The Use of Associative Concepts in the Incremental Building of a Logical Context

Sébastien Ferré** and Olivier Ridoux

IRISA, Campus Universitaire de Beaulieu, 35042 RENNES cedex,
{ferre,ridoux}@irisa.fr

Abstract A formal context associates to objects a description that combines automatically extracted properties (*intrinsic*) and manually assigned ones (*extrinsic*). The extrinsic properties are expressed by users according to intentions that are often subjective and changing, and determine the classification and retrieval of objects. So, we find it important to assist users in this task through the automatic suggestion of extrinsic properties to be assigned and even the discovery of rules to automate these assignements. The principle is to learn from the description of existing objects the extrinsic description of a new object. Because of the changing nature of users' intentions, the assistance given in the incremental building of a logical context must be interactive. We present formal principles, and an application to the classification of email messages.

1 Motivation

Whereas much has been said on the construction of concept lattices, the construction of contexts is often left in the background. The construction process falls into two categories: off-line and on-line. In the off-line case, the context is built once for all after the data have been gathered and the problem is to find an object description language appropriate to the intended analysis. In the on-line case, the context is built progressively along the arrival of data and a problem is to properly describe new objects at the time they arrive.

Hypothesis 1 (on-line construction) *We consider in this paper only the on-line case, as we focus on information systems [FR00]. For each new piece of data that arrives, an object is created, and added to the context.*

The description given to an object is two-parts. The first part, the *intrinsic description*, is automatically extracted from the object contents, and depends on the kind of contents and on the logic of the application. For instance, let us consider that objects are incoming emails. In this application, the building of the context is clearly on-line; and possible components of the intrinsic description are the `from`, `to`, and `subject` fields.

** This author is supported by a scholarship from CNRS and Région Bretagne

The second part, the *extrinsic description*, is manually assigned by users according to personal intentions and preferences. We must consider there are no known rules to infer extrinsic properties, as if the contrary holds they could be integrated in the intrinsic description. In a usual email application, extrinsic properties are managed by storing emails in different folders according to personal needs. However, extrinsic properties need not be organized in a hierarchical relation as folders often are.

Users encounter two problems in the manual assignment of properties to new objects: inconsistencies between object descriptions, and tediousness of the task. Inconsistencies result from differences between the intended meaning and the use of an extrinsic property. E.g., an extrinsic property may be forgotten in the description of a new object though it is true for this object. Or the meaning of a property evolves in the user's mind with time, possibly leading to a completely different meaning. In databases, a schema prevents this difficulty by defining a kind of form for the description of objects. The drawback of data-base schemas is that they must be defined a priori, i.e., before any object has been created; it is difficult to change them; and heterogeneous data are not well-handled.

Hypothesis 2 (changing world) *The world outside an application is in constant evolution. We must adapt to incoming objects that create new concepts, while recognizing when a new object belongs to an already existing concept.*

The assignment of an extrinsic description can also become a tedious task. Consider for instance the classification of unsolicited messages (a.k.a. spam) in an email application. There may be many of them, and one would like to find rules for detecting most of them and for automating their classification.

This article presents a method for the incremental building of a logical context, where the existing context plays the role of a dynamic schema, helping users to keep consistency in their descriptions and to find relevant rules to gradually automate the assignment of extrinsic properties. More concretely, this assistance consists in suggesting extrinsic properties to be assigned. This can be compared to the context-based suggestion of query refinements in information retrieval, that we presented in a previous paper [FR01] (see also conclusion).

After introducing useful theoretical elements in Section 2, we show in Section 3 how learning hypotheses (first introduced by Finn [Fin83] and introduced in FCA-terms by Kuznetsov [Kuz99]) can help in characterizing extrinsic properties to be suggested. Then, we develop in Section 4 a more direct, local and efficient way to do it. This leads to the notion of *associative concepts*, which are closely related to modified and new concepts in the incremental concept formation [GMA95]. Experimental results are given in Section 5, and the limits and perspectives of the induction process are discussed in the conclusion.

2 Logical Context, Feature Context, and Sub-Context

This section introduces useful theoretical elements for the rest of the paper.

2.1 Logical Context

We recall the main definitions and results about Logical Concept Analysis (LCA). More explanations and results can be found in [FR00].

Definition 1 (context) *A* logical context *is a triple* $(\mathcal{O}, \mathcal{L}, d)$ *where:*
— $\mathcal{O}$ *is a finite set of objects,*
— $\mathcal{L}$ *denotes a logic whose deduction relation is* $\dot{\models}$*, and whose disjunctive and conjunctive operations are respectively* $\dot{\vee}$ *and* $\dot{\wedge}$*, such that* $\langle \mathcal{L}; \dot{\models} \rangle$ *is a lattice of formulas, whose supremum is* $\dot{\vee}$*, and whose infimum is* $\dot{\wedge}$ [1]*,*
— d *is a mapping from* $\mathcal{O}$ *to* $\mathcal{L}$ *that describes each object by a unique formula.*

Contrary to, say, a boolean logic, $\langle \mathcal{L}; \dot{\models} \rangle$ needs not be a complemented lattice. Given a formal context K, one can form a Galois connection between sets of objects and formulas with two applications σ_K and τ_K.

Definition 2 *Let* $K = (\mathcal{O}, \mathcal{L}, d)$ *be a logical context,*

$$\sigma_K : \mathcal{P}(\mathcal{O}) \to \mathcal{L}, \quad \sigma_K(O) := \dot{\bigvee}_{o \in O} d(o)$$
$$\tau_K : \mathcal{L} \to \mathcal{P}(\mathcal{O}), \quad \tau_K(f) := \{o \in \mathcal{O} | d(o) \dot{\models} f\}$$

Logical concepts can be derived from logical contexts.

Definition 3 (concept) *In a context* $K = (\mathcal{O}, \mathcal{L}, d)$*, a* concept *is a pair* $c = (O, f)$ *where* $O \subseteq \mathcal{O}$*, and* $f \in \mathcal{L}$*, such that* $\sigma_K(O) \dot{\equiv} f$ *and* $\tau_K(f) = O$*. The set of objects* O *is the concept* extent *(*$ext(c)$*), whereas formula* f *is its* intent *(*$int(c)$*). A* non-empty concept *is a concept with a non-empty extent.*

The set of all concepts that can be built in a context K is denoted by $\mathcal{C}(K)$, and it is partially ordered by $\leq^c$ defined as follows.

Definition 4 $(O_1, f_1) \leq^c (O_2, f_2) \iff O_1 \subseteq O_2 \quad (\iff f_1 \dot{\models} f_2)$.
For any formulas $f, g \in \mathcal{L}$*, we say* f implies g in context K *iff* $\tau_K(f) \subseteq \tau_K(g)$.

Definitions 3 and 4 lead to the following *fundamental theorem.*

Theorem 1 *Let* $K = (\mathcal{O}, \mathcal{L}, d)$ *be a context. The ordered set* $\langle \mathcal{C}(K); \leq^c \rangle$ *is a finite lattice. We write* $\vee^c$ *its supremum and* $\wedge^c$ *its infimum.*

In the sequel, an email application is considered to illustrate our principles by examples. Objects are emails, and the chosen logic (L) extends a propositional logic with atoms that are valued attributes, where values allow reasoning on strings (`is "..."`, `contains "..."`, `begins with "..."`, `ends with "..."`), plus the epistemic modalities (`O[...]`, `[...]`) of logic All I Know [Lev90,Fer01] to add Closed Word Assumption on object descriptions. Note in particular that entailment between atoms is non-trivial: e.g., `is "abc"` $\dot{\models}$ `begins with "ab"` $\dot{\models}$ `contains "b"`. We choose to describe emails only with the intrinsic fields `from`, `to`, and `subject` (automatically extracted) and the extrinsic field `kwd` (manually assigned keywords). Other fields (and the body) would certainly be useful in a real application, but they are not necessary for our explanations. Here is an example of an object description,

[1] E.g., if formulas are sets of attributes, the conjunction is the set union, and the disjunction is the set intersection.

```
O[ kwd: private & -kwd: spam
 & from: is "Alice@rennes.fr"
 & to: is "Bob@paris.fr", is "Chloe@paris.fr"
 & subject: is "Hello!" ]
```

which says that *all we know* (`O[...]`) is that `private` is a *keyword* of this message, `spam` is a *non-keyword*, the sender is Alice, the receivers are Bob and Chloe, and the subject is "Hello!". It entails the following query,

```
  - [kwd: public] & [-kwd: spam]
& ([from: begins with "Alice"] | [from: begins with "Bob"])
& - [subject: contains "money"]
```

which says that `public` is not a *keyword*, `spam` is a *non-keyword*, the sender *begins with* either Alice or Bob, and the subject does not *contain the word* `money`. Square brackets delimit an epistemic operator which leads to the distinction of two negations: the *extensional* form `-[f]` ($\tau(\text{-}[\text{f}]) = \mathcal{O} \setminus \tau([\text{f}])$), and the *intentional* form `[-f]` ($\tau([\text{-f}]) \subseteq \mathcal{O} \setminus \tau([\text{f}])$). This distinction can be compared with Wille's between *negation* and *opposition* in his double boolean algebra [Wil00].

2.2 Feature Context

Logical languages contains usually infinitely many formulas, whose complexity is unbounded, which is a problem for algorithms that perform a search among formulas (e.g., learning [GK01], navigation [FR01]). For efficiency and readability of results, we restrict the search space of formulas to a finite subset $F \subseteq \mathcal{L}$ whose elements are called *features*. Features differ from attributes of standard formal contexts in three ways: (1) features belong to a fixed logical language and so, have a semantics, (2) features are automatically ordered according to the deduction $\dot{\models}$, and (3) a newly introduced feature can have a non-empty extent.

It is possible to extract a formal context, with F as the set of attributes, from the logical context: we call it the *feature context*. This context is not intended to be actually build from the logical context, but it is defined to allow reasoning about the logical context with a coarser grain than the full logic.

Definition 5 *Let $K = (\mathcal{O}, \langle \mathcal{L}; \dot{\models} \rangle, d)$ be a logical context, and $F \subseteq \mathcal{L}$ be a finite set of features. The* feature context *of K is the formal context $K_F = (\mathcal{O}, F, I_{K_F})$, where $I_{K_F} = \{(o, x) \in \mathcal{O} \times F \mid d(o) \dot{\models} x\}$. We also define* description features *for any object o by $D_{K_F}(o) = \uparrow_F d(o)$, where for any $f \in \mathcal{L}$, $\uparrow_F f = \{x \in F \mid f \dot{\models} x\}$.*

Lemma 1 relates the Galois connections of logical and feature contexts.

Lemma 1. *Let $O \subseteq \mathcal{O}$, $X \subseteq F$, $\sigma_{K_F}(O) = \uparrow_F \sigma_K(O)$, and $\tau_{K_F}(X) = \tau_K(\dot{\bigwedge} X)$.*

Theorem 2 shows the existence of a mapping that approximates a logical concept in a feature concept and then define equivalence classes among logical concepts.

Theorem 2 *Let $(O, f) \in \mathcal{C}(K)$ be a logical concept. The feature concept generated from O (intent: $\sigma_{K_F}(O)$) and the feature concept generated from f (extent: $\tau_{K_F}(\uparrow_F f)$) are in fact the same feature concept $(\tau_{K_F}(\uparrow_F f), \uparrow_F f)$, the smallest concept in $\mathcal{C}(K_F)$ whose extent is larger than or equal to O.*

Proof: First, we show that the intents of generated concepts are equal to $\uparrow_F f$.
$\sigma_{K_F}(\tau_{K_F}(\uparrow_F f)) = \sigma_{K_F}(\tau_{K_F}(\uparrow_F \sigma_K(O)))$ (because (O, f) is a logical concept)
$= \sigma_{K_F}(\tau_{K_F}(\sigma_{K_F}(O))) = \sigma_{K_F}(O)$ (by Lemma 1)
$= \uparrow_F \sigma_K(O) = \uparrow_F f$.

Second, we show that the generated concept is the smallest larger than O.
Let $(\tau_{K_F}(X), X) \in \mathcal{C}(K_F)$ such that $O \subseteq \tau_{K_F}(X)$,
then $\tau_{K_F}(\sigma_{K_F}(O)) \subseteq \tau_{K_F}(X)$ (properties of Galois connections)
$\Longrightarrow \tau_{K_F}(\uparrow_F f) \subseteq \tau_{K_F}(X)$. (first point of this proof) ■

We assume that the set of features F is automatically extracted from object descriptions of the logical context. As an illustration, we give a set of features automatically extracted from the object description presented in Section 2.1.

```
[kwd: private]                    [to: contains "Bob"]
[-kwd: spam]                      [to: contains "paris"]
[kwd: ?]                          [to: contains "fr"]
[from: is "Alice@rennes.fr"]      [to: ?]
[from: contains "Alice"]          [to: is "Chloe@paris.fr"]
[from: contains "rennes"]         [to: contains "Chloe"]
[from: contains "fr"]             [subject: is "Hello!"]
[from: ?]                         [subject: contains "Hello"]
[to: is "Bob@paris.fr"]           [subject: ?]
```

Note that this set of formulas is not flat w.r.t. the deduction relation $\dot{\models}$. It defines a structured search space among formulas (a *learning bias*).

2.3 Sub-Context

It is often useful to reason on a *sub-context* by restricting the set of objects and the set of features. For instance, the search of positive hypotheses [GK00] is done in the sub-context of positive examples. And the need for *views* has been recognized in object retrieval [CS00,FR01].

Definition 6 *Given a* domain $D \subseteq \mathcal{O}$, *restricting the set of objects, and a* view $V \subseteq F$, *restricting the set of features* F, *we define the* sub-context *of a feature context* K *by the formal context* $K_F(D,V) = (D, V, I_{K_F} \cap (D \times V))$.

Lemma 2 relates the Galois connections of feature contexts and sub-contexts.

Lemma 2. *Let* $O \subseteq D$, $X \subseteq V$,
$\sigma_{K_F(D,V)}(O) = \sigma_{K_F}(O) \cap V$, *and* $\tau_{K_F(D,V)}(X) = \tau_{K_F}(X) \cap D$.

A domain can be specified by the answers $\tau_K(q)$ to a query q. The specification of a view depends on the logic. E.g., in the logic presented in Section 2.1, the formula `([from: ?] | [to: ?]) & -[from: <"paris">]` would select all features talking about sender and receiver addresses, except senders that contain the domain name `paris`. A view V can be used as a *projection* ψ_V [GK01] through the relation $\psi_V(f) = \dot{\bigwedge}\{x \in V \mid f \dot{\models} x\}$, for all $f \in \mathcal{L}$.

3 Induction through Hypotheses

In this section, we use the learning model based on positive and negative hypotheses [Kuz99,GK00] to present an off-line version for the induction of extrinsic properties. This version will be used as a starting point of our on-line version, developed in the next section.

Instead of searching hypotheses in the whole logical language $\mathcal{L}$, we constrain them to be conjunctions of features taken in a set F and belonging to a view V. Furthermore, instead of considering the property to be learnt as an "external" one [GK01], we consider it can be any formula of the logical language: it is called the *goal*, which we denote by $g \in \mathcal{L}$. So, in a context K, positive examples are simply $\tau_K(g)$, negative examples are $\tau_K(\overline{g})$ (where $\overline{g}$ denotes the intentional negation (opposite) of g), and the undetermined examples are $\mathcal{O} \backslash (\tau_K(g) \cup \tau_K(\overline{g}))$. To summarize, what is learnt is parameterized by a goal g and a view V, allowing many different learning applications in a same logical context.

Definition 7 *Let K be a logical context, F be a set of features, and g be a formula. A subset of features $h \subseteq F$ is a* positive (g,V)-hypothesis *iff h is an intent of $K_F(\mathcal{O},V)$, the* support $\tau_K(\dot{\bigwedge} h)$ *is not empty*[2]*, and $\tau_K(\dot{\bigwedge} h) \subseteq \tau_K(g)$ holds. The set of all positive (g,V)-hypotheses of context K_F is denoted by $H_{K_F}(g,V)$.*

A *negative (g,V)-hypothesis* is simply defined as a positive $(\overline{g},V)$-hypothesis. Usually, the view V should exclude features logically entailing the goal (or non-goal) to avoid trivial hypotheses.

We now apply the learning of hypotheses to the classification of unsolicited emails. The goal is the extrinsic property `[kwd: spam]`, the non-goal is `[-kwd: spam]`, and the view selects features logically below the intrinsic properties `[from: ?]`, `[to: ?]`, or `[subject: ?]`, which means we search explanations in the sender and receiver addresses, and in the subject. For the remaining of this section *(g,V)* stands for this goal and this view.

The results of an experiment where the learning context has 67 positive examples and 81 negative examples taken from a real mailbox on a period of one month without any prefiltering are as follows (numbers on the left of hypotheses are sizes of support sets):

- 31 minimal positive *(g,V)*-hypotheses, whose 5 most significant ones are

```
(26)  [to: is "undisclosed-recipients" & from: ?]
(16)  [from: contains "yahoo" & to: ?]
(12)  [subject: contains "adv" & to: ?]
(12)  [from: contains "com" & to: contains "irisa"
      & to: contains "fr" & subject: ?]
(11)  [from: contains "hotmail" & from: contains "com"
      & to: ? & subject: ?]
```

[2] A hypothesis is justified by the fact that it is inhabited by some object. This is why we exclude those with empty support.

– 17 minimal negative *(g,V)*-hypotheses, whose 5 most significant ones are

```
(60)  [from: contains "fr" & subject: ?]
(26)  [to: contains "fr" & subject: contains "re" & from: ?]
(10)  [to: contains "edu" & from: ? & subject: ?]
(9)   [to: contains "ac" & to: contains "uk" & subject: ? & from: ?]
(8)   [from: contains "edu" & subject: ? & to: ?]
```

Now, we use each *(g,V)*-hypothesis as the premise of a rule that classify a new email as a spam or a non-spam. We added 28 new spams and 32 new non-spams in the context (without describing them as spam or not). In the following table, each spam and non-spam is classified as *positive* if its description entails a positive *(g,V)*-hypothesis and no negative one (cautious attitude), as *negative* in the reverse situation, as *none* if it entails neither a positive rule nor a negative one, and finally as *both* if it entails both a positive and a negative rule.

class	positive	negative	none	both
spam	24	0	3	1
non-spam	0	30	1	1

To summarize, 54 out of 60 (90%) emails have been correctly classified (true positive and true negative), 6 out of 60 (10%) are not classified (none and both) and need a manual classification, and no email has been badly classified (false positive and false negative). This shows that, despite our cautious attitude, *(g,V)*-hypotheses have a good generalization capability on the classification of spams (at least in this context), and make no error. Some emails are not automatically classified and need an interaction with users, but this cannot be totally avoided as new kind of spams appear everytime (see Hypothesis 2).

Now we have established the validity of *(g,V)*-hypotheses to support the induction of extrinsic properties of new objects, we want to extend their use in two directions. The first direction is time; we want to update rules any time a new object is not classified or badly classified according to a goal (see Hypothesis 2). The second direction is the set of features; we want to learn and apply hypotheses for every feature instead of choosing one goal g. To realize these two extensions using induction through hypotheses, we should execute the learning algorithm every time for every feature, which is not tractable as the learning algorithm has a high complexity (even when using an incremental algorithm [GK00]).

The following section presents a new approach that focuses on the induction of properties rather than on the extraction of knowledge, which leads to a more interactive and efficient method.

4 Induction through Associative Concepts

Let us consider the situation where a new object o^* is added to a logical context $K = (\mathcal{O}, \mathcal{L}, d)$ along with an intrinsic description $d^*(o^*)$ to form a new context $K^* = (\mathcal{O} \uplus \{o^*\}, \mathcal{L}, d^*)$ with $d^*(o) = d(o)$ for all $o \in \mathcal{O}$. Our aim is to induce from the old context K a set of extrinsic properties $Ind_{K_F}(o^*) \subseteq F$ for the new object.

Definition 8 *We say a feature g is an induced property iff there exists a $(g, D_{K_F}(o^*))$-hypothesis. $Ind_{K_F}(o^*) = \{g \in F \mid H_{K_F}(g, D_{K_F}(o^*)) \neq \emptyset\}$ is the set of all induced properties of an object.*

This means induced properties are features for which a hypothesis can be found among subsets of description features of the new object.

4.1 Associative Concepts

We now give a characterization of $Ind_{K_F}(o^*)$ that uses the notion of "associative concept", instead of *(g,$D_{K_F}(o^*)$)*-hypotheses.

Definition 9 *A non-empty concept of $K_F(\mathcal{O}, D_{K_F}(o^*))$ is called an* associative concept *of o^* in K_F. The set of all such associative concepts is denoted by $AC_{K_F}(o^*)$.*

$AC_{K_F}(o^*)$ organizes the feature context K_F in a concept lattice (where the empty concept is missing) that is less finely detailed than $\mathcal{C}(K_F)$. However, this coarser concept lattice is relevant to the features of o^*. Conversely, the finer details in $\mathcal{C}(K_F)$ cannot be expressed with the features of o^*.

Theorem 3 *$Ind_{K_F}(o^*) = \bigcup_{c \in Min(AC_{K_F}(o^*))} \sigma_{K_F}(ext(c))$, where Min is defined according to the order $\leq^c$ between concepts (see Definition 4).*

Proof: Let $g \in Ind_{K_F}(o^*)$. By Definition 8, this is equivalent to
$g \in F$ and $H_{K_F}(g, D_{K_F}(o^*)) \neq \emptyset$
$\Longleftrightarrow$ $g \in F$ and there exists $c \in \mathcal{C}(K_F(\mathcal{O}, D_{K_F}(o^*)))$
s.t. $ext(c) \neq \emptyset$ and $\tau_K(\dot{\bigwedge} int(c)) \subseteq \tau_K(g)$ (Def. 7)
$\Longleftrightarrow$ $g \in F$ there exists $c \in AC_{K_F}(o^*)$ s.t. $\tau_K(\dot{\bigwedge} int(c)) \subseteq \tau_K(g)$ (Def. 9)
$\Longleftrightarrow$ there exists $c \in AC_{K_F}(o^*)$ s.t. $\tau_{K_F}(int(c)) \subseteq \tau_{K_F}(\{g\})$
(because $g \in F$ and $int(c) \subseteq D_{K_F}(o^*) \subseteq F$)
$\Longleftrightarrow$ there exists $c \in AC_{K_F}(o^*)$ s.t. $g \in \sigma_{K_F}(ext(c))$ (Galois properties)
Moreover, for every $c, c' \in AC_{K_F}(o^*)$,
$c \leq^c c' \Longrightarrow ext(c) \subseteq ext(c') \Longrightarrow \sigma_{K_F}(ext(c)) \supseteq \sigma_{K_F}(ext(c'))$.
Hence, $g \in Ind_{K_F}(o^*)$
$\Longleftrightarrow$ there exists $c \in Min(AC_{K_F}(o^*))$ s.t. $g \in \sigma_{K_F}(ext(c))$
$\Longleftrightarrow$ $g \in \bigcup_{c \in Min(AC_{K_F}(o^*))} \sigma_{K_F}(ext(c))$. ■

Intuitively, an associative concept c of a new object o^* is an already existing concept (for previous objects in K) that has some similarity with the description of o^*. $ext(c)$ is the *support* of the associative concept, whose objects share features of $int(c)$ with the new object. $\sigma_{K_F}(ext(c))$ extends $int(c)$ with induced features, i.e., the features shared by all objects in K_F having $int(c)$ among their description features.

Theorem 3 suggests an algorithm for computing induced features without explicitly extracting all *(g,V)*-hypotheses of every feature $g \in F$, which results in a gain of efficiency. More precisely, if the number of description features $|D_{K_F}(o^*)|$

of the new object is bounded by k, then the search for the extents of associative concepts $AC_{K_F}(o^*)$ is in $O(2^k|\mathcal{O}|)$ (2^k being the worst case for the number of associative concepts; in practice it is much less for large k); and the search for induced features $\sigma_{K_F}(ext(c))$ for each associative concept c is in $O(k|\mathcal{O}|)$. This results in a total complexity, in the worst case, of $O(k2^k|\mathcal{O}|^2)$.

Lemma 3. $\forall x \in \bigcup_{c \in Min(AC_{K_F}(o^*))}(\sigma_{K_F}(ext(c)) \setminus int(c)) : x \notin D_{K_F}(o^*)$.

Thus, features in $\sigma_{K_F}(ext(c)) \setminus int(c)$ do not belong to description features of o^* but can be induced and justified by observed features $int(c)$: we call them *expected features* and we denote them by $Exp_{K_F}(o^*)$.

Lemma 4. $\forall x \in \bigcap_{c \in Min(AC_{K_F}(o^*))}(D_{K_F}(o^*) \setminus int(c)) : \tau_K(x) = \emptyset$.

This property shows that object features of o^* taken into account by no associative concept have an empty extent in K: we call them *new features* and we denote them by $New_{K_F}(o^*)$. There is a simple relation between description, induced, expected, and new features.

Theorem 4 $Exp_{K_F}(o^*) = Ind_{K_F}(o^*) \setminus D_{K_F}(o^*)$,
and $New_{K_F}(o^*) = D_{K_F}(o^*) \setminus Ind_{K_F}(o^*)$.

4.2 Induction

For every associative concept $c \in AC_{K_F}(o^*)$, if the description of the new object o^* is replaced by a formula such that $D_{K_F}(o^*) = \sigma_{K_F}(ext(c))$ then the concept lattice of K^* is isomorphic to the one of K, i.e., every implication (Definition 4) is kept and no implication is added. Considering that the set of implications can only be kept equal or reduced by the insertion of a new object (much like entropy), associative concepts help to maintain relevant implications (for the user) through the incremental building of a logical context.

In fact, it is not desirable in general to add all induced features $Ind_{K_F}(o^*)$ to the new object. Firstly, the set of induced features can be inconsistent. For instance, in the context $(\{1,2\}, L, \{1 \mapsto$ `[a & c]`$, 2 \mapsto$ `[b & -c]`$\})$ with features {`[a]`, `[b]`, `[c]`, `[-c]`}, the induced features of a new object described by the formula `[a & b]` would be `[c]` and `[-c]`, which are opposite and so contradictory. Secondly, implications in a context are empirical and can be contradicted at any moment by a counter-example (see Hypothesis 2). For instance, in the email application, while no spam has been seen the induced property is always `[-kwd: spam]`; this induction is wrong the first time a spam is seen.

This is why an interaction with users is necessary. When an object is created, the system displays expected features along with intents of associative concept as justification for these inductions. If the user agrees with some of these suggestions, he can notify it to the system which displays suggestions again, until he validates the current object description. The user can also gradually automate the process by defining rules to add some features to created objects. The formulation of rule conditions is helped by the justifications given for expected features. Section 5 gives more details on this interactive process.

4.3 Connection with Incremental Concept Formation

In this section, we compare our incremental building of a context to the incremental formation of concepts [GM94,GMA95,VM01]. Both methods are based on the search for specific concepts: associative concepts in our case; old, modified, generator, and new concepts in the other case. Surprisingly, we found a close relationship between associative concepts and both modified and new concepts, which can be redefined as follows.

Definition 10 *The* modified *concepts are those whose intent is included in the description features of* o^**:* $M_{K_F}(o^*) = \{c \in \mathcal{C}(K_F) \mid int(c) \subseteq D_{K_F}(o^*)\}$.

Definition 11 *The* new *concepts are those whose intent is the intersection of the intent of an existing concept and of the features description of* o^**, and does not already exist:* $N_{K_F}(o^*) =$
$\{(\tau_{K_F}(int'), int') \mid \exists c \in \mathcal{C}(K_F) : (int' = int(c) \cap D_{K_F}(o^*)) \notin int(\mathcal{C}(K_F))\}$.

The update of a concept lattice $\mathcal{C}(K_F)$ due to inserting a new object o^* consists in inserting all new concepts, and by inserting o^* in the extension of all modified and new concepts. Therefore, the main task of incremental concept formation is to find all modified and new concepts. The two following theorems characterize these concepts in terms of associative concepts.

Theorem 5 *The following statements are equivalent: (1)* $c \in M_{K_F}(o^*)$ *and* $ext(c) \neq \emptyset$*, and (2)* $c \in AC_{K_F}(o^*)$ *and* $\sigma_{K_F}(ext(c)) = int(c)$.

Theorem 6 *The following statements are equivalent: (1)* $c \in N_{K_F}(o^*)$ *and* $ext(c) \neq \emptyset$*, and (2)* $c \in AC_{K_F}(o^*)$ *and* $\sigma_{K_F}(ext(c)) \neq int(c)$.

To summarize, the non-empty modified and new concepts are exactly the associative concepts. There is an empty modified concept when $D_{K_F}(o^*) \supseteq F$ and it is $(\emptyset, F)$. There is an empty new concept when $\tau_{K_F}(D_{K_F}(o^*)) = \emptyset$ and it is $(\emptyset, D_{K_F}(o^*))$. So, it is sufficient to traverse the concept lattice of $K_F(\mathcal{O}, D_{K_F}(o^*))$ instead of the whole context K_F, because of the definition of associative concepts (Def. 4.1). By the way, we avoid the case where two generator concepts are found but only one new concept is generated. This suggests that the incremental concept formation proposed by Missaoui et al. could be optimized by the use of associative concepts (see perspectives).

5 Experimentation

The aim of this section is to present through experimentations the kind of interactions that help a user to assign extrinsic properties to incoming objects. The principle is as follows.

First, when a new object o^* is created, a description is computed from its content. Also, every rule whose condition is satisfied by this description is applied, whose effect is generally to add or subtract a property.

Second, given this first description, the system displays expected features $Exp_{K_F}(o^*)$ to the user. Expected features are displayed with the intents of associative concepts from which they have been induced, and a support that can be seen as the number of objects supporting the expected feature.

Third, the user can accept or reject each expected features, which causes an update on the description of o^*, and in this case the system displays expected features again. Indeed, the acceptance of a feature can induce other features. Once the user is satisfied by the resulting description, the new object is finally inserted in the context. This interactivity ensures that new concepts can always be learned (see Hypothesis 2).

However, this process can be gradually automated. Indeed, displayed intents of associative concepts are contextual explanations for the expected features. They are premises for suggested rules whose effect is to add automatically the expected feature to new objects. The user is responsible for validating them.

5.1 Filtering spams

We consider here the assisted filtering of spams. The following display shows the initial description of a new email (a spam), with its new and expected features. The context on which the induction of expected features is based is made of the learning and test contexts of Section 3 (200 emails).

```
Current description
       0[ from: is "hh2732774@dtcom.net"
        & to: is "undisclosed-recipients"
        & subject: is "earn money without a job!"]

Expected features
28 [kwd: spam]
    <- [from: contains "net"] & [to: is "undisclosed-recipients"]
    <- [from: ?] & [to: is "undisclosed-recipients"] & [subject: ?]
    <- [from: contains "net"] & [to: ?] & [subject: ?]
    <- [from: ?] & [to: ?] & [subject: contains "earn"]
     & [subject: contains "money"]
2  [to: contains "irisa"]/[to: contains "fr"]/[from: contains "com"]
    <- [subject: contains "earn"] & [subject: contains "money"]
     & [from: ?] & [to: ?]
...
```

We can see that, whereas the `from` and `subject` field are new, several features are induced. This is possible because email fields are split in more or less general words, which enables to find common features between the new object and existing ones: e.g., `[from: contains "net"]`, `[subject: contains "money"]`, `[to: is "undisclosed-recipients"]`.

We also see that the above email is well-recognized as a spam, and this feature is even the most strongly induced one with several explanations and a weight of 28 supporting objects. These explanations suggest some rules such

as "every email sent to `undisclosed-recipients` is a spam", or "every email whose subject contains words `earn` and `money` is a spam".

Figure 1 shows the filtering rate of spams during the incremental building of an email context. The dashed line represents the rate of well-classified emails (as spam or non-spam) at the n-th insertion. The solid line represents the rate of classified emails (well or not). So, the part between the two lines represents badly classified emails (e.g., spam as non-spam), and the part above the solid line represents non-classified emails. This plot shows that after a transient phase of about 50 emails, the rate of well classified emails steadily reaches 85%, and there are nearly no bad classification.

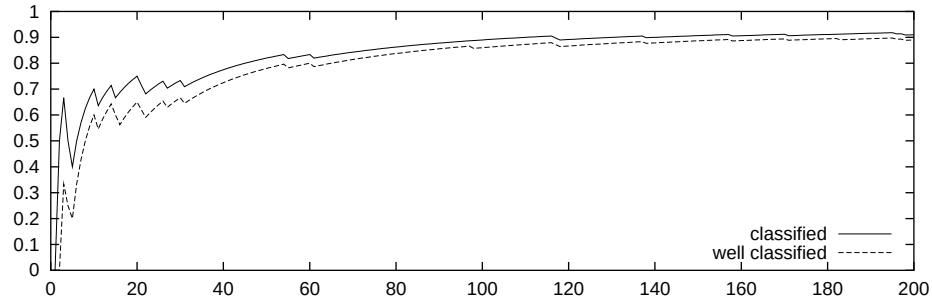

Figure1. Filtering spams

The rates in Figure 1 are accumulated from the start of the experiment: $classified(n) = \frac{total\ classified(n)}{n}$. So they take into account the bad rates of the transient phase. The pseudo-instantaneous rates, $classified(n) = \frac{total\ classified(n) - total\ classified(n-\delta t)}{\delta t}$ are over 90% after the 80^{th} email for a time window (δt) of 50 emails (average number of emails per week).

5.2 Classifying emails

The second application is a variant of the first one in which keywords are not limited to two values. We classify emails in about 20 non-exclusive categories such as teaching, research, spam, call-for-paper, and so on. Note that these categories were not fixed a priori, but appeared only when required by the meaning of incoming messages. Thus, the vocabulary of categories remains open for ever.

Figure 2 shows the results of this experiment. The "automatic" line shows the rate of automatic classification using rules that are suggested by the system and accepted by the user (40 rules, including 15 for spams). The "suggested" line shows the rate of correct suggested classification. It tends to decrease simply because the sum of the two rates must be less than 1. Both rates are measured in number of features. The solid line shows the sum of the two rates.

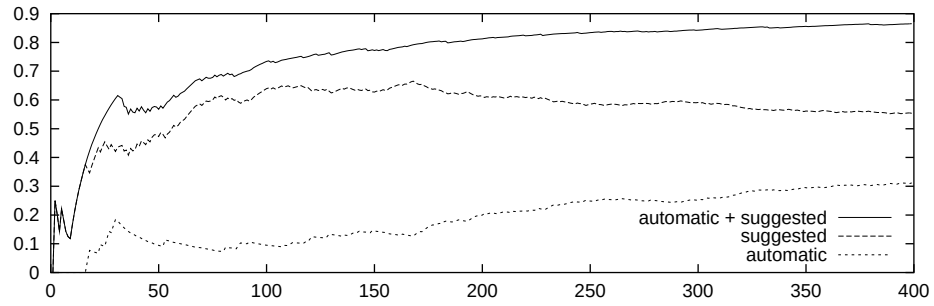

Figure2. Classifying emails

Note again that these rates are cumulative; the instantaneous rates of "automatic+suggested" is constantly over 85% after the 100^{th} email for a time window of 50 emails.

6 Conclusion and perspectives

We have proposed a learning schema by which an incoming object is given extrinsic properties according to its intrinsic ones. This is done by using the intent of the concepts to which the new object seems to belong. However, to facilitate the exploration of the intents, only a scaled version of the property language is used; we call it a language of features. This introduces a *learning bias* as important information may be missing in the selected features: e.g., in the experimentations presented in this article, we do not consider the body of emails. Moreover, as the feature set is finite and the logical language is infinite, there may be two concepts that cannot be distinguished by only using features. A perspective is to generate feature sets of increasing precision when needed during the learning process, for distinguishing two objects when they must be classified differently. Thus, the learning bias of features would be eliminated, and only the logic bias would remain.

An important application of this learning schema (it was our motivation) is to help in building complex formal contexts. We have proposed a protocol by which, when presented a new object, a system extracts intrinsic properties from it and suggests extrinsic ones. The user can (in)validate them or propose his preferred extrinsic properties. So doing, the classifications represented in the formal context are kept consistent. This method contrasts with using a fixed schema as a form that the user must fill in. We believe it is preferable for evolving context in which the schema must change also, and for heterogeneous contexts.

Another application is to use the learning schema for navigating. A description would play the role of a query, and its associative concepts could be proposed to the user as alternative queries. The advantage is that though the initial query could have an empty answer, the alternative ones always correspond to non-

empty concepts. So, it makes it possible to start a search with only an example of what the user is looking for.

Another interesting perspective is to develop further the correspondence with new concepts and modified concepts that we have presented in Section 4.3. It may lead to improved algorithms for computing concepts incrementally.

References

[CS00] R. Cole and G. Stumme. CEM - a conceptual email manager. In G. Mineau and B. Ganter, editors, *Int. Conf. Conceptual Structures*, LNCS 1867, pages 438–452. Springer, 2000.

[Fer01] S. Ferré. Complete and incomplete knowledge in logical information systems. In S. Benferhat and P. Besnard, editors, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, LNCS 2143, pages 782–791. Springer, 2001.

[Fin83] V. K. Finn. On machine-oriented formalization of plausible reasoning in the style of F. Backon–J.S. Mill. *Semiotika Informatika*, 20:35–101, 1983. In Russian.

[FR00] S. Ferré and O. Ridoux. A logical generalization of formal concept analysis. In G. Mineau and B. Ganter, editors, *Int. Conf. Conceptual Structures*, LNCS 1867, pages 371–384. Springer, 2000.

[FR01] S. Ferré and O. Ridoux. Searching for objects and properties with logical concept analysis. In H. S. Delugach and G. Stumme, editors, *Int. Conf. Conceptual Structures*, LNCS 2120, pages 187–201. Springer, 2001.

[GK00] B. Ganter and S. Kuznetsov. Formalizing hypotheses with concepts. In G. Mineau and B. Ganter, editors, *Int. Conf. Conceptual Structures*, LNCS 1867, pages 342–356. Springer, 2000.

[GK01] B. Ganter and S. Kuznetsov. Pattern structures and their projections. In H. S. Delugach and G. Stumme, editors, *Int. Conf. Conceptual Structures*, LNCS 2120, pages 129–142. Springer, 2001.

[GM94] R. Godin and R. Missaoui. An incremental concept formation approach for learning from databases. *TCS*, 133(2):387–419, 1994.

[GMA95] R. Godin, R. Missaoui, and H. Alaoui. Incremental concept formation algorithms based on Galois (concept) lattices. *Computational Intelligence*, 11(2):246–267, 1995.

[Kuz99] S. Kuznetsov. Learning of simple conceptual graphs from positive and negative examples. In J. M. Żytkow and J. Rauch, editors, *Principles of Data Mining and Knowledge Discovery*, LNAI 1704, pages 384–391. Springer, 1999.

[Lev90] H. Levesque. All I know: a study in autoepistemic logic. *Artificial Intelligence*, 42(2), March 1990.

[VM01] P. Valtchev and R. Missaoui. Building concept (Galois) lattices from parts: Generalizing the incremental methods. In H. S. Delugach and G. Stumme, editors, *Int. Conf. Conceptual Structures*, LNCS 2120, pages 290–303. Springer, 2001.

[Wil00] R. Wille. Boolean concept logic. In G. Mineau and B. Ganter, editors, *Int. Conf. Conceptual Structures*, LNCS 1867, pages 317–331. Springer, 2000.