TOWARDS A THEORY OF FORMAL CLASSIFICATION

Fausto Giunchiglia, Maurizio Marchese and Ilya Zaihrayeu

May 2005

Technical Report # DIT-05-048

Towards a Theory of Formal Classification

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 become Web pages, pictures and any kind of electronic information items. 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 almost impossible 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 logical concept language. The main property of Formal Classifications is that each node can be associated a normal form formula which univocally describes its contents. This in turn allows us to reduce document classification and query answering to fully automatic propositional reasoning.

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" [6]. 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 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 Decimal Classification System (DDC)[2] or the Library of Congress classification system (LCC)[3]) are attempts to develop static, hierarchical classification structures into which all of human knowledge can be classified. Although these are standard and universal techniques; they have a number of limitations:

[1] The WWW Virtual Library project, see http://vlib.org/.
[2] The Dewey Decimal Classification system, see http://www.oclc.org/dewey/.
[3] The Library of Congress Classification system, see http://www.loc.gov/catdir/cpso/lcco/lcco.html/.

- both classification and search tasks do not scale to large amounts of information. This is because, among other things, at any given level in such a hierarchy, there may be more than one choice of topic under which an object might be classified or searched.
- the semantics of a given topic is implicitly codified in a natural language label. These labels must therefore be interpreted and disambiguated.
- the semantic interpretation of a given topic depends also on the meanings associated to the labels at higher levels in the hierarchy [10].

In the present paper we propose a formal approach to classification, capable of capturing the implicit knowledge present in classification hierarchies, and of supporting automated reasoning to help humans in their classification and search tasks. To this end, we propose a two 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 language[4], that we call the Concept Language.
- then we further convert a FC into a *Normalized Formal Classification* (*NFC*). In NFCs each node is associated a Concept Language formula, that we call the *concept at a node*, which univocally codifies the node contents, taking into account both the label of the node and its position within the classification.

NFCs and concepts at nodes have many nice properties. Among them:

- they can be expressed in Conjunctive and/or Disjunctive Normal Forms (CNF / DNF). This allows humans and machines to easily inspect and reason on classifications (both visually and computationally).
- document classification and query answering can be done simply exploiting the univocally defined semantics codified in concepts at nodes. There is no need to inspect the edge structure of the classification.
- concepts of nodes are organized in a taxonomic structure where, from the root down to the leaves of the classification, child nodes are subsumed by their parent nodes.

The remainder of the paper is organized as follows. In Section 2 we introduce and present examples of standard classifications. In Section 3 we introduce the definition of FC and discuss its properties. In Section 4 we introduce the notion of NFC and its properties. In Section 5 we show how the two main operations performed on classifications, namely classification and search, can be fully automated in NFCs as a propositional satisfiability problem. The related and future work conclude the paper.

[4] A Propositional Description Logic language is a Description Logic language [1] without roles.

2 Classifications

Classifications are hierarchical structures used to organize large amounts of objects [10]. 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 objects classification and retrieval. Users browse the hierarchies and quickly catalogue or access the objects associated with different concepts and linked to natural languages labels. We define the notion of Classification as follows:

Definition 1 (Classification) *A Classification is a rooted tree described by a triple $H = \langle C, E, l \rangle$ where C is a finite set of nodes, E is a set of edges on C, and l is a function from C to a set L of labels expressed in a natural language.*

In the rest of this section we describe and briefly discuss two different Classifications: a librarian classification hierarchy Dewey Decimal Classification system (DDC), and an example from a modern web catalogue, namely the Amazon book categories catalogue.

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	$\rightarrow$ 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 in 155.95 as part of the discipline of psychology; customs associated with clothing belong in 391 as part of the discipline of customs; and clothing in the sense of fashion design belongs in 746.92 as part of the discipline of the arts. However, the final Dewey number associated to a book is unique and the classifier needs to impose a classification choice.

As an example, let's see how to determine the Dewey number for the following book: Michael Hanlon, "The Real 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" criterion[5]: when you add a new object, get as specific as possible: dig deep into the classification schema, looking for the appropriate sub-category; it is bad practice to submit an object to a top level category, if one more specific exists. At present, the enforcement of such criterion 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 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.

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

Example 2 (Amazon book directory). Many search engines like Google, Yahoo as well as many eCommerce vendors, like Amazon, offer mechanisms to search for relevant items. This is the case, for instance, of the web directory catalogue for books (among other items) used in Amazon. At present Amazon has 35 main subjects. Books are inserted by the classifier in the web directory, and users browse such classification hierarchy to access the books they are interested in.

In Amazon, as in DDC, books can be classified both in leaf and non-leaf nodes[6], following the "Get Specific" criterion, but also the "Related Directory" criterion[7], when the classifier browses through the hierarchy looking for an appropriate category that lists similar documents. In this classification hierarchy, a book can be often reached from different paths of the hierarchy, thus providing efficient tools to arrive at items of interest using different perspectives.

In the following we present an example of classification for a software programming book in the Amazon Book Web Directory. The book title is "Enterprise Java Beans, Fourth Edition". In the current Amazon book directory[8], the example title can be found through two different search paths (see Figure 2), namely:

```
Subjects → Business and Investing →
Small Business and Entrepreneurship → New Business Enterprises

Subjects → Computers and Internet → Programming → Java Language →
Java Beans
```

From the brief presentation and from the two specific examples we can see that Web catalogues are more open than classifications like Dewey. In fact, their aim is not to try to position a resource in a unique position, but rather to position it in such a way, that a user, who navigates the catalogue, will be facilitated to find appropriate or similar resources related to a given topic.

3 Formal Classifications

Let us use the two examples above to present and discuss a number of characteristics that are relevant to classifications and that need to be considered in a formal theory of classification.

Let us start from the characteristics of edges. People consider classifications top down. Namely, when classifying or searching for a document first upper level nodes are considered, and then, if these nodes are too general for the given criteria, lower level nodes may also be inspected. Child nodes in a classification are

[6] Amazon implements it by assigning to non-leaf nodes a leaf node labeled "General", where items related to the non-leaf nodes are classified

[7] Look at http://www.google.com/dirhelp.html#related to see how Google implements this rule

[8] See http://www.amazon.com, April 2005.

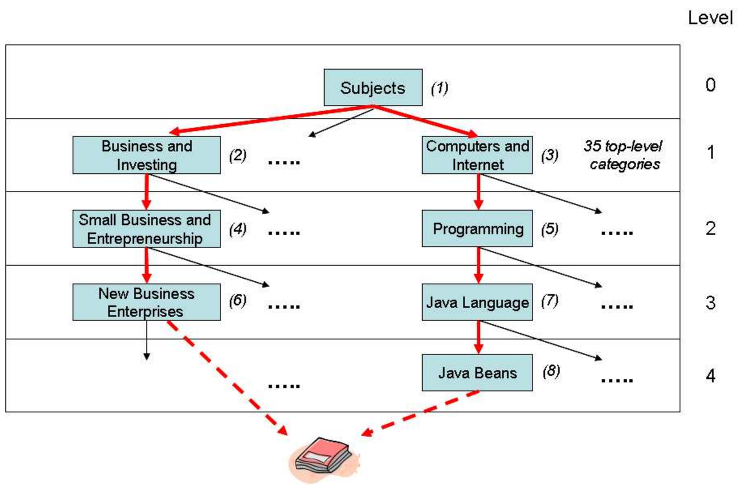

Fig. 2. Amazon Book Directory

always considered in the context of their parent nodes, and therefore *specialize* the meaning of the parent nodes. In a classification there are two possible meaningful interrelationships between parent and child nodes as shown on Figure 3:

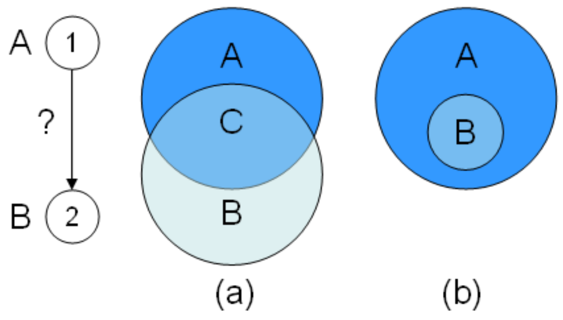

Fig. 3. Edge semantics for formal classifications

– Case (a) represents edges expressing the “general intersection” relation, and, intuitively, the meaning of node 2 is area C, which is the intersection of areas A and B.
  For instance, in our Amazon example, the edge in Figure 2 `Computers and Internet` → `Programming` codifies all the items that are in common (see Figure 4) to the categories `Computers and Internet` (i.e., hardware, software,

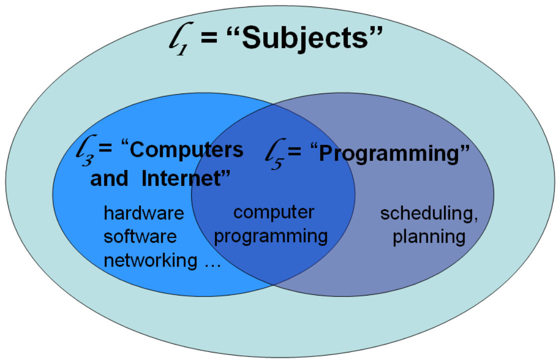

Fig. 4. Example of general intersection

networking, etc) and `Programming` (i.e., scheduling, planning, computer programming, web programming, etc). This kind of edges are also present in library systems, such as DDC, at lower levels of the hierarchy where different facets of a particular parent category are considered.

– Case (b) represents a more specific case where the child node is "subsumed by" the parent node. In this case the meaning of node 2 is area B. This kind of edges is also called an "is-a" edge. Note that in this case, differently from case (a), node A does not influence what is classified in node B.
  Many edges in DDC impose the "is-a" relation, in particular in the higher levels of the hierarchy. Also some edges in the Amazon book directory impose the "is-a" links, the most obvious ones are the edges from the root category.

Notice that, in the case of edges leading to the same resource the "general intersection" relation must hold for *all* the categories in all the different paths. The latter fact can be used to improve the classification representation: either by trying to prohibit this situation (if the goal is to classify unambiguously a resource, as it happens in a library classification, such as DDC) or by enhancing this kind of situation (if the goal is improving the recall of relevant resources, as it happens in a web catalogue, such as Amazon).

Let us now move to consider the characteristics of labels. As from Definition 1, the concept of a specific node is described by a label expressed in words and, possibly, separators between them. The node labels possess interesting structure, relevant to formal classification hierarchies:

– Natural language labels are composed by atomic elements, namely words. These words can be analyzed in order to find all their possible basic forms and eventual multiple senses, i.e., the way in which the word can be interpreted. In this paper, we use WordNet [12] to retrieve word senses[9], however, in

[9] We may change the actual senses of a word from WordNet for the sake of presentation.

practice, a different thesaurus can be used. For example the word "Java" in the label "Java Language" in Figure 2 possesses different equivalent forms (e.g., Java, java) and three different senses:
  1. an island in Indonesia;
  2. a beverage consisting of an infusion of ground coffee beans; and
  3. an object oriented programming language.
- Words are combined to build complex concepts out of the atomic elements. Consider for example the labels `Computers and Internet` and `Java Language` in Figure 2. The combination of natural language atomic elements is used by classifier to aggregate (like in `Computers and Internet`) or disambiguate atomic concepts (like in `Java Language`, where the sense of the word `Java` that denotes "an island in Indonesia" together with the sense "a type of coffee" can be discarded while the correct sense of "an object oriented programming language" is maintained).
- Natural language labels make use of the structure of the classification hierarchy to improve the semantic interpretation associated to a given node. We call this property *parental contextuality* of a node. For instance the sense of words composing labels of different nodes in an hierarchy path can be incompatible; thus the correct meaning of a particular word in a specific label can be disambiguated by considering the senses of the words in some labels along the path. For example, in the path `Java Languages` $\rightarrow$ `Java Bean`, the possible correct (but wrong) sense of `Java Bean` as "a particular type of coffee bean" can be pruned by the classifier taking into account the meaning of the parent node's label, `Java Languages`.

Let us see how we can convert classifications into a new structure, which we call a *Formal Classification* (FC), more amenable to automated processing:

Definition 2 (Formal Classification) *A Formal Classification is a rooted tree described by a triple $H^F = \langle C, E, l^F \rangle$ where C is a finite set of nodes, E is a set of edges on C, and l^F is a function from C to a set L^F of labels expressed in a Propositional Description Logic language L^C.*

As it can be noticed, the key step is that in FCs labels are substituted by labels written in a formal logical language. In the following we will call L^C, the *Concept Language*. We use a Propositional Description Logic language for several reasons. First, we move from an ambiguous language to a formal language with clear semantics. Second, given its set-theoretic interpretation, L^C "maps" naturally to the real world semantics. For instance, the atomic proposition $p =$`computer` denotes "the set of machines capable of performing calculations automatically". Third, natural language labels are usually short expressions or phrases having simple syntactical structure. Thus no sophisticated natural language processing and knowledge representation techniques are required – a phrase can be often converted into a formula in L^C with no or little loss in the meaning. Forth, a formula in L^C can be converted into an equivalent formula in a propositional logic language with boolean semantics. Thus a problem expressed in L^C can therefore be converted into a *propositional satisfiability problem*[10].

Apart from the atomic propositions, the language L^C includes logical operators, such as *conjunction* (denoted by $\sqcap$), *disjunction* (denoted by $\sqcup$), and *negation* ($\neg$); as well as comparison operators: *more general* ($\sqsupseteq$), *more specific* ($\sqsubseteq$), and *equivalence* ($\equiv$). In the following we will also say that A *subsumes* B, if $A \sqsupseteq B$; and we will also say that A is *subsumed* by B, if $A \sqsubseteq B$. The interpretation of the operators is the standard set-theoretic interpretation.

We build FCs out of classifications by translating, using natural language processing techniques, natural language labels, l_i's, into concept language labels, l_i^F's. For lack of space we do not describe here how we perform this step. The interested reader is referred to [10]. As an example, recall the classification example shown on Figure 2. For instance, the label `Java beans` of node n_8 is translated into the following expression:

$$l_8^F = (\texttt{Java}^1 \sqcup \texttt{Java}^2 \sqcup \texttt{Java}^3) \sqcap (\texttt{Bean}^1 \sqcup \texttt{Bean}^2) \tag{1}$$

where $\texttt{Java}^1$ denotes the Java island, $\texttt{Java}^2$ is a brewed coffee, $\texttt{Java}^3$ is the object oriented programming language Java, $\texttt{Bean}^1$ is a kind of seeds, and $\texttt{Bean}^2$ is a Java technology related term. The disjunction $\sqcup$ is used to codify the fact that `Java` and `Bean` may mean different things. The conjunction $\sqcap$ is used to codify that the meaning of `Java beans` must take into account what `Java` means *and* what `Beans` mean.

As it is mentioned above, some senses of a word in a label may be incompatible with the senses of the other words in the label, and, therefore, these senses can be discarded. A way to check this in L^C is to convert a label into *Disjunctive Normal Form* (DNF). A formula in DNF is a disjunction of conjunctions of atomic formulas or negation of atomic formulas, where each block of conjunctions is called a *clause* [11]. Below is the result of conversion of Formula 1 into DNF:

$$\begin{aligned} l_8^F = {} & (\texttt{Bean}^1 \sqcap \texttt{Java}^1) \sqcup (\texttt{Bean}^1 \sqcap \texttt{Java}^2) \sqcup (\texttt{Bean}^1 \sqcap \texttt{Java}^3) \sqcup \\ & (\texttt{Bean}^2 \sqcap \texttt{Java}^1) \sqcup (\texttt{Bean}^2 \sqcap \texttt{Java}^2) \sqcup (\texttt{Bean}^2 \sqcap \texttt{Java}^3) \end{aligned} \tag{2}$$

The first clause in Formula 2 (i.e., $(\texttt{Bean}^1 \sqcap \texttt{Java}^1)$) can be discarded, as there is nothing in common between seeds and the island. The second clause, instead, is meaningful – it denotes the coffee seeds. Analogously, clauses 3, 4 and 5 are discarded and clause 6 is preserved. The final formula for the label of node n_8 therefore becomes:

$$l_8^F = (\texttt{Bean}^1 \sqcap \texttt{Java}^2) \sqcup (\texttt{Bean}^2 \sqcap \texttt{Java}^3) \tag{3}$$

Note, that sense $\texttt{Java}^1$ is pruned away in the final formula as it has nothing to do with any sense of the word "bean". Analogously, all the other labels in

[10] For translation rules from a Propositional Description Logic to a Propositional Logic, see [2, 5].

the classification shown on Figure 2 are translated into expressions in L^C and further simplified. At this point, the "converted" Classification represents a FC.

Note, that each clause in DNF represents a distinct meaning encoded into the label. This fact allows both agents and classifiers to operate on meanings of labels, and not on meanings of single words.

4 Normalized Formal Classifications

As discussed in Section 2, in classifications, child nodes are considered in the context of their parent nodes. We formalize this notion of parental context in a FC following the definition of concept at a node from [5]:

Definition 3 (Concept at a node) *Let H^F be a FC and n_i be a node of H^F. Then, the concept at node n_i, written C_i, is its label l_i^F if n_i is the root of H^F, and, otherwise, it is the conjunction of the label of n_i and the concept at node n_j, which is the parent of n_i. In formulas:*

$$C_i = \begin{cases} l_i^F & \text{if } n_i \text{ is the root of } H^F \\ l_i^F \sqcap C_j & \text{if } n_i \text{ is a non-root node of } H^F\text{, where } n_j \text{ is the parent of } n_i \end{cases}$$

Applying Definition 3 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 H^F to n_i:

$$C_i = l_1^F \sqcap l_2^F \sqcap \ldots \sqcap l_i^F \tag{4}$$

The notion of concept at a node explicitly captures the classification semantics. Namely, the interpretation of the concept at a node is the set of objects that the node and all its ascendants have in common (see Figure 3). From the classification point of view, the concept at a node defines what (class of) documents can be classified in this node.

The definition of concept at a node possesses a number of important properties relevant to classification:

Property C.1: each C_i codifies both the label of n_i and the path from the root to n_i. There are two important consequences of this: first, it allows it to prune away irrelevant senses along the path; and, if converted to DNF, C_i represents the union of all the possible distinct meanings of a node in the FC's tree.

Recall the Amazon running example. According to Formula 4, the concept at node n_8 is:

$C_8 = (\texttt{Subject}^*) \sqcap (\texttt{Computer}^* \sqcup \texttt{Internet}^*) \sqcap (\texttt{Programming}^*) \sqcap (\texttt{Java}^* \sqcap \texttt{Language}^*) \sqcap (\texttt{Java}^* \sqcap \texttt{Bean}^*)$ [11]

The possible correct (but wrong) sense ($\texttt{Bean}^1 \sqcap \texttt{Java}^2$) as "a particular type of coffee bean" (the first clause in Formula 3) can be pruned by converting the concept at node n_8 into DNF, which contains the clause ($\texttt{Language}^1 \sqcap \texttt{Java}^2 \sqcap$

[11] We write $\texttt{X}^*$ to denote the disjunction of all the senses of $\texttt{X}$.

`Bean`^{1}) and checking it as a propositional satisfiability problem: since the meaning of `Language`^{1} is "incompatible" with `Java`^{2} the expression results into an inconsistency.

Property C.2: each C_i has a normal form. In fact it is always possible to transform each C_i in Conjunctive Normal Form (CNF) namely a conjunction of disjunctions of atomic formulas or negation of atomic formulas [11]. Therefore C_i codifies in one logical expression *all* the possible ways of conveying the same concept associated to a node.

We use the notion of the concept of a node to define a further new structure which we call *Normalized Formal Classification* (NFC):

Definition 4 (Normalized Formal Classification) *A Normalized Formal Classification is a rooted tree described by a triple $H^N = \langle C, E, l^N \rangle$ where C is a finite set of nodes, E is a set of edges on C, and l^N is a function from C to a set L^N of concepts at nodes.*

Also the proposed NFC possesses a number of important properties relevant to classification:

Property NFC.1: when all C_i are expressed in CNF (see property C.2), all the nodes expressing *semantically equivalent* concepts will collapse to the same CNF expression. Even when two computed concepts are not equivalent, the comparison of the two CNF expressions will provide enhanced similarity analysis capability to support both classification and query-answering tasks. Following our example, the normalized form of the concept at node n_8 with the path (in natural language):

`Subjects` $\rightarrow$ `Computers and Internet` $\rightarrow$ `Programming` $\rightarrow$ `Java Language` $\rightarrow$ `Java Beans`

will be equivalent, for instance, to the concept associated to a path like:

`Topic` $\rightarrow$ `Computer` $\rightarrow$ `Internet` $\rightarrow$ `Programming` $\rightarrow$ `Languages` $\rightarrow$ `Java` $\rightarrow$ `Java Beans`

and similar (i.e., be more general, or more specific) to (say):

`Discipline` $\rightarrow$ `Computer Science` $\rightarrow$ `Programming languages` $\rightarrow$ `Java` $\rightarrow$ `J2EE` $\rightarrow$ `Java Beans`

Property NFC.2: any NFC is a taxonomy, in the sense that for any non-root node n_i and its concept C_i, the concept C_i is always subsumed by C_j, where n_j is the parent node of n_i. We claim that NFCs are the "correct" translations of classifications into ontological taxonomies as they codify the intended semantics/use of classifications. Notice that, under this assumption, in order to capture

the classification semantics no expressive ontological languages are needed, and a Propositional Description Logic is sufficient. In this respect our work differs substantially from the work described in [10].
Consider in our running Amazon example the path in the natural language classification:

$\texttt{Subject} \rightarrow \texttt{Computers and Internet} \rightarrow \texttt{Programming}$

As described in Section 2, this path contains a link expressing the "general intersection" relation, namely the link is $\texttt{Computers and Internet} \rightarrow \texttt{Programming}$ (see Figure 4). The same relation is maintained when we move to FCs. In our notation: $l_1^F = \texttt{Subject}^*$, $l_3^F = (\texttt{Computer}^* \sqcup \texttt{Internet}^*)$, $l_5^F = \texttt{Programming}^*$. But, when we move to the NFC for the given example, our elements become: $C_1 = l_1^F$; $C_3 = l_1^F \sqcap l_3^F$; $C_5 = l_1^F \sqcap l_3^F \sqcap l_5^F$; and the only relation holding between successive element is the subsumption.

The above properties of both C_i and NFC have interesting implications in classification and query answering, as described in the next Section.

5 Document classification and query answering

We assume that each document d is 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 document's keywords is retrieved using text mining techniques (see, for example, [14]); the keywords are then converted into a corresponding concept using the same techniques used to translate natural language labels into concept language labels (see Section 3).

There exists a number of approaches to how to classify a document. In one such approach a document is classified only in one node (as in DDC), in another approach it may be classified under several nodes (as in Amazon). However, in most cases, the general rule is to classify a document in the node or in the nodes that most specifically describe the document, i.e., to follow the "Get Specific" criterion discussed in Section 2. In our approach, we allow for a document to be classified in more than one node, and we also follow the "Get Specific" criterion. We express these criteria, in a formal way, as follows:

Definition 5 (Classification Set) *Let H^N be a NFC, d be a document, and C^d be the concept of d. Then, the classification set for d in H^N, written Cl^d, is a set of nodes $\{n_i\}$, such that for any node $n_i \in Cl^d$ the following two conditions hold:*

1. *the concept at node n_i is more general than C^d, i.e. $C^d \sqsubseteq C_i$; and*
2. *there is no such node n_j ($j \neq i$), whose concept at node is more specific than C_i and more general than C^d.*

Document d is classified in all the nodes from the set Cl^d in Definition 5.

Suppose we are given two documents: a book on Java programming (d_1) and an article on high tech entrepreneurship (d_2). Suppose now that these documents are assigned the following concepts: $C_1^d = \texttt{Java}^3 \sqcap \texttt{Programming}^2$, and $C_2^d = \texttt{High\_tech}^1 \sqcap \texttt{Venture}^3$, where $\texttt{Java}^3$ is the programming language, $\texttt{Programming}^2$ is computer programming, $\texttt{High\_tech}^1$ is "highly advanced technological development", and $\texttt{Venture}^3$ is "a commercial undertaking that risks a loss but promises a profit". Intuitively, C_1^d is more specific than the concept at the node labeled `Java language` in the classification shown on Figure 2. In fact, logical inference confirms the intuition, namely it is possible to show that the following relation holds: $C_1^d \sqsubseteq C_7$. It is also possible to show that the second condition of Definition 5 holds for node n_7. Thus, document d_1 is classified in node n_7. Analogously, it can be shown that the classification set for d_2 is composed of the single node n_6. For lack of space we do not show the full formulas and the proofs of these statements.

Moving to query answering, when a 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 discussed in Section 3. 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 for q:

$$A^q = \{d | C^d \sqsubseteq C^q\} \tag{5}$$

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_s^q). We compute N_s^q as follows:

$$N_s^q = \{n_i | C_i \sqsubseteq C^q\} \tag{6}$$

In fact, as $C^d \sqsubseteq C_i$ for any document d classified in any node $n_i \in N_s^q$, and $C_i \sqsubseteq C^q$, then $C^d \sqsubseteq C^q$. Thus, all the documents classified in the set of nodes N_s^q belong to the answer A^q (see Formula 5).

We extend N_s^q by adding nodes, which constitute the classification set of a document d, whose concept is $C^d = C^q$. We call this set, the *query classification set*, written Cl^q; and we compute it following Definition 5. In fact, nodes in Cl^q may contain documents satisfying Formula 5, for instance, documents whose concepts are equivalent to C^q.

Suppose, for instance, that a user defines the following query to the Amazon NFC: $C^q = \texttt{Java}^3 \sqcup \texttt{COBOL}^1$, where $\texttt{COBOL}^1$ is "common business-oriented language". It can be shown, that $N_s^q = \{n_7, n_8\}$ (see Figure 2 for the Amazon classification). However, this set does not include node n_5, which contains the book "Java for COBOL Programmers (2nd Edition)". Node n_5 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.

Thus, for any query q, a user can compute a sound query answer A_s^q by taking the union of two sets of documents: the set of documents which are classified in the set of nodes N_s^q (computed as $\{d \in n_i | n_i \in N_s^q\}$); and the set of documents which are classified in the nodes from the set Cl^q and which satisfy Formula 5 (computed as $\{d \in n_i | n_i \in Cl^q, C^d \sqsubseteq C^q\}$). We have therefore:

$$A_s^q = \{d \in n_i | n_i \in N_s^q\} \cup \{d \in n_i | n_i \in Cl^q, C^d \sqsubseteq C^q\} \tag{7}$$

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

Note, that the structure of a NFC (i.e., the edges) is *not* considered neither during document classification nor during query answering. In fact, given the proposed classification algorithm, the edges information becomes redundant, as it is implicitly encoded in the concepts at the nodes. We say *implicitly* because there may be more than one way to "reconstruct" a NFC resulting into the same set of concepts at nodes. But, all the possible NFCs are equivalent, in the sense that the same set of documents is classified into exactly the same set of nodes.

The algorithms presented in this section are sound and complete in the document classification part, as Propositional Logic allows for sound and complete reasoning on documents according to Definition 5. The proposed solution for query answering is sound but not complete as $A_s^q \subseteq A^q$. For lack of space we do not provide evidence of the incompleteness property of the solution.

6 Related Work

In our work we adopt the notion of the concept at node as first introduced in [4] and further elaborated in [5]. Moreover, the notion of label of a node in a FC, semantically corresponds to the notion of the concept of a label introduced in [5]. In [5] 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.

This work as well as the work in [4, 5] mentioned above is crucially related and depends on the work described in [2, 10]. In particular, in [2], the authors, for the first time ever, introduce the idea that in classifications, natural language labels should be translated in logical formulas, while, in [10], the authors provide a detailed account of how to perform this translation process. The work in [4, 5] improves on the work in [2, 10] 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. As a matter of fact the work in [4] classifies the work in [2, 4, 5, 10] as *semantic matching* and distinguishes it from all the previous work, classified under the heading *syntactic matching*. This paper, for the first time, recognizes the crucial role that the ideas introduced in [2, 4, 5, 10] have in the construction of a new theory of classification, and in introducing the key notion of FC.

A lot of work in information theory, and more precisely on formal concept analysis (see for instance [16]) has concentrated on the study of concept hierarchies. NFCs are 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. Remember that concept hierarchies are ontologies which are trees where parent nodes subsume their child nodes.

The 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 DL the underlying structure for classification is not predefined by the user, but is build bottom-up from atomic concepts by computing the subsumption partial ordering. Interested readers are referenced to [7], where the authors propose sound and complete algorithms for realization and retrieval.

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 prepopulated into the categories in order to allow the system to automatically classify a larger set of objects (see, for example, [3, 13]). The latter approach uses various machine learning techniques to classify objects, for instance, data clustering [8]. There exist some approaches that apply (mostly) supervised classification techniques to the problem of documents classification into hierarchies [9, 15]. 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. Notice how in this latter case the user has no or little control over the language used in classifications. Our approach has the potential, in principle, to allow for the automatic classification of the (say) Yahoo! documents into the Yahoo! directories. Some of our current work is aimed at testing the feasibility of our approach with very large sets of documents.

7 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 query answering and document classification. However much more can be done. Our future work includes the development of a sound and complete query answering algorithm; as well as the development and evaluation of tools that implement the theoretical framework presented in this paper. There are two tools of particular importance, namely the document classifier and query answering tools, which will provide the functionality described in Section 5. The performance of the tools will then be compared to the performance of the most advanced heuristics based approaches. Yet another line of research will be the development of a theoretical framework and algorithms allowing for the interoperability between NFCs. The latter particularly includes distributed query answering and multiple document classification under sound and complete semantics.

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. G.Adami, P.Avesani, and D.Sona. Clustering documents in a web directory. *In Proceedings of Workshop on Internet Data management (WIDM-03)*, 2003.
4. F. Giunchiglia and P. Shvaiko. Semantic matching. *"Ontologies and Distributed Systems" workshop, IJCAI*, 2003.
5. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-match: An algorithm and an implementation of semantic matching. *In Proceedings of ESWS'04*, 2004.
6. A.D. Gordon. *Classification.* Monographs on Statistics and Applied Probability. Chapman-Hall/CRC, Second edition, 1999.
7. 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.
8. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.
9. 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.
10. 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.
11. A. Mendelson. *Introduction to Mathematical Logic.* Chapman-Hall, 4th ed. London, 1997.
12. George Miller. *WordNet: An electronic Lexical Database.* MIT Press, 1998.
13. 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.
14. Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002.
15. Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *ICDM*, pages 521–528, 2001.

16. Rudolf Wille. Concept lattices and conceptual knowledge systems. *Computers and Mathematics with Applications*, 23:493–515, 1992.