Semi-Automatic Transformation of a Paper Dictionary into an Electronic Lexical Knowledge Base *

Alessandro Castaldelli
casta@artu.usr.dsi.unimi.it

Gaetano Aurelio Lanzarone
lanzarone@hermes.mc.dsi.unimi.it

Lothar Lemnitzer
lothar@zappa.uni-muenster.de

Wolf Paprotté
wolfp@zappa.uni-muenster.de

Abstract

We present an approach to the problem of extracting lexical-semantic and encyclopaedic information from paper dictionaries. The innovative aspects are not to be found in the parsing techniques used, but in the way parsing techniques are combined to semi-automatically create Lexical Knowledge Bases (LKBs) for a multimodal context of use. The proposed method takes as input a normal dictionary in paper, textual and loosely form, and semi-automatically transforms it into an electronic, hypertextual, well-structured LKB. We describe an implemented prototype which combines the phases of parsing, translation, correction, and enrichment with new information. It transforms a paper dictionary into an electronic one, restructures the dictionary, ignores the ill-formed parts, and transforms the linear sequence of the dictionary entries into an hypertext. The prototype has been applied to (part of) a french dictionary; the resulting electronic version is available on the World Wide Web, where it can not only be navigated through the usual hypertextual links, but can also be queried at will.

1 Introduction

The use of natural languages in information systems has become increasingly widespread during the last years. Two of the major environments

*The project whose results are described in this paper has been conducted by the Universities of Milan and Muenster as part of an ERASMUS program on Techniques in Artificial Intelligence.

where they are used are in the implementation of natural language interfaces and in automatic translation systems. In both cases, natural language is used in input by the user (to insert questions, information or phrases to be translated) and in output by the system (to supply answers). The construction of natural language processing (NLP) systems, which can interpret and generate phrases, thus becomes necessary. A large part of natural language expressions contains ambiguities that cannot be resolved through syntactic analysis only. Fillmore (1968; Lyons (1977) point out that syntax is insufficient for understanding natural languages, and that a profound structure to represent the semantic relations of the lexicon is needed. The use of a Lexical Knowledge Base (LKB) is thus fundamental, because it can resolve many problems in interpreting and generating natural language phrases. The results of the work we are going to present may be of interest in so far as we successfully dealt with one of the most intransigent problems of NLP: the acquisition of lexical knowledge. A pervasive practical problem of NLP is the lack of large LKB with full lexical descriptions. Lexica are the bottlenecks of language technology in such domains as machine translation, full text information retrieval (Lancaster, 1986; Krovetz, 1989), natural language interfaces, grammar checking, and even seemingly trivial applications such as spell checkers. The problem becomes even greater when semantic descriptions of lexical entries are needed, for example for systems capable of message understanding, automatic abstracting or inferencing. Whereas large lexical modules do not exist, due to their being costly and time consuming to develop, copyrighted printed dictionaries are widely available. The notion of reusability of such printed materials has therefore been propagated.

To understand the wider theoretical implications of this work, it should be borne in mind that the linguistic sign, the *word*, is a unit of form and meaning. The meanings that we express via linguistic signs are aspects of the knowledge we have of the world. From a theoretical point of view, then, the analysis of dictionary data is also interesting for the semantic and encyclopaedic information they contain both implicitly and explicitly. By parsing a dictionary one can expect to extract *world knowledge* and to represent and model it for Artificial Intelligence applications (Boguraev and Levin, 1993; Sanfilippo, 1993; Pustejovsky, 1991; Salton, 1989). Computational linguistics has turned to a particular method of lexical acquisition, based on the restructuring of existing resources, in a semiautomatic way, into large LKB (Fasolo, Pazienza, and Velardi, 1991; Pazienza and Velardi, 1989; Calzolari, 1988; Byrd et al., 1987). Two main strategies are adopted (Grefenstette, 1994; Aitchison and Gilchrist, 1987): (i) the use of *corpora*, by extracting lexical information with statistical methods from newspaper articles and other texts and (ii) the use of *machine-readable dictionaries* (MRD), in digital form, from which to automatically extract lexical knowledge.

In this work, only the second method has been considered, as being typical of NLP, while the first is more frequently used to build Information Retrieval systems. More specifically, we propose extraction and representation strategies of lexical knowledge, and apply them to the *Dictionnaire des Idées par les Mots* (DIM) (Delas and Delas-Demon, 1985).

2 Overview of the Proposed Approach

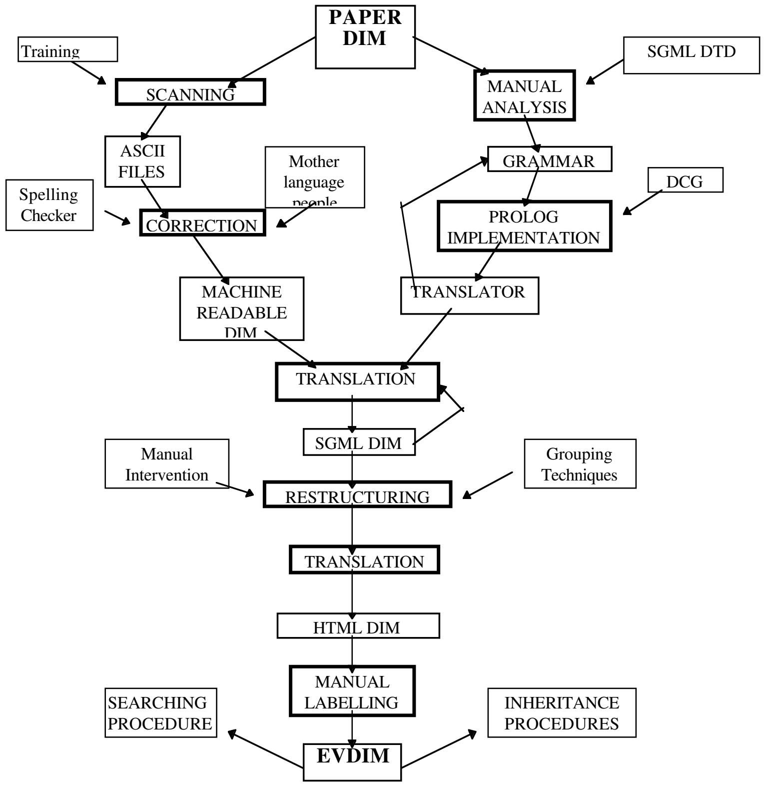

Figure 1: Converting a paper DIM into an EVDIM

Figure 1 shows the basic steps involved in the process, emphasizing how it is possible to pass from a paper copy of the DIM to an electronic version of it (EVDIM). The digitalizing operations of the DIM (scanning and correction, both automatic and manual) are not treated in this paper, which instead details the other steps. The first operation is a translation from a machine-readable DIM into an SGML format. This is followed by a series of restructuring steps, and then again by a translation (from SGML into HTML), transforming the DIM into a hypertext. A labelling operation then converts the 'disordered' hypertextual structure into a semantic network. We maintain that the proposed strategy, applied here to the DIM, can easily be generalized and adapted to other types of dictionaries.

2.1 Analysis of the DIM Structure

The authors of the DIM define it to be an 'Analogical Dictionary' i.e. a dictionary characterized by similarity connections (analogies) among the various lemmas. From a different point of view, the DIM resembles a Thesaurus, where the lemmas are grouped thematically. In order to better understand this double nature of the DIM, we have analyzed its structure in detail.

2.1.1 The Macro-structure

By *macro-structure*, we mean the dictionary in its entirety with its main components. Usually, a dictionary is made up of a first section (preface, introduction, descriptive tables, etc.), the body of the dictionary (list of entries and their respective definitions) and a concluding section (notes, bibliography, etc.). The DIM does not contain such a concluding section.

The first section contains a preface (*Avant-propos* and *Bibliographie*), describing the historical precedents of the DIM and giving a justification of the realization method used by the authors. The first section also contains the classification taxonomy of the knowledge contained in the DIM, consisting of three tables representing three different levels of detail.

The first table (*distribution des thèmes*) contains nine general categories, and indicates the subcategories that each of them includes. The second table (*regroupement thématique des mots-centres*) gives a greater level of detail, indicating the set of central words (*mots-centres*) contained in each subcategory. The third table (*liste alphabétique des mots-centres avec références aux catégories du tableau II*) lists all the central words (1029) in alphabetic order, giving the subcategories to which they belong. The set of the three tables can be exemplified as in Figure 2.

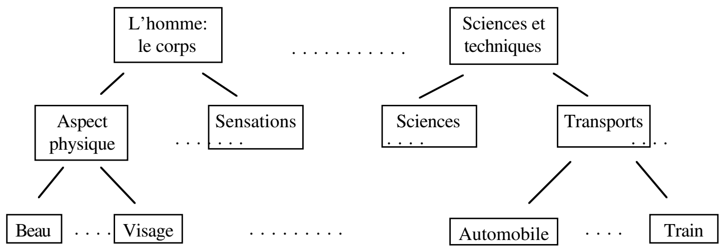

Figure 2: Classification taxonomy of the DIM

2.1.2 The Micro-structure

By *micro-structure* we mean the organization of the individual lexical entries, which appear in alphabetical order in the body of the dictionary, and can be either simple or complex. A simple entry consists only of one lemma (and its possible morphological variant) that points, using an arrow, to other lemmas (which are always central words). We found that the pointed lemmas are always central words, contained in Table 3.

A complex, or structured, entry (example follows) is made up of two parts: the simple part (similar to a simple entry) and a text (the structured part). The text contains functional information about using the lexical entries, and is divided into paragraphs (introduced by a dash ('–') and by a title in bold). Each paragraph is divided into subparagraphs (introduced by □), each of which is a list of words separated by punctuation marks (comma, colon, slash ('/'), semicolon) with different meanings: the semicolon separates expressions belonging to *different ideas* within the same subparagraph; the colon introduces a list; the slash is used to compact such expressions as "AB, AD" into "AB/D"; and the comma separates expressions that are considered semantically correlated in any list, that is, those which belong to the *same idea*.

EXAMPLE 1 (complex entry)

> ROUE ↦ aider, bicyclette, gonfler gymnastique.
>
> – **Éléments et accessoires d'une roue.** Bandage pneumatique, pneu; barbotin; boudin ; chapeau de roue, enjoliveur; couronne; esse, essieu; frette; fusée; jante, déjanter; moyeu; oeil; rai, rayon; voile. □ Alluchon, axe, came, cliquet, dent, galet, godet, lanterne, mentonnet, molette, pale, pignon, pivot, rochet, toutillon, touret, volant. □ Engrenage, rouage, roulement, toulette, train de roulement, transmission. – **Sortes de roues.** □ Automobile; roues amovibles/chenillées/couplées/ directrices / indépendantes / jumelées/ motrices / pleines / à rayons / de secours; écarement, empattement, ouverture, pincement. □ Horlogerie ; roue d'armage/d'artêt/à colonne/à rochet. □ Mécanique; roue d'angle/à augets/ de champ/dentée/de friction/hydraulique / hyperbolique / libre / maîtresse/ à palettes/à sabots. □ Turbine, turboalternateur, turbocompresseur, turbopropulseur, turboréacteur. – **Poulie.** Bras ou toile, canal, engoujure, essieu, gorge, jante, joue, noix, rainure, réa; bigue, bouc, moufle, palanquin ; roue é tagée/fixe/folle; roue à croc/à émerillon/ à fouet; rouet. – **Utilisation des roues.** Automobile, avion, bateau à aubes, bicyclette, brouette, moulin à eau/à vent, turbine, véhicule, wagon. □ Carrosser, carrossage; centrer; charron; embattre, embattage; encliqueter, encliquetage; engrener, engrenage; s'enrayer, enrayement; fretter; voiler, dévoiler. □ Roue de supplice, rouer en place de Grève; roue de la Fortune.

2.2 Descriptive grammar of the DIM

As can be seen from the preceding analysis, the knowledge contained in the DIM is represented using syntactic rules, which are here formalized through a context-free grammar. The grammar of the DIM is to be defined by specifying a Document Type Definition (DTD) for the SGML format, which is a consolidated standard for text labelling. Each word is labelled, indicating its role in the context where it is found. A short example of our DTD follows, while the DTD of the entire DIM is given in the Appendix.

EXAMPLE 2 (macro structure DTD)

```
<!ELEMENT book -- (front_matter, dictionary_body)>
<!ELEMENT front_matter -- front_page, preface, bibliography,
user_guide, tables)>
<!ELEMENT front_page -- (#PCDATA)>
<!ELEMENT preface -- (#PCDATA)>
<!ELEMENT bibliography -- (#PCDATA)>
<!ELEMENT user_guide -- (#PCDATA)>
<!ELEMENT tables -- (title, table 1, table 2, table 3)>
<!ELEMENT title -- (#PCDATA/CAPITAL LETTERS BOLD)>
```

To better understand the DTD, note that the expression `#PCDATA` indicates any sequence of terminal characters, while the slash '/' is used to indicate a specific type of character. For example, `#PCDATA/CAPITAL LETTER BOLD` indicates a string of capital letters in bold face.

As is easy to see, the DTD is a context-free grammar that describes the syntactic structure of the dictionary, but it also describes how the text is to be labelled. From this follows one of the major advantages of the SGML format: the DTD of a document permits its automatic reading and use in other applications, with no need for further layout information. The DIM in SGML format can be useful in many applications, e.g. editing the DIM for errors and inconsistencies in view of a reprint or electronic reissue, building lexical knowledge bases or building relational data bases.

2.3 Translator

It has previously been mentioned that the DIM was digitalized through scanning the paper version into a machine- readable dictionary. The translator that will be described here converts the machine-readable (ASCII) DIM into the SGML format.

The reasons for the DIM requiring a translator (and thus a parser) are twofold: The consistency of the proposed structure must be verified through the grammar. Since we examined only some parts of the DIM in our micro/macro structure analysis, it might not be adequate for the entire dictionary. The best way to assure the correctness of our interpretation is to use the resulting grammar in a parser which can test the entire DIM. Some discrepancies identified by the parser are described below. These differences between the proposed grammar and the dictionary data can be interpreted as data errors, but if they occur frequently, the interpretation of the structure must be revised. Obviously, a parser can only identify syntactic inconsistencies; other types of inconsistencies exist (such as semantic ones) and will be treated later.

The typographical conventions of the dictionary must be converted into SGML format, in order to preserve all the information. We have seen that some typographical information can be lost during the scanning process (such as the distinction between different font types). The SGML format guarantees that such information is saved, by separating each word from the rest with labels. These labels also contain all the necessary information to find the function that the words had in context. As mentioned above, the translator has two goals: consistency checking and SGML conversion, operations accomplished simultaneously, in a single pass, while the translator both parses and translates. The translator has been implemented with the Prolog programming language, which, thanks to the power of logic programming, allows the programmer to work at an abstraction level that is very close to the human way of thinking. The data produced by the scanning process has no information about the fonts used in the dictionary to express the different functions of the words. The grammatical analysis must therefore limit itself to the information given by the punctuation marks, which must be recovered in the case of manual correction. Two types of alphabets can be distinguished for this purpose: *delimiter alphabet* and *word alphabet*. The first contains symbols such as: *arrow* '$\mapsto$', *dash* '–', *puce carrée* '□', *comma* ',', *period* '.', etc., while the second contains all the characters commonly used to shape words. The period, however, is included in both alphabets, since it is not only a punctuation mark but can also appear as part of an abbreviated word (for example: *etc.*, *fam.*, *pop.*). This problem has been resolved by adding a list of all possible abbreviations found in the text, which permits the recognition of a string that ends with a period as a word if it appears in that list. Otherwise, a period is considered to be a normal punctuation mark. The translator is also based on another distinction among the characters, upper- and lower-case, which is saved during scanning and is used to distinguish words not separated by punctuation marks. In fact, some words are separated only by blanks or by end-of-line, but since these are also used within single expressions, the translator cannot use them only. The only way to separate these words is to consider any string of characters starting with a capital letter as a new word.

The translator has been implemented using the Definite Clause Grammars (DCG), which facilitate parser development. Once a (well-designed) grammar describing the document syntax is available, it remains only to transform it into the DCG notation. The DTD of SGML, as previously shown, can be interpreted as a grammar describing the formalism used in the DIM. The parser has thus been easily implemented by exploiting the similarity between the two formalisms.

The parser developed is of the top-down, depth-first type, thus following the Prolog strategy. The most evident advantage of this choice w.r.t. a bottom-up parser is the ease of implementation, as well as the possibility of corrections and changes. The interaction between grammar and data that permitted the realization of the Parser was based on these characteristics. Its efficiency problems have been resolved by keeping to a minimum the text portions submitted each time to the parser.

The translator was developed according to the following iterative process: starting from the conceptual analysis of a small part of the dictionary, the initial set of rules was developed. Then, these rules were used to implement a first translator prototype, which was then applied to a larger part of the dictionary. For each error found in the lemmas whose structure appeared correct, the initial grammar rules were corrected. This procedure was repeated until all the parts recognised as correct in the dictionary had passed the syntactical analysis of the translator.

Note that the notion of correctness/incorrectness of the dictionary parts is relatively subjective. In the beginning, as no manual was available with these kinds of indications, we based our work on suppositions that had to be verified and revised. In the output of the translator, all symbols are removed that do not belong to either the *delimiter alphabet* or the *words alphabet.* The words that have thus been cleaned of 'non-layout' characters are separated by the labels defined in the DTD. This is shown in the following Example, were, in order avoid problems due to different ASCII codes, we have made the following substitutions in the input files: '`->`'for '$\mapsto$' (*arrow*), '*' for □ (*puce carrée*) and '' for '–' (*dash*).

EXAMPLE 3 (simple_basic_entry)

3.a Translator Input (Lemma AUTOMATION):
AUTOMATION, AUTOMATIQUE `->` entreprise, habitude, industrie, machine, mecanique.

3.b Translator output:

```
<e_g><s_b_e><l_g><l>AUTOMATION</l>
                 <l>AUTOMATIQUE</l>
            </l_g>
            <c_g><c>entreprise</c>
                 <c>habitude</c>
                 <c>industrie</c>
                 <c>machine</c>
            <c>m ecanique</c>
            </c_g>
     </s_b_e>
</e_g>
```

3 Restructuring the Dictionary into SGML

The analytic study of the DIM has revealed various problems, mainly due to print and compilation errors of the DIM itself. For example, lemmas without pointers (articles that do not point to other lemmas) imply an error, since they lack the characteristic syntactic/semantic connections among the lemmas. Also lemmas that point to simple lemmas signal an error, since normally only complex lemmas (corresponding to *mots-centres*) are pointed to.

The following initial cleaning procedures can be automatically applied to the DIM in SGML format:

1. All the central words of the tables are verified to be structured lemmas and vice versa. As already mentioned, this double checking was performed manually on more than one hundred lemmas, with no contradiction being found. To automatically check this characteristic, each central word is taken in sequential order from Table 3 and verified to be a structured lemma. At the same time, all the DIM lemmas not found in Table 3 are verified to be simple lemmas. This check fails when either the central word is not a structured lemma, or the structured lemma is not a central word in the initial tables.

2. Each article is verified automatically by the translator to contain pointers to other lemmas, since the lack of pointers (*category-group*) is considered to be a syntactic error. If a lemma without pointers is found, a lemma for it to point to has to be chosen. A method for verifying the appropriateness of the newly introduced connections is proposed below.

3. A procedure has been implemented to verify that pointers reference structured lemmas. Two alternatives exist in this case as well: eliminate the pointer to the simple lemma or transform the simple lemma into a complex one.

4. Each lemma that points to another one is verified to appear at least once in the pointed text. This means that if LEMMA1 points to LEMMA2, then LEMMA1 must appear at least once in the LEMMA2 text. One of two alternatives must be chosen if this does not occur: add the lemma in the text or remove the pointer.

These procedures allow a completely automatic consistence and correctness verification, but require the manual intervention of an expert in case of incongruencies. A natural approach is to perform the operations sequentially, passing to the following phase only once all the errors of the preceding one have been corrected, until all the procedures have been applied.

3.1 Restructuring the Taxonomic Hierarchy

We have seen up to now how to initially clean the DIM of errors and inconsistencies. This was followed by more strictly semantic procedures, to suggest a better structuring of the DIM, or in any case, to check that their structure is consistent.

Let's examine some semantic problems that arose during the DIM analysis. The criteria for choosing the lemmas to be grouped into a subcategory are not always clear and convincing. Since each subcategory can be seen as a *semantic field*, the lemmas it contains should be strictly correlated, or should at least be closely related to the subcategory name. This was found not to be always true. An example is the TRANSPORTS category, which contains the lemmas GONFLER and ANCRE at the same level as AUTOMOBILE and TRAIN. Most probably, the common use of lemmas has had a greater influence on their assignment to a subcategory than their conceptual relationship. For example, an association of MOURIR with RIRE comes from the expression *c'est à mourir de rire*. The organization of the initial DIM tables taxonomy leads to another problem, being quite poor in its three levels and sometimes questionable. For example, the SCIENCES ET TECHNIQUES category contains the subcategories SCIENCES, TECHNIQUES, AIR,EAU,FEU, COULEUR ET LUMIÈRE, MÉTAUX and TRANSPORTS. While it is clear that sciences and techniques belong to this category, it is puzzling that AIR,EAU,FEU, COULEUR ET LUMIÈRE, etc. do. They can be seen as objects of SCIENCES or materials, tools of TECHNIQUES. We now show a way to transform each subcategory into a network of complex lemmas, emphasizing some problems that are inherent to the assignment of lemmas to the same subcategories We have chosen as a significant example the subcategory G2 ANIMAUX within the category G. LA TERRE, LES PLANTES ET LES ANIMAUX, since it is one of the largest in the DIM with its 48 lemmas. The following example lists its lemmas in the left-hand column, with its pointers ('-->') to other lemmas of the same subcategory (upper-case) or of other subcategories (lower-case). Asterisks indicate lemmas belonging to more than one subcategory.

EXAMPLE 4 (Subcategory ANIMAUX)

```
AILE --> architecture, armée, automobile, INSECTE, OISEAU
ALOUETTE --> OISEAU
ANE --> BÉTAIL, CHEVAL, ignorer, sot
ANGUILLE --> POISSON
ANIMAL --> #### no pointers ####
ARAIGNÉE --> ANIMAL, péche
BALEINE --> ANIMAL, pluie
BATRACIEN --> ANIMAL
BEC --> faim, OISEAU, parler, pont
BERGER --> CHIEN, garder, MOUTON
BÉTAIL --> ANIMAL, BERGER, élevage, ferme
BEURRE --> BOEUF, gras, LAIT
BOEUF --> ANIMAL, BÉTAIL, course, VIANDE
CANARD --> journal, OISEAU
CERF --> blason, CHASSE
CHASSE* --> ANIMAL, CHIEN, sport, VIANDE
```

```
CHAT --> ANIMAL, gorge, jouer, MAMMIFÈRES
CHEVAL --> cavalerie, chevalerie, course, HARNAIS
CHÈVRE --> ANIMAL, CERF, CUIR, MAMMIFÈRES
CHIEN --> ANIMAL, avare, difficile, météorologie
CORNE --> cavalerie, CERF, INSECTE, pli, respiration
CRI* --> ANIMAL, appeler, applaudir, bruit
CRUSTACÉS --> ANIMAL
CUIR* --> ANIMAL, chaussure, cheveu, peau
HARNAIS --> CHEVAL
INSECTE --> ANIMAL
LAIT --> BEURRE, blanc, BOEUF
LAPIN --> CRI, élevage, POIL, RONGER
LOUP --> ANIMAL, faute, féte, MAMMIFÈRES, POISSON
MAMMIFÈRES --> ANIMAL, SINGE
MOLLUSQUES --> ANIMAL
MOUCHE --> fusil, INSECTE, visage
MOUTON --> cloche, doux, prison
OISEAU --> AILE, BEC, doute, maéonnerie
PAPILLON --> inscription, ISECTE, neger
PARASITE --> trouble
PLUME --> CANARD, écrire, littérature, OISEAU
POIL --> cheveu, CUIR, force, parasse, peau, tendance
POISSON --> ANIMAL, élevage, lac, mer, pécher
POLYPE --> tumeur
PORC --> ANIMAL, CRI, grossier
QUEUE* --> boule, groupe, prendre
RAT --> avare, danse, livre, RONGER
REPTILES --> ANIMAL, soumettre
RONGER* --> dent, détruire, douleur, manger
SINGE --> adroit, ANIMAL
VER --> ANIMAL, soie VIANDE* --> aliment, chair
```

Let us call: R (*Relatives*) the lemmas belonging to the same subcategory, and S (*strangers*) the lemmas belonging to other subcategories. A lemma of a subcategory can point only to R, like MAMMIFÈRES, or only to S, like MOUTON, or to both R and S, like LOUP.

The following statistics can be found: N indicates the number of lemmas

Lemmas belonging only to ANIMAUX		Lemmas belonging to more that one subcategory	
$N = 42$	$P = 87{,}5\%$	$N = 6$	$P = 12{,}5\%$
$N_r = 9$	$P_r = 21{,}9\%$	$N_r = 0$	$P_r = 0{,}0\%$
$N_s = 3$	$P_s = 7{,}3\%$	$N_s = 3$	$P_s = 50{,}0\%$
$N_{rs} = 29$	$P_{rs} = 70{,}7\%$	$N_{rs} = 3$	$P_{rs} = 50{,}0\%$

Table 1: statistics for lemmas

belonging to that subcategory; P represents the respective frequency; N_r indicates the number of lemmas pointing only to relatives; N_s indicates the number of lemmas pointing only to strangers; N_{rs} indicates the number of lemmas with pointers both to strangers and to relatives.

Note that the lemma ANIMAL has no pointers and was not counted, being considered an anomaly of the dictionary.

Table 1 shows that (i) if the lemma belongs to more subcategories, it is improbable that it points only to *Relatives* ($P_r = 0$) and that (ii) if the lemma belongs only to the subcategory being examined, it is improbable that it points only to *Strangers* ($P_s = 0.073$).

A general and intuitive rule can be derived from these observations (and from observations on other subcategories), we call it the *connection principle*: each lemma belonging to a subcategory should be connected to the lemmas in that subcategory, pointing to at least one of them. This means that a lemma should not point only to *Strangers* if it belongs only to the subcategory being examined, while it cannot point only to *Relatives* if it belongs to more than one subcategory (this would imply that none of the other subcategories to which it belongs could contain pointers to *Relatives*).

An exception to this principle could be the subcategory-lemma, which represents the subcategory itself. Note that ANIMAL has no pointers, due to an error, and that the possible connections could be, as we have seen, to the *Stranger* lemmas MANGER and RESPIRER. For all of the other lemmas in a subcategory, verifying the connection principle means verifying that those lemmas belong to the semantic field (subcategory) in which they are found. It would be absurd to admit a lemma to a semantic field if it did not point to some other lemma of the field, that is, if the connection principle were not satisfied. If such a thing were to occur, that lemma would be considered a marginal detail common to many lemmas in the semantic field, and it would necessarily be linked to the subcategory- lemma, thus satisfying again the connection principle.

Due to the above points, we considered it necessary to check if there exists a lemma not satisfying the connection principle among the lemmas within a subcategory. In order to better understand how to handle this situation, we have built a *network* (see Figure 1 in appendix) of complex lemmas belonging to the same subcategory, according to the following rules:

1. the network nodes consist of all, and only, the *Relative* lemmas of the subcategory (*Stranger* lemmas are excluded as having little significance for the semantic field covered by the subcategory)

2. the network arcs are the connections among the lemmas of the subcategory, and are directed from the pointing lemma to the pointed lemma (double arrows indicate reciprocal pointers between the lemmas).

A network of this type shows:

- the number of arcs in input to a given node, to be interpreted as the pertinence and relevance of the corresponding notion

- the number of arcs in output from a given node, to be interpreted as the multiplicity of aspects of the corresponding notion
- the reciprocity of arcs between two nodes, to be interpreted as a close relationship between the corresponding notions
- the weakly connected nodes (with only one arc in input or output) and the unconnected nodes, to be interpreted as their marginality.

The network shows that the lemmas PARASITE, POLYPE, and MOUTON do not satisfy the connection principle. The first two are totally unconnected, without input or output arcs, while MOUTON has only an arc in input. The QUEUE node is completely unconnected, but this lemma belongs to more than one subcategory and thus does not contradict the connection principle. We can count the input and output arcs, in order to find out the importance of the nodes and how central they are to the subcategory. Assuming that the importance of a notion is proportional to the number of times it is cited, we have assigned a higher value to input arcs of the lemmas than to the output ones. In this way, we can put the lemmas in order of importance, as in the example below:

EXAMPLE 5 (Centrality ordered list (ANIMAUX))
1 node with 21 input, 0 output (ANIMAL)
1 node with 5 input, 2 output (OISEAU)
1 node with 4 input, 1 output (INSECTE)
1 node with 3 input, 2 output (MAMMIFÈRES)
1 node with 2 input, 3 output (BOEUF)
1 node with 2 input, 2 output (BÉTAIL)
6 nodes with 2 input, 1 output
2 nodes with 2 input, 0 output
1 node with 1 input, 3 output (CHASSE)
4 nodes with 1 input, 2 output
5 nodes with 1 input, 1 output
1 node with 1 input, 0 output (MOUTON)
1 node with 0 input, 4 output (CHÈVRE)
2 nodes with 3 output
5 nodes with 0 input, 2 output
12 nodes with 0 input, 1 output
3 nodes (*isolated*) with 0 input, 0 output (PARASITE, POLYPE, QUEUE)

Some important considerations emerge from this ordered list, and from observation of the network of lemmas, viz. (i) the presence of unconnected or weakly connected nodes (lemmas), each of which requires individual treatment, (ii) the possibility of updating the network using *Stranger* type nodes, and (iii) the issue concerning centrality (generality) of the lemmas that are at the top of the *ordered list*.

The presence of *unconnected nodes* merits a certain attention, and can be due to various reasons, e.g. having forgotten some fundamental links among the lemmas in the subcategory during the compilation of the dictionary, or it may be due to the peripheral nature of the weakly connected nodes, interpreted as having a marginal weight (marginal significance) in the subcategory context. This case includes the lemmas like PARASITE and QUEUE. The problem of whether these lemmas should belong to the subcategory at all must be resolved. The same situation exists for the weakly connected nodes, with only one input or output arc, such as BALEINE, BATRACIEN, MOLLUSQUES, etc. (0 input, 1 output), MOUTON (1 input, 0 output).

We believe that the choice of eliminating a lemma or not cannot be based solely on the connection principle. This principle, or more generally a weak connection, must be an (automatic) indicator of possible problems tied to certain lemmas, but it remains to the human expert to choose to eliminate a lemma or to add new connections for it, on the basis of semantic considerations which include many factors, such as the context in which the word is used. An examination of the structured part (*functional information*) of the lemma in question can provide a help in this situation, as it contains information about the use of the lemma.

The *Stranger* nodes can be used to introduce new, if weak, connections in the Network, in the following way: if two *Relative* nodes point to the same *Stranger* node, it may make sense to insert a direct connection between the two *Relative* nodes. The significance of the new connections has to be accurately verified, for example by using a *Stranger* lemma: the lemma ARMÉE justifies a new connection MARINE-AVIATION, since ARMÉE points both to MARINE and to AVIATION

It is fundamental to verify the sense of new connections, since two *Relative* nodes could point to the same *Stranger* node. The following criterion, that we call consistency principle, can be used in the verification: given two lemmas, the arc that connects LEMMA1 to LEMMA2 is consistent only if LEMMA1 appears within the text (structured-part) of LEMMA2. This principle, which seems to be the basis of the true connections among the DIM lemmas, will be used further on.

The *top lemma* in the ordered centrality list (ANIMAL) seems to have a primary function. In fact, it has the most input arcs, indicating that the concept it expresses is important to all of the subcategory nodes. This lemma could thus have a *centroid* function for the ANIMAUX subcategory. The intention is to assume that the *centroid* represents the subcategory, and to thus raise the central word to the role of subcategory.
Building the complex lemma network involves the following steps:

1. determine all the subcategory lemmas and their pointers

2. distinguish the *Relatives* form the *Strangers*, and use only the *Relatives* to build the network, leaving the *Strangers* for later consideration

3. find the isolated lemmas within the network, which neither point to *Relatives* nor are pointed by them, and study in detail the *semantic* reasons for their unconnectedness

4. find those nodes that do not respect the connection principle (that are those of the subcategory that do not point to any *Relatives*), and add connections to other lemmas in the subcategory if possible (if not, eliminate them)

5. examine the overall structure of the network and reorder it with uniform criteria, possibly by using *grouping algorithms* for the manual intervention. This last point will be described below.

3.2 Assimilation Algorithms

By *assimilation algorithms* we mean grouping algorithms that, finding properties shared by different lemmas, allow their placement within the same class. Some suggestions are given about how to apply these algorithms to the networks of complex lemmas previously described.

We have found that the same word can be used in the DIM for different functions. For example, DROIT, GÉOGRAPHIE and ART are all used as both subcategory names and lemma names. In other cases, morphological variants of the same word are used to emphasize the different functions: TRANSPORT, SCIENCE and MÉTAL are lemma names while TRANSPORTS, SCIENCES and MÉTAUX are subcategory names. In this second case, the plural form of the word indicates the subcategory, and the singular the particular lemma. This lack of uniformity in naming, besides creating confusion, also leads to problems later, when we wish to define the relation between the lemma and its subcategory. What relation exists between TRANSPORT and TRANSPORTS, SCIENCE and SCIENCES,or MÉTAL and MÉTAUX?

Our intention is to find a way to make the use of names more uniform,through a grouping algorithm that extracts the *centroid* lemma in each subcategory,to represent the entire subcategory (subcategory-lemma), thus eliminating the other name that is used to indicate the subcategory. In the subcategory ANIMAUX, for example, the term ANIMAUX could be substituted by ANIMAL,the same one that is used for the lemma. As we have seen, a network could be built for each subcategory. Large and small structured networks result for many subcategories, such as for transport and animals, while they would be better adapted to a hierarchical structure, because of their semantic content.

One of the problems found while trying to use inheritance procedures on the DIM structures in their original form, is the flat hierarchy that does not allow the use of many inferences. The hierarchical structure of the subcategory networks could be improved by using the grouping algorithm on them, allowing hierarchical inference algorithms.

The proposed grouping algorithm is based on counting the input and output arcs of each node of the network, referring to the ordered centrality list of the lemmas previously described.
The assimilation algorithm works as follows:

1. the top lemma in the ordered list is defined the *centroid* of the subcategory

2. defining a base number L of input arcs, all the lemmas that have more that L input arcs are considered *subcentroids*

3. the *subcentroids* are connected to the *centroids*

4. the remaining lemmas are connected to the various *subcentroids*, on the basis of semantic criteria

The *centroid*, which is very easy to find, performs a general or central function for all of the lemmas of the subcategory. The *subcentroids* are lemmas that perform a central function for the subgroup of the subcategory, and are found sequentially in the ordered list, as in the example: OISEAU (5 inputs), INSECTE (4 inputs), MAMMIFÈRES (3 inputs). These lemmas are assumed to be *subcentroids*, to which all the remaining lemmas are to be connected.

The problem remains of defining the base level (number of input arcs) above which the lemma is considered a subcentroid. A possible solution is to calculate the average number of input arcs for each node. over the total of nodes that have at least one input. If we use this value (2.56) in the example, the subcentroids are the same ones assumed previously in an arbitrary manner: OISEAU, INSECTE and MAMMIFÈRES.

Once we have found the *centroids* and the *subcentroids*, the connections in the network must be reordered through direct intervention. An example of how the network should result is shown in Figure 3, where the centroid lemma = ANIMAL, subcentroid lemma 1 = OISEAU, subcentroid lemma 2 = INSECTE and subcentroid lemma 3 = MAMMIFÈRES:

The way to move towards a network like the one in the Figure 3, where the *centroid* is pointed only by the *subcentroids* and all the other lemmas are divided among the various *subcentroids*, could be the following:

- directly connect all subcentroids (MAMMIFÈRES, INSECTE and OISEAU) to the centroid (ANIMAUX); assumption of the new connection OISEAU–ANIMAL (confirmed by the consistency principle) is also acceptable.

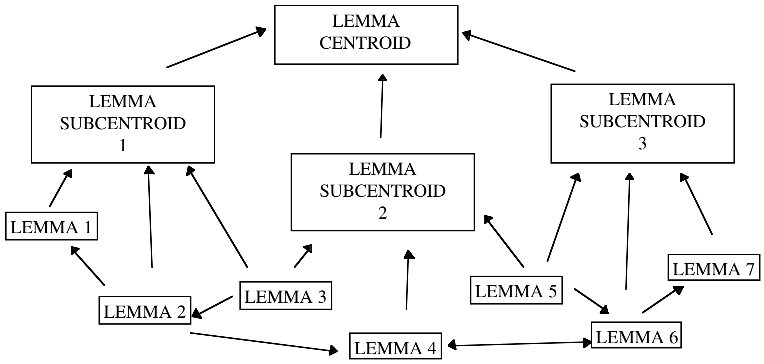

Figure 3: Example network ANIMAL

- if a lemma is connected to a subcentroid, eliminate any of its connections to the centroid, e.g. CHÈVRE–ANIMAL makes CHÈVRE–MAMMIFÈRES unnecessary, and the same goes for CHAT–ANIMAL.

These are only some of the automatic interventions possible, but to arrive at a truly hierarchical network some manual intervention is always necessary.

4 SGML–HTML Translator

After having restructured the DIM into SGML, we proceeded to find an alternative mode of representation of its content. As we have seen, the DIM is divided into two parts: the initial part, with the taxonomic classification of the structured lemmas and the body of the dictionary, where each lemma is described by pointers and (only for the structured lemmas) by functional information

It was decided to use a hierarchical network to represent the initial part, since this would maintain the taxonomic classifications of the lemmas, due to the way it is built. For the body of the dictionary, there is the problem of associating names with the various connections. The problem is not in finding meaningful names for each relation, but in finding names that can be used uniformly throughout the DIM. In fact, the inference associated with each of these selections must be defined. This difficulty was resolved by using hypertexts, which allow non-formalized connections without having to label the relations.

The structure of the EVDIM can therefore be imagined as a large hypertext, with each lemma being a node and each connection between the lemmas being a link. This could be considered as a semantic network for those parts that allow the labelling of the relations and, in particular, a taxonomic network when the relations are hierarchical (IS-A type). We have used hypertextual links to maintain non-formalizable and labelable relations such as not to lose any information present in the DIM. In fact, non-formalizable connections have much linguistic importance.

The HTML (HyperText Mark-up Language) format, commonly applied to realize hypertexts distributed on the WWW (World Wide Web), was used to implement the prototype. Two types of nodes make up our hypertextual network: *lemmas*, which are the leaf elements of the network, comprised of the simple and complex lemmas of the DIM, and *classes*, which are the internal elements of the network and can be categories or subcategories of the DIM.

The network arcs are the links between the lemmas, categories and subcategories. Other arcs can be added, when new links are automatically added, as described later. The structure (representation) of the nodes is shown in the tables below.[1] A class has the type structure shown in Table 2, whereas the lemma type structure is depicted in Table 3. The *up-links* are labels used as hierarchical links toward nodes that are higher up in the hierarchy. The *relative-links* indicate links to peer-level lemmas. The *attributes* indicate attributes that can be added to a node, and the *aux-links* indicate auxiliary links, such as that to the first page of the DIM, or those to particular procedures (e.g. searching, inheritance inference).

identificator	`<CLASS#>`	Identification label of the node
title	`<TITLE>...</TITLE>` `<H1>...</H1>`	Title
up-links	`<UPL#>...</UPL#>*`	Can be void
attributes	`<A#>...</A#>*`	Can be void
down-link	`<DWL#>...</DWL\#>`	Elements belonging to the class
aux-link	`<AL#>...</AL#>?`	Auxiliary link, can be void

Table 2: Class type structure

The EVDIM has been realized by using a Prolog Translator to transform the DIM from our DTD into HTML. The SGML–HTML translator was not difficult to implement, since the HTML is practically just another DTD of the SGML. The translator also builds up the HTML network, with each node being stored on a file, and the name of the file being used as a pointer from all of the other nodes that point to it. The following additional operations are executed during the translation phase: (i) labelling of the relations, in uniform *link-to* mode, and (ii) adding new links between the individual lexical entries within the structured lemmas and the other complex lemmas.

The first operation names the relations among the various lemmas, but

[1]We use the symbols '*', '+' and '?' to denote 'zero or more', 'one or more' and 'zero or one', respectively.

lacking any precise rules about how to do this, we decided to use common names for the same types of relations: lemma *link-to* lemma, lemma *element-of* class, etc.

The second intervention adds new links between the individual words and corresponding lemmas (if they exist), within the structured part of complex lemmas. The procedure was to verify if each word in the text corresponded to a lemma with the same name, and to use as links only those words that satisfy this condition. This has the great advantage of allowing navigation within the DIM structure at a lower level than that given by the DIM itself.

The prototype was then enriched with a search procedure that permits a very fast search for any lemma. We observed that the automatic labelling performed by the translator was not very significant from the semantic point of view. That is, no interesting results would be obtained by applying inference to the structure thus produced, thus manual labelling by an expert in the sector is again mandatory. Only this makes possible the selection of appropriate names for each relation, while frequently maintaining a simple hypertextual link. An example of manual labelling is given in Figure 2 of Appendix. To enrich the EVDIM, it would be useful to intervene manually to add descriptive attributes for the various lemmas and categories, which could then be inherited following the hierarchical inference rules.

5 Uses of the hypertextual structure

Some useful applications of an electronic thesaurus are examined below, along with some inference mechanisms that could be applied to a lexical knowledge base. In particular, we have implemented multiple inheritance procedures in Prolog, and tested them on the taxonomic structure of the EVDIM and on some typical AI examples.

A first advantage of the electronic DIM, though not strictly connected to natural language processing, is its ease of consultation. We have mentioned

identificator	`<LEMMA#>`	Identification label of the node
title	`<TITLE>...</TITLE>` `<H1>...</H1>`	Title
up-link	`<UPL#>...</UPL#>*`	Can be void
relative links	`<RL#>...</RL#>+`	Links to other lemmas
attributes	`<A#>...</A#>*`	Can be void
functional-inf	`<FI#>...</FI#>?`	For complex lemmas it cannot be void
aux-links	`<AL#>...</AL#>?`	Auxiliary link, can be void

Table 3: Lemma type structure

that an analogic dictionary is based on similarity relations among the various lemmas. What distinguishes it from normal dictionaries is the way that it is consulted; not limited to searching in alphabetic order for the various lemmas, but by jumping from one to the other based on semantic links. As can easily be imagined, our hypertextual structure makes this much easier, by using the hypertextual links to navigate among the various thesaurus entries.

An electronic thesaurus is also essential in the applications such as interpreting and generating natural language phrases (Mann, Paris, and Swartout, 1991), eliminating lexical ambiguities (Sowa, 1984) and understanding the meaning of lemmas.

Finally, the EVDIM is a lexical knowledge base; the labelling process has transformed a *simple* hypertextual structure into a semantic and hierarchical network. Thus, we have constructed a LKB consisting of a data base with all the basic facts and with all the information relevant to the domain in question. A traditional data base must possess an inference engine in order to be called a knowledge base and to be used within intelligent systems. Based on the distinction between semantic and hierarchical networks, we have found that the initial part of the DIM can be reorganized using a hierarchical network, while the rest of the DIM can become a more general semantic network, and in some cases, a hypertextual structure. This different organization is due to difficulties encountered in the formalization of the relations and links present among the various nodes (lemmas, categories, subcategories).

While inheritance inference procedures can be applied on a hierarchical network, a more general semantic network requires the definition of specific inferential procedures for each type of relation We have not examined all the possible inferential rules necessary for a semantic network (some of which are proposed in (Thomason and Touretzky, 1991)), and have limited our work to some multiple inheritance procedures applicable to hierarchical networks, using IS-A relations. Inheritance by default is one of the most appropriate inferential mechanisms for a hierarchical structure, and thus we have realized a multiple inheritance with default program. Based on the minimal inferential distance principle (Touretzky, 1984) and on class priorities (Ballim, Russell, and Warwick-Armstrong, 1993), this has led to resolving many inconsistency and ambiguity problems that are typical of inheritance by default. The algorithm was tested on the DIM structure and on some typical AI examples (Touretzky, 1984; Knight and Rich, 1991).

6 Concluding remarks

We feel that the current work has allowed us to reach some important results:

- consistency verification and correction of the dictionary was made easier, with the objective of reorganizing it for a new version, while digitalization and analysis of the structure are the first steps necessary for future reprinting

- an analogical dictionary prototype in electronic form was produced *semi-automatically*

- a lexical knowledge base, useful for natural language processing, was also produced *semi-automatically.*

Many future developments remain after reaching these initial objectives. Besides the enrichment of the EVDIM with syntactical and morphological information, missing in the DIM, the implemented tools and the studied methods could be applied to other types of dictionaries similar to analogical ones, like those for synonyms and for antonyms. We think, however, that the next step is the application of these methodologies to more complex structures like encyclopaedias, which also contain references between entries and thus have an organizational model that is very close to that of DIM. In addition, thanks to the enormous knowledge of the world available, they would become an important reason for the automatic construction of LKB.

The implemented prototype is available on the World Wide Web, where (part of) the dictionary can not only be navigated through the usual hypertextual links, but can also be queried at will.

References

Aitchison, J. and A. Gilchrist. 1987. *Thesaurus Construction. A Practical Manual.* Aslib.

Ballim, A., G. Russell, and J. C. Warwick-Armstrong. 1993. Practical approach to multiple default inheritance for unification-based lexicons. In T. Briscoe, V. de Paiva, and A. Copestake, editors, *Inheritance, Default, and the Lexicon.* Cambridge University Press.

Boguraev, B. and B. Levin. 1993. Models for lexical knowledge bases. In J. Pustejovsky, editor, *Semantics and the Lexicon.* Kluwer Academic Publishers.

Byrd, R. J., N. Calzolari, M. S. Chodorow, J. L. Klavans, M. S. Neff, and O. A. Rizk. 1987. Tools and methods for computational lexicology. *Computational Linguistics*, 13(3).

Calzolari, N. 1988. The dictionary and the thesaurus can be combined. In M. Walton Evens, editor, *Relational Models of Lexicon. Representing Knowledge in Semantic Networks.* Cambridge University Press.

Delas, D. and D. Delas-Demon. 1985. *Dictionnaire des Idées par les Mots.* Robert.

Fasolo, M., M. T. Pazienza, and P. Velardi. 1991. How to encode semantic knowledge: A method for meaning representation and computer-aided acquisition. *Computational Linguistics*, 17(2).

Fillmore, C. J. 1968. The case for case. In E. Bach and R.T. Harms, editors, *Universals in linguistic theory.* Holt Rinehart Winston.

Grefenstette, G. 1994. *Explorations in Automatic Thesaurus Discovery.* Kluwer Academic Publishers.

Knight, K. and E. Rich. 1991. *Artificial Intelligence.* McGraw Hill, second edition.

Krovetz, R. 1989. Lexical acquisition and information retrieval. In *Proceedings of the 1st International Lexical Acquisition Workshop*, Detroit, Michigan.

Lancaster, F. W. 1986. *Vocabulary Control for Information Retrieval.* Information Resources Press.

Lyons, J. 1977. *Semantics.* Cambridge University Press.

Mann, W. C., C. L. Paris, and W. R. Swartout. 1991. *Natural Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer Academic Publishers.

Pazienza, M. T. and P. Velardi. 1989. Computer aided interpretation of lexical cooccurrences. In *Proceedings of the 1st International Lexical Acquisition Workshop*, Detroit, Michigan.

Pustejovsky, J. 1991. The generative lexicon. *Computational Linguistics*, 17(4).

Salton, G. 1989. *Automatic Text Processing. The Transformation, Analysis, and Retrieval of Information by Computer.* Addison Wesley.

Sanfilippo, A. 1993. Lkb encoding of lexical knowledge. In T. Briscoe, V. de Paiva, and A. Copestake, editors, *Inheritance, Default, and the Lexicon.* Cambridge University Press.

Sowa, J. F. 1984. *Conceptual Structures. Information Processing in Mind and Machine.* Addison Wesley.

Thomason, R. H. and D. S. Touretzky. 1991. Inheritance theory and networks with roles. In J. F. Sowa, editor, *Principles of Semantic Networks. Explorations in the Representation of Knowledge.* Morgan Kaufmann.

Touretzky, D. S. 1984. Implicit ordering of default in inheritance system. In *Proceedings of 5th National Conference on Artificial Intelligence*, Austin, Texas.