compute the concept at that node which leads to the properties discussed above, but also to further disambiguate the senses of the words in its label taking into account the context of the path and, accordingly, to delete corresponding atomic concepts from its label in the NFC. In this task we apply exactly the same technique as the one discussed in Section 4 for sense disambiguation in labels with the only difference in that now all the remaining words' senses in *all* the labels on the path to the root are compared.

Example 4 (Disambiguating edges in a web directory). Recall the example of the part of the DMoz directory shown in Figure 2 and let us see how the concept at node n_7 can be computed. Remember the three senses of the word "java" (which is the label of n_7) discussed earlier in the paper, and consider the parent node's label, "programming languages", which is recognized as a multi-word with only one sense whose gloss is "a language designed for programming computers". Comparing this gloss with the gloss of the third sense of the word "java" (defined as "a simple platform-independent object-oriented programming language...") results that the similarity of the two glosses exceeds a certain threshold and, therefore, a relation is found between these two senses. We therefore compute the concept at node n_7 as:

$l_7^N = (\texttt{computer\#1} \sqcup \texttt{computer\#2}) \sqcap \texttt{programming\_language\#1} \sqcap \texttt{java\#3} \Box$

6 Document classification

Before some document d can be classified, it has to be assigned an expression in L^C, which we call the *document concept*, written C^d. The assignment of concepts to documents is done in two steps: first, a set of n keywords is retrieved from the document using text mining techniques (see, for example, [23]); the keywords are then converted into a concept by means of the conjunction of the formulas representing the keywords, translated to L^C as discussed in Section 4.

The interpretation of the document concept of any document includes the document itself (*i.e.*, $d \in (C^d)^{\mathcal{I}}$) as well as other documents which have equivalent or more specific document concepts. In Figure 5 we show an example of how a document and the interpretation of its concept can be interrelated. There, the interpretation of the concept of document d_1, C^{d_1}, includes d_1 itself, d_2, whose concept is equivalent to C^{d_1}, and d_3, whose concept C^{d_3} is more specific than C^{d_1}.

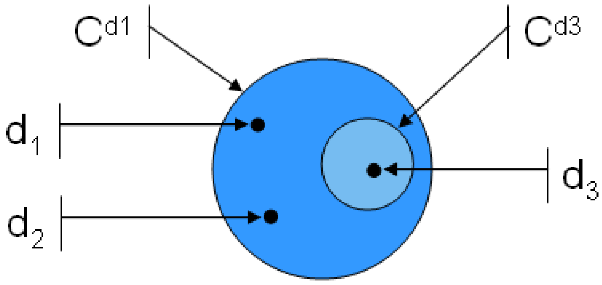


Fig. 5. Document concept

We say that node n_i is a *classification alternative* for the classification of some document d with concept C^d, if $C^d \sqsubseteq l_i^N$. In fact, if this relation holds, and given that $d \in (C^d)^{\mathcal{I}}$, it follows that $d \in (l_i^N)^{\mathcal{I}}$, *i.e.*, document d belongs to the set of documents which can be classified in n_i. For any given document d and a NFC, we compute the set of classification alternatives for d in the NFC as follows:

$$A(C^d) = \{n_i | C^d \sqsubseteq l_i^N\} \tag{3}$$

By computing Equation 3, we can automate step 3 described in Section 3. The automation of step 4, *i.e.*, making classification choices, depends on what classification algorithm is used. Below we show how it can be automated for some set A of classification alternatives if the get-specific rule (see Section 3) is used:

$$C(A) = \{n_i \in A | \nexists n_j \in A \ (i \neq j), \ \textit{such that} \ l_j^N \sqsubseteq l_i^N\} \tag{4}$$

The set $C(A)$ includes all the nodes in the NFC, whose labels are more general than the document concept, and more specific among all such labels. As labels of child nodes in NFCs are always more specific than the labels of their parent nodes, $C(A)$ consists of nodes which lie as low in the CNF tree as possible, and which are still classification alternatives for the given document. Note, that the get-specific rule applies not only to nodes located on the same path from the root, but also to nodes located in different branches. For instance, a document about computer graphics will *not* be classified in the node `top/computers/` if the more specific node `top/arts/computers/` exists.

Formula 4 implies that the set of documents classified in some node n_i may (and, in most cases will) be a subset of the interpretation of its label l_i^N. In fact, the set of documents which are *actually* classified in n_i excludes those, which belong to the interpretation of labels, which are more specific than l_i^N. We encode this set in the concept l_i^C which univocally identifies the set of documents classified in node n_i, and, therefore, defines the classification level semantics of n_i in the NFC. We compute l_i^C as follows:

$$l_i^C = l_i^N \sqcap \neg \bigsqcup (l_j^N | j \neq i, l_j^N \sqsubseteq l_i^N) \tag{5}$$

Noteworthy, the concepts which represent the classification semantics at the three levels discussed in Section 2 are related by the subsumption relation as shown below:

$$l_i^C \sqsubseteq l_i^N \sqsubseteq l_i^F \tag{6}$$

Computing Equations 3, 4 and 5 requires verifying whether the subsumption relation holds between two formulas in L^C. In a more general case, 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. Relation $rel(A, B)$ is the formula encoding the relation between concepts A and B translated to the propositional logic according to the rules proposed in [6]. As discussed in Section 4, 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. These relations can be translated into axioms, which *explicitly* capture the classification semantics of the relation that holds between the two senses. Thus, for instance, the set of documents which are about *cars* is a subset of the set of documents which are about a hypernym of the word "car", *vehicle*. 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.

Example 5 (Document classification). As en example, recall the classification in Figure 2, and suppose that we need to classify the book: *"Java Enterprise in a Nutshell, Second Edition"*, whose concept is `java`#3 $\sqcap$ `enterprise`#2 $\sqcap$ `book`#1. It can be shown, by means of propositional reasoning, that the set of classification alternatives includes all the nodes of the corresponding NFC. For sake of presentation we provide concrete formulas only for nodes n_7 and n_8, whose labels are:

$l_7^N = \texttt{computer} * \sqcap \texttt{programming} * \sqcap \texttt{language} * \sqcap \texttt{java}*$, and
$l_8^N = \texttt{business} * \sqcap (\texttt{publishing} * \sqcup \texttt{printing}*) \sqcap \texttt{publishing} *$
$\sqcap \texttt{books} * \sqcap \texttt{computer}*$.

We can extract the following knowledge from WordNet: the programming language Java is a kind of programming languages, and it is a more specific concept than computer is; books are related to publishing; and enterprise is a more specific concept than business is. We encode this knowledge in the following set of axioms:

$a_1 = (\texttt{java\#3} \sqsubseteq \texttt{pr\_language\#1})$; $a_3 = (\texttt{book\#1} \sqsubseteq \texttt{publishing\#1})$;
$a_2 = (\texttt{java\#3} \sqsubseteq \texttt{computer\#1})$; $a_4 = (\texttt{enterprise\#1} \sqsubseteq \texttt{business\#2})$.

Next, we translate the axioms and the labels into the propositional logic language, and we verify if the condition in Formula 3 holds for the two labels by constructing two formulas, following the pattern of Equation 7, as shown below:

$$(a_2 \wedge a_3 \wedge a_4) \rightarrow (C^d \rightarrow l_8^N); \quad (a_1 \wedge a_2) \rightarrow (C^d \rightarrow l_7^N).$$

We then run a SAT solver on the above formulas, which shows that they are tautologies. It means that both nodes n_7 and n_8 are classification alternatives for the classification of the book. Among all the classification alternatives, only these two nodes satisfy the get-specific rule, and, therefore, they are the final classification choices for the given book. The latter can be shown by computing Equation 4 by means of propositional reasoning. □

Note, that the edges of the NFC are *not* considered in document classification. In fact, the edges of the NFC become redundant, as their information is implicitly encoded in the labels. Note that given a set of labels, there may be several ways to reconstruct the set of edges of a NFC. However, from the classification point of view, all these NFCs are equivalent, as they classify documents *identically.* In other words, nodes with equivalent labels are populated with the same set of documents.

7 Query answering

When the user searches for a document, she defines a set of keywords or a phrase, which is then converted into an expression in L^C using the same techniques as discussed in Section 4. We call this expression, a *query concept*, written C^q. We define the answer A^q to a query q as the set of documents, whose concepts are more specific than the query concept C^q:

$$A^q = \{d | C^d \sqsubseteq C^q\} \tag{8}$$

Searching directly on all the documents may become prohibitory expensive as classifications may contain thousands and millions of documents. NFCs allow us to identify the maximal set of nodes which contain *only* answers to a query, which we call, the *sound classification answer* to a query (written N^q_s). We compute N^q_s as follows:

$$N^q_s = \{n_i | l^N_i \sqsubseteq C^q\} \tag{9}$$

In fact, as $C^d \sqsubseteq l^N_i$ for any document d classified in any node $n_i \in N^q_s$ (see Formulas 3 and 4), and $l^N_i \sqsubseteq C^q$ (as from Formula 9 above), then $C^d \sqsubseteq C^q$. Thus, all the documents classified in the set of nodes N^q_s belong to the answer A^q (see Formula 8).

We extend N^q_s by adding nodes, which constitute the classification set of a document d, whose concept is $C^d \equiv C^q$. We call this set, the *query classification set*, written Cl^q; and we compute it following Formula 4. In fact, nodes in Cl^q may contain documents satisfying Formula 8, for instance, documents whose concepts are equivalent to C^q.

Thus, for any query q, the user can compute a sound query answer A^q_s by taking the union of two sets of documents: the set of documents which are classified in the set of nodes N^q_s (computed as $\{d \in n_i | n_i \in N^q_s\}$); and the set of documents which are classified in the nodes from the set Cl^q and which satisfy Formula 8 (computed as $\{d \in n_i | n_i \in Cl^q, C^d \sqsubseteq C^q\}$). We have therefore:

$$A^q_s = \{d \in n_i | n_i \in N^q_s\} \cup \{d \in n_i | n_i \in Cl^q, C^d \sqsubseteq C^q\} \tag{10}$$

Under the given definition, the answer to a query is not restricted to the documents classified in the nodes, whose concepts are equivalent to the concept of the query. Documents from nodes, whose concepts are more specific than the query concept are also returned. For instance, a result for the above mentioned query may also contain documents about Java beans.

Note that the proposed approach to query answering allows it to search by comparing the *meaning* of the query, of the nodes, and of the documents by means of propositional reasoning on their formulas. For instance, in our approach documents about "Ethiopian villages" will be returned as the result of the user searching for "African settlements". This makes a fundamental difference with the standard search techniques based on information retrieval and word indexing. These are based on exact term matching and on ranking the results following relative term frequencies; no query semantics is taken into account in these approaches.

Example 6 (Query answering). Suppose that the user defines a query to the Amazon NFC which is translated to the following concept: $C^q = \texttt{java}\#3 \sqcup \texttt{cobol}\#1$, where $\texttt{cobol}\#1$ is "a common business-oriented language". It can be shown, that $N^q_s = \{n_7, n_8\}$ (see Figure 3 for the Amazon classification). However, this set does not include node n_5, which contains the book "Java for COBOL Programmers (2nd Edition)". The relevance of node n_5 to the query can be identified by computing the query classification set for query q, which in fact consists of the single node n_5, *i.e.*, $Cl^q = \{n_5\}$. However, n_5 may also contain irrelevant documents, which are excluded from the query result by computing

Formula 10. □

For what regards the complexity of the query answering and document classification algorithms, since both are reduced to the validity problem, they represent co-NP-complete problems. However, as discussed in [10], in most of the cases the time complexity is (or can be reduced to) polynomial.

8 Related Work

In our work we adopt the notion of the concept at a node as first introduced in [6] and further elaborated in [7]. Moreover, the notion of label of a node in a FC, semantically corresponds to the notion of the concept of a label introduced in [7]. In [7] these notions play the key role in the identification of semantic mappings between nodes of two schemas. In this paper, these are the key notions needed to define NFCs which can be used for document classification and query answering in a completely new way.

This work as well as the work in [6, 7] mentioned above is crucially related and depends on the work described in [2, 17]. In particular, in [2], the authors, for the first time, introduce the idea that in classifications, natural language labels should be translated in logical formulas, while, in [17], the authors provide a detailed account of how to perform this translation process. The work in [6, 7] improves on the work in [2, 17] by understanding the crucial role that concepts at nodes have in matching heterogeneous classifications and how this leads to a completely new way to do matching. This paper, for the first time, recognizes the crucial role that the ideas introduced in [2, 6, 7, 17] have in the construction of a new theory of classification, and in introducing the key notion of FC.

In [24], the authors propose a very similar approach to converting natural language labels in classifications to concept language formulas. Our approach is different in at least two respects. First, the target application in [24] is matching, whereas in the present paper we focus on document classification and query answering. Second, DL roles are used in [24] to encode the meaning of labels. The advantage of our approach is in that, while using a simpler subset of DLs, we are able to explicitly capture the semantics of a large portion of the label data in a real classification.

A related approach to converting generic thesauri and related resources from their native format to RDF(S) and OWL was recently proposed in [30]. In that work, the authors discuss a set of guidelines for how to perform a conversion of syntactic elements (*e.g.*, structure, entity names) and semantic elements (*e.g.*, property types) from the native format to RDF(S) and OWL. Our approach is different because it aims at extracting semantic information implicitly encoded in the classification schema (by using NLP) in order to enable the automation through reasoning, and not to perform meaning-preserving structure conversion from one format to another in order to improve interoperability as it is the case in [30].

The approach presented in this paper can potentially allow for automatic classification of objects into user-defined hierarchies with no or little intervention of the user. This, in turn, provides the user with less control over the classification process. Therefore, for any classified object, the user may want to be given explanatory details for why the object was classified in one and not another way. From this perspective, the work presented in [18] is particularly relevant to our approach, as it allows to monitor the reasoning process and present the user with the trace of the main reasoning steps which led to the obtained conclusion.

A lot of work in information theory, and more precisely on formal concept analysis (see for instance [31]) has concentrated on the study of concept hierarchies. NFCs are very similar to what in formal concept analysis are called concept hierarchies with no attributes. The work in this paper can be considered as a first step towards providing a computational theory of how to transform the "usual" natural language classifications into concept hierarchies.

The document classification and query answering algorithms, proposed in this paper, are similar to what in the Description Logic (DL) community is called *realization* and *retrieval* respectively. The fundamental difference between the two approaches is in that in the DL approach the underlying structure for the classification is *not* predefined by the user, but is built *bottom-up* from atomic concepts by computing the partial order on the subsumption relation. Interested readers are referenced to [12], where the authors propose sound and complete algorithms for realization and retrieval. In our approach, classifications are built in the *top-down* fashion by the user and in the way decided by the user. Therefore, after their conversion to NFCs, the order of labels imposed by the edges does not necessarily represent the partial order, which requires algorithms and optimizations different from those used in the DL approach.

In Computer Science, the term *classification* is primarily seen as the *process* of arranging a set of objects (*e.g.*, documents) into *categories* or *classes*. There exist a number of different approaches which try to build classifications *bottom-up*, by analyzing the contents of documents. These approaches can be grouped in two main categories: *supervised classification*, and *unsupervised classification*. In the former case, a small set of training examples needs to be pre-populated into the categories in order to allow the system to automatically classify a larger set of objects (see, for example, [5, 20]). The latter approach uses various machine learning techniques to classify sets of objects (*e.g.*, data clustering [13]), and it usually has much lower precision than the former one. There exist some approaches that apply (mostly) supervised classification techniques to the problem of documents classification into hierarchies [15, 26]. The classifications built following our approach are better and more natural than those built following these approaches. They are in fact constructed *top-down*, as chosen by the user and not constructed bottom-up, as they come out of the document analysis. Moreover, in our approach there is no need to have a pre-populated set of documents in order to classify another, larger set. Last but not least, our approach has the ability to classify documents one-by-one and not only in sets by contrasting reciprocal properties of documents in the set as it is done in the above approaches.

9 Future Work and Conclusions

In this paper we have introduced the notion of Formal Classification, namely of a classification where labels are written in a propositional concept language. Formal Classifications have many advantages over standard classifications all deriving from the fact that formal language formulas can be reasoned about far more easily than natural language sentences. In this paper we have highlighted how this can be done to perform automatic document classification and semantics-aware query answering. Our approach has the potential, in principle, to allow for the automatic classification of (say) the Yahoo! documents into the Yahoo! directories.

The primary goal of this paper is to present the theory of how to translate classifications into Lightweight Ontologies and how they can be used for the automation of essential tasks on classifications. Therefore, large-scale experiments are out of the scope of the present paper even if the first experiments show the proof of concept of our approach. However, the results shown in the related approach of semantic matching allow us to expect promising results for our approach. In fact, in both cases, the core underlying technologies are NLP (applied to classifications) and propositional reasoning, which gives us a reason to think that the results of the two approaches will be comparable. Note that semantic matching outperforms other similar approaches in some primary indicators [8].

However much more can be done. Our future work includes testing the feasibility of our approach with very large sets of documents, such as those classified in the DMOZ directory, as well as the development of a sound and complete query answering algorithm. Apart from this, we will explore the ways of combining the proposed approach to query answering with those based on document indexing and keywords-based search. Our future work also includes a study of how dynamic changes made to classifications can be fully supported at the level of the corresponding FCs and NFCs.

References

1. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook : Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. 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.
3. Lois Mai Chan and J.S. Mitchell. *Dewey Decimal Classification: A Practical Guide.* Forest P.,U.S., December 1996.
4. eCl@ss: Standardized Material and Service Classification. see http://www.eclass-online.com/.

5. G.Adami, P.Avesani, and D.Sona. Clustering documents in a web directory. *In Proceedings of Workshop on Internet Data management (WIDM-03)*, 2003.
6. F. Giunchiglia and P. Shvaiko. Semantic matching. *Knowledge Engineering Review*, 18(3):265–280, 2003.
7. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match: An algorithm and an implementation of semantic matching. *In Proceedings of ESWS'04*, 2004.
8. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. Semantic schema matching. In *CoopIS*, 2005.
9. F. Giunchiglia and M. Yatskevich. Element level semantic matching. In *Meaning Coordination and Negotiation workshop, ISWC*, 2004.
10. F. Giunchiglia, M. Yatskevich, and E. Giunchiglia. Efficient semantic matching. In *ESWC*, 2005.
11. A.D. Gordon. *Classification.* Monographs on Statistics and Applied Probability. Chapman-Hall/CRC, Second edition, 1999.
12. Ian Horrocks, Lei Li, Daniele Turi, and Sean Bechhofer. The instance store: DL reasoning with large numbers of individuals. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, pages 31–40, 2004.
13. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.
14. Johnson-Laird. *Mental Models.* Harvard University Press, 1983.
15. Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In Douglas H. Fisher, editor, *Proceedings of ICML-97, 14th International Conference on Machine Learning*, pages 170–178, Nashville, US, 1997. Morgan Kaufmann Publishers, San Francisco, US.
16. Douglas B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, 1995.
17. Bernardo Magnini, Luciano Serafini, and Manuela Speranza. Making explicit the semantics hidden in schema models. *In: Proceedings of the Workshop on Human Language Technology for the Semantic Web and Web Services, held at ISWC-2003, Sanibel Island, Florida*, October 2003.
18. D. L. McGuinness, P. Shvaiko, F. Giunchiglia, and P. Pinheiro da Silva. Towards explaining semantic matching. In *International Workshop on Description Logics at KR'04*, 2004.
19. George Miller. *WordNet: An electronic Lexical Database.* MIT Press, 1998.
20. Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.
21. Natalya F. Noy. Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec.*, 33(4):65–70, 2004.
22. The OpenNLP project. See http://opennlp.sourceforge.net/.
23. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
24. Luciano Serafini, Stefano Zanobini, Simone Sceffer, and Paolo Bouquet:. Matching hierarchical classifications with attributes. In *ESWC*, pages 4–18, 2006.
25. J. F. Sowa. *Conceptual Structures: Information Processing in Mind and Machine.* Addison-Wesley, 1984.
26. Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *ICDM*, pages 521–528, 2001.
27. MeSH: the National Library of Medicine's controlled vocabulary thesaurus. see http://www.nlm.nih.gov/mesh/.

28. DMoz: the Open Directory Project. See http://dmoz.org/.
29. Michael Uschold and Michael Gruninger. Ontologies and semantics for seamless connectivity. *SIGMOD Rec.*, 33(4):58–64, 2004.
30. Mark van Assem, Maarten R. Menken, Guus Schreiber, Jan Wielemaker, and Bob Wielinga. A method for converting thesauri to RDF/OWL. In *the Third International Semantic Web Conference (ISWC'04)*, number 3298, pages 17–31, Hiroshima, Japan, November 2004. Lecture Notes in Computer Science.
31. Rudolf Wille. Concept lattices and conceptual knowledge systems. *Computers and Mathematics with Applications*, 23:493–515, 1992.