A UNIFIED MANAGEMENT AND PROCESSING OF WORD-FORMS, IDIOMS AND ANALYTICAL COMPOUNDS

Dan Tufis
Octav Popescu
Research Institute for Informatics
Miciurin 8-10, 71316, Bucharest, 1
Fax:653095
Romania

ABSTRACT

The paper presents a morpho-lexical environment, designed for the management of root-oriented natural language dictionaries. It also encapsulates the basic morpho-lexical processings: analysis and synthesis of individual word-forms or compounds (idioms and analytic constructions).

INTRODUCTION

Lately, a proliferation of **computational lexicon environments** (CLE) has been noticed, which significantly influence the work on natural language (mainly, machine translation) (Byrd et al. 1987), (Nirenburg and Raskin 1987), (Ritchie et al. 1987) etc. With more and more computing power incorporated, the modern CLEs are capable to process not only individual inflected words or derivatives but also idioms and collocations. Nonetheless, there are many applications in language industry which consider a CLE an unfordable luxury. We believe that such an objection may be refused if the CLE is so designed that it should function in a data-driven manner.

We have purposely developed a morpho-lexical management and processing environment aimed at providing an unified and satisfactory solution to a wide range of applications: intelligent text-processing, textual information retrieval, natural language interfacing, natural language understanding, machine translation. Also, and more important, the environment is intended to be used for a large class of natural languages (at least for those of which morphology may be described in terms of our **paradigmatic model** (Tufis 1989)).

In order to reach these objectives, we made a clear distinction between the **morphological processings** and the **knowledge** governing them. This distinction is beneficial not only with respect to natural language independence from the processing environment but also with respect to the desired degree of complexity of the process in case. The lack of information in such an approach will not block the system but will produce a simplified result.

An interesting characteristic of our system is its capability to treat, besides idioms, analytical compounds as well as grammatical and lexical collocations.

The work reported here is developed within the context of the **paradigmatic theory of morphology** as defined in Tufis (1990). The terminology used in the following is taken from the above-mentioned paper. In the same paper, it is shown that learnability is the great advantage of paradigmatic morphology. The *PARADIGM* system, described in Tufis (1989) and Tufis (1990) allows a novice user to informally teach the program how to (de)compose inflexional word-forms, that is to enable the morphological processing by a natural language processor.

THE MORPHO-LEXICAL KNOWLEDGE BASE

Obviously, the main depository of morpho-lexical knowledge is **the dictionary,** to be discussed in the following.

Other morphological knowledge sources are **the endings tree** and **the paradigms table.** These data structures do not depend on a specific lexical stock because they encode general linguistic knowledge for the language in case (parts of speech, relevant categories for the inflexional behaviour, endings, paradigms, etc.). Since their organization and acquisition are described elsewhere ((Tufis 1989) and (Tufis 1990)) we will not dwell on them.

THE DICTIONARY

In our system, the dictionary is a two-way accessible collection of hierarchically structured entries. During parsing, the access is provided by a **root index.** Each root in this index is associated with one or (in case of root-homonymy) more dictionary entries. During generation, the access is ensured by a **meaning index.** Each symbol in this index labelling a meaning description structure (see below) is associated with one or (in case of synonymy) more dictionary entries.

The formal structure of a dictionary entry is described by the regular expression below:

$$\begin{aligned}\langle entry\rangle ::= (&\langle lemma\rangle\ \langle part\text{-}of\text{-}speech\rangle\\ &(\langle valency\text{-}model\rangle\ \langle semantic\text{-}description\rangle^*)^*\\ &(\langle non\text{-}regular\text{-}root\rangle\ \langle paradigmatic\text{-}description\rangle^*)^*\\ &(\langle phono\text{-}hyphen\rangle)^*\\ &(\langle syntagmatic\text{-}description\rangle)^*)\end{aligned}$$

where:

<lemma> and *<part-of-speech>* have the usual meaning.

<valency-model> is a list of idiosyncratic features of interest mainly for syntactic processing (syntactic patterns, required prepositions, positions with respect to the dominant constituent for adjectives and adverbs, etc.).

<semantic-description> is the name of a **case-frame structure** placed in a generic-specific hierarchy. The actual semantic descriptions reside in a different data space than the rest of the dictionary. This separation is motivated by various reasons, among them being:

- the intention to enable for a meaning-based transfer, via the semantic descriptions area, between monolingual dictionaries;
- the capability of interchanging domain-oriented semantic descriptions;
- the lexical stock independence from the meaning representation formalism;
- a more precise treatment of synonymy, antonymy and generalization-specialization relations.

Concerning the last reason invoked above, it is quite obvious that synonymy, antonymy or generalization-specialization relations cannot be established directly between dictionary entries. This is because such relations, more often than not, are defined over specific meanings of a pair of words and rarely a word is monosemantic. On the other hand, such relations are frequently domain dependent. Therefore, we let them be expressed between semantic case-frames (descriptors of individual meanings), but, because the meaning representation of the lexical stock is beyond the purpose of this paper, we will not refer to it.

<non-regular-root> and the *<paradigmatic-description>*s describe — for non-regular inflecting words — the conditions under which the *<non-regular-root>* may be considered in forming a word-form. A formal definition of what we call **non-regular inflecting**, as opposed to the **regular inflecting**, is given in Tufis (1989). Informally, a word is a regular-inflecting one iff any grammatical form of it may be written as *<constant-part>* + *<ending>*. The *<constant-part>* is called the **regular root** of the word. If a word is not a regular-inflecting one, it is called non-regular. One may note that a non-regular inflecting word is characterized by more than one root. These roots are called **non-regular-roots**. A *<paradigmatic-description>* is a bit-map codification for the endings in a paradigm which may be combined, under a feature-values set of restrictions, with the *<non-regular-root>*.

<phono-hyphen> is a place-holder for the pronunciation transcription of the lemma or of the non-regular roots. This field also contains information about the hyphenation of the corresponding item.

<syntagmatic-description> is a parameterized pattern, describing groups of words which are to be recognized or generated as stand-alone processing units. Given the importance of what we called syntagmatic processing (probably the most attractive feature of our system) we shall devote the next section to the presentation in greater detail of this topic.

THE SYNTAGMS

We mean by **syntagm** a sequence of at least two lexical items which are to be **processed as a single unit**. In accordance with this definition, the collocations, idioms and analytical compounds are syntagms.

A syntagm is represented in the dictionary as a pair *(<result> <pattern>)* and it is associated with the lemma of the entry in case. This lemma is called the **pivot element** of the syntagm and it may appear in whatever position of the sequence.

In order to clarify the syntagm processing let us examine its formal structure:

<syntagm> ::= (<result> <pattern> <position-of-pivot-in-pattern>)
<pattern> ::= (<element> <element>$^{+}$)
<element> ::= <word-form> | <lemma> | <category>
| <compound-element>)
<compound-element> ::= (<displacement> <lemma> <restriction>*)
| (<displacement> <category> <restriction>*)
| <choice-list>
<choice-list> ::= (<element>$^{+}$ <obligativity>)
<obligativity> ::= TRUE | FALSE
<displacement> ::= + | < | — | >
<restriction> ::= (<feature> <value>*)
<result> ::= (<syntagm-value> <restriction>*)
<syntagm-value> ::= NULL | <lemma>

The replacement element of a syntagm is either the empty string or a lemma which will be associated with the appropriate morpho-lexical features as resulted from its processing. This lemma may correspond to an element in the <*pattern*> specially marked as syntagm substituter and in this case <*syntagm-value*> is *NULL* (the empty replacement string corresponds to the *NULL* value of <*syntagm-value*> and no substituter element in the <*pattern*>).

The <*element*> in the <*pattern*> of a <*syntagm*> may be a word-form, an (un)restricted lemma, an (un)restricted grammar category or any one in a choice list of specified <*element*>s. In case of a choice, if <*obligativity*> is *FALSE*, besides the specified <*element*>s, the empty string is a valid candidate too.

The <*displacement*> specified in a <*compound-element*> of the pattern of a <*syntagm*> determines the role to be played further on by the considered element. The meaning of the value in this field depends on whether the syntagm is to be recognized or generated:

- the value '+' specifies that the current element is either the replacer of the syntagm (during analysis), or one of the elements of the syntagm expansion, in the specified position (during generation);
- for analysis purposes, the values '<' and '>' specify that the current element is an *"alien"* constituent which must be transferred in front of or behind the syntagm replacer, respectively; during generation phase, the same values specify that the first item from the left or from the right of the syntagmatic item which is to be expanded will be moved — obeying the possible restrictions — to the output string, in the current position;
- the '—' value is the default and says that the element in case will either be deleted from the input string (during analysis) or inserted in the output string (during generation).

The <*restriction*>s are the principal means by which a lexicon designer expresses the rules governing the correct use of a syntagm. Depending on its format, the meaning of a <*restriction*> differs:

a) *(feature)*
In this case, the first (from the left to the right) matching value of the feature discussed has to be the same for all subsequent occurrences of the a-type restrictions over the same feature. This type of restriction is used to express feature congruency for different constituents appearing in the <*pattern*> of a syntagm as well as the inheritance of a feature value from the <*pattern*> to the <*result*> or vice-versa.

b) *(feature value)*
A <*pattern*> element restricted like that must match (during analysis) an input item having the specified value for the feature in case. In generation phase, it represents a word-forming parameter. If the restriction is associated with the <*result*> it simply represents an assignment (in case of analysis) or an expanding parameter (in case of generation).

c) *(feature value$_1$ value$_2$,... value$_n$)*
Such a restriction may act on each feature only once

in the *<pattern>* and once in the *<result>*. The paired multiple-valued features (one from the *<pattern>* and one from the *<result>*) positionally specify the relations between the values of a feature existing in both *<pattern>* and *<result>*. That is, if, during analysis, a *<pattern>* element matched an input item having for a given feature, say f_m, one of the values specified in its restriction, say the k^{th}, then the feature f_m will be assigned in the *<result>* the k^{th} value in its associated restriction. With generation, things are similar.

In Tufis and Popescu (1990b), the flow of control as well as the formal power of syntagmatic processing are outlined by means of annotated examples of syntagms codifying the rules governing the compound verbal forms (including interrogative forms and *"aliens"* (adverbs, reflexive pronoun insertion) for English, French, Romanian, Russian and Spanish.

As an example we give below a syntagm describing one of the possible ways of forming two negative analytical verbal forms (passé-composé and plus-que-parfait) in French:

```
((NULL (pérsonne) (nombre) (genre) (modalité négative)
       (temps passé-composé plus-que parfait))
 ("ne"
  (— être (pérsonne) (nombre) (temps présent imparfait))
  (> ADVERBE (modalité négative))
  (+ VERBE (temps participe-passé) (nombre) (genre)))
 2)
```

A more elaborated example, describing the basic compound tenses in English (not including the syntagms for handling adverbs insertion or negative and interrogative constructions) is the following:

```
(1)  ((NULL (VOICE ACTIVE) (ASPECT CONTINOUS) (TENSE))
      ((— BE (VOICE ACTIVE) (ASPECT INDEFINITE) (TENSE))
       (+ VERB (TENSE PRESENT-PARTICIPLE)))
      1)

(2)  ((NULL (VOICE PASSIVE) (ASPECT INDEFINITE) (TENSE))
      ((— BE (VOICE ACTIVE) (ASPECT INDEFINITE) (TENSE))
       (+ VERB (TENSE PAST-PARTICIPLE)))
      1)

(3)  ((NULL (VOICE PASSIVE) (ASPECT CONTINOUS)
            (TENSE PRESENT PAST))
       ((— BE (VOICE ACTIVE) (ASPECT CONTINOUS)
              (TENSE PRESENT PAST))
        (+ VERB (TENSE PAST-PARTICIPLE)))
       1)

(4)  ((NULL (VOICE ACTIVE) (ASPECT INDEFINITE)
            (TENSE SIMPLE-FUTURE PRESENT-CONDITIONAL))
       ((— SHALL (TENSE PRESENT PAST))
        (+ VERB (TENSE PRESENT-INFINITIVE)))
       1)
```

(5) ((NULL (VOICE ACTIVE) (ASPECT INDEFINITE)
(TENSE PRESENT-PREFECT PAST-PERFECT FUTURE-PERFECT
PAST-CONDITIONAL PERFECT-INFINITIVE))
((— HAVE (VOICE ACTIVE) (ASPECT INDEFINITE)
(TENSE PRESENT PAST SIMPLE-FUTURE
PRESENT-CONDITIONAL PRESENT-INFINITIVE))
(+ VERB (TENSE PAST-PARTICIPLE)))
1)

THE ENDINGS TREE AND THE PARADIGMS TABLE

The endings tree (a **discrimination tree**) is a knowledge source for the parsing process. Internally, it represents all the known **endings** (we use the term 'ending' without further noticing its eventual structure — e.g. suffix + desinence), and their morphological feature values. The nodes are labelled with letters appearing in different endings. A proper ending is represented by the concatenation of the letters labelling the nodes along a certain path, starting from a **terminal node** towards the root of the tree (this organization is due to the **retrograde parsing strategy** (Kotkova 1985) used in our system). A terminal node is not necessarily a leaf node because of the possibility of including one ending into a longer one. Such a case is called **intrinsic ambiguity**. All terminal nodes are attached to the paradigmatic information specific to the endings they stand for. More often than not, an ending does not uniquely identify a paradigm but a set of paradigms. In this case, the ending is called **extrinsically ambiguous**. Both types of ambiguity are theoretically solved by checks on the congruency between paradigmatic information attached to the respective endings (taken from the endings tree) and the candidate roots (taken from their dictionary entries).

The paradigms table is the data structure used during the word-form generation process. The paradigms are automatically classified during the learning (acquisition) phase (Tufis 1990) into an inheritance hierarchy. A **compilation phase** transforms this hierarchy into the paradigms table. The internally assigned code of a given paradigm is used as the index in the paradigms table, an entry of which has the following structure:

<fixed-feature-values>
<variable-feature-values> *<ending>*$^{+}$

The *<fixed-feature-values>* field represents a list of morphological features with predetermined values for the paradigm in case. These feature-values (if any) are collected while compiling the paradigms hierarchy and represent the discrimination criteria, according to which a more general paradigm is split into different specific paradigms. The *<variable-feature-values>* represents a list of (ordered) morphological features which may take any value out of the legal ones. An efficient numeric algorithm converts an arbitrary ordered set of feature-values into a code used as a displacement identifying the appropriate *<ending>* in the current entry of the table. Let us mention that the variable features have **default values**, so that, even if the generation criteria set was not completely specified, an inflected word-form is still generated. Moreover, if the endings tree or the paradigms table are not defined, the system does not crash but instead functions as if it had been designed for a word-form dictionary (the trivial morphology approach).

FINAL REMARKS

Due to the lack of space, we will discuss here neither the processing units of our environment nor the control flows between them. The interested reader may find all the necessary details in Tufis and Popescu (1990a) and Tufis and Popescu (1990b). Yet, we have to say that the proper morpho-lexical processings (analysis and generation), were thought to work in a **concurrent manner**. For instance reading characters from the keyboard, parsing individual word-forms, spelling checking and parsing syntagms are usually simultaneously active processes; similarly, individual word-forms generation and syntagms expansion are typical coroutines.

It is worth mentioning that, by default, the result of parsing as provided by our system is not a linear sequence of unique lexical items. The result includes all valid interpretations of every word in the input (including unknown words) thus generating **lexically ambiguous elements**, as well as all legal groupings of syntagmatic components thus genera-

ting **lexically ambiguous structures.**

If such a complete analysis is not desirable, a set of **general-purpose heuristics** may be used to filter the parsing (for instance when a word may be segmented in different ways, taking into account only the roots corresponding to the longest endings, considering the syntagms with the maximum number of constituents, etc., see Tufis (1990)).

With respect to spelling errors recovery, we distinguish between **typing** and **linguistic anomalies.** The typing errors are the usual misspellings taken into account by the spelling checkers of text editors. Anyway, there is an important difference: because (normally) our dictionaries are root-oriented, the standard spelling checking refers to the roots. With the endings, due to the limited number and limited length and thanks to the discriminating organization of the endings tree, the recovery is much more precise (the recovery is always complete when the root was found in the dictionary).

The case when the morphological features of the root of a word-form are not completely congruent with the morphological features of its recognized ending is considered a linguistic error. The congruency checking allows for an easy recovery of such mistakes. The distinct treatment of this type of error is very useful in case of CAI systems for language learning (Zock et al. 1990) and we intend, in the near future, to provide an explanation module to the congruency checker for such applications.

The generation process is bound to the morphological level, i.e. the lexical items are produced by a higher level module in the order they are supposed to appear in the output natural language string.

An exception from this rule is given by syntagmatic symbols generation. As previously shown, the pattern of a syntagm may specify one or more *"alien"* constituents (such as adverbs or pronouns). While expanding such a pattern, a '<' or '>' - marked constituent is imported into the syntagmatic sequence from the left or from the right of the syntagmatic symbol, thus changing the initial ordering.

The *MORPHIS* system, described in this paper is partially implemented in *GOLDEN COMMONLISP* for *IBM PC-AT* compatible personal computers.

At present, a user-friendly interface is under development, which is supposed to decrease as much as possible the level of expertise required to a user in order to build his/her own morphological knowledge base.

The interface will also include on-line consulting facilities and the system will be equipped with configuration possibilities and standard linking interfaces for three main types of applications: advanced text-editing, language-learning and machine translation (including NL interfaces).

REFERENCES

Byrd, R.J.; Calzolari, N.; Chodorow, M.S.; Klavans, J.L.; Neff, M.S. 1897 Tools and Methods for Computational Linguistics. *Journal of Computational Linguistics* 13(3-4) (special issue on lexicon): 219-240.

Koktova, E. 1985 Towards a New Type of Morphemic Analysis. *Proceedings of the Second Conference of ECACL*, Geneva, Switzerland: 179-186.

Nirenburg, S.; Raskin, V. 1987 The Subworld Concept and the Lexicon Management System. *Journal of Computational Linguistics* 13(3-4) (special issue on lexicon): 276-289.

Ritchie, G.D.; Pullman, S.G.; Black, A.W.; Russell, G.J. 1987 A Computational Framework for Lexical Description. *Journal of Computational Linguistics,* 13(3-4) (special issue on lexicon): 290-307.

Tufis, D. 1989 It Would Be Much Easier if WENT Were GOED. *Proceedings of the 4-th Conference of ECACL*, Manchester, England: 145-152.

Tufis, D. 1990 Paradigmatic Morphology Learning. *Computers and Artificial Intelligence*, 9(3): 273-290.

Tufis, D.; Popescu, O. 1990a The MORPHIS User Manual. ICI, Bucharest, Romania (in Romanian).

Tufis, D.; Popescu, O. 1990b Processing Idioms and Analytical Compounds Within an Integrated Dictionary Environment. Research Report, ICI, Bucharest, Romania.

Zock, M.; Laroui, A.; Francopoula, G. 1990 <<See What I Mean?>> Interactive Sentence Generation as a Way of Visualising the Meaning-Form Relationship. *Proceedings of the Fifth World Conference on Computers in Education*, Sydney, Australia.