Morphological Categorization using Attribute Value Trees and XML

Stefan Diaconescu

SOFTWIN Str. Fabrica de Glucoza Nr. 5, Sect.2, 72246 Bucharest, ROMANIA
sdiaconescu@softwin.ro

Abstract. Abstract. In order to handle information into the natural language processing systems the morphological categorization of parts of speech POS must be represented in two kinds of forms: external - most human readable form and internal - most computer readable form. This document presents a General Model that contains an external form to represent morphological categorization based on attribute value trees AVT and two internal forms based on XML: one for general morphological information and one for particular information attached to a POS. Finally it is presented a LIR model that can efficiently be used for strong inflected languages to generate inflected forms of POS, to spell and annotate POS in a text and to full indexing a text.

1 Introduction

From a certain point of view we can consider that there are two directions in which the NLP (Natural Language Processing) is developed: one stress on the lexicon and the other one stress on the syntax. If we try to put too much linguistic information in the lexicon, we can arrive in the situation where for each POS (Part of Speech) we must consider not only information about its intrinsic value but also a lot of information about its relation with the others POS. If we try to put too much linguistic information in the syntax description, we can find ourselves in the situation to consider all the POS of a language with all their inflected forms as terminals. In both cases the rule numbers (or the description volume) we need becomes too great and difficult to manage. We consider that there is a middle way, where the syntactic description is detailed only to the level of pseudo terminals that have attached a set of categories with their values. The morphological description will contain all the rules that are observed by morphological categories and will associate these categories and their values to the information from the lexicon. In this paper we will describe a General Model that contains a mode to structure morphological information. This model (section 2) and AVT (Attribute Value Trees) [1] and XML [3]. The three steps of the model are described in the sections 3,4,5. The morphological information from the General Model can then be exploited in a particular model. In the section 6 we will describe such a model named LIR (Lexicon, Inflection, Roots).

2 The General Model

We will present here a language processing model, the General Model that can be used to process and store linguistic information. The main components of the General Model are (see the figure 1):

- *The Morphological Configurator Creating (1).* It has as input the general morphological knowledge (usually taken from the classical grammars) and has as output an AVT morphological configurator (2).
- *The conversion from the AVT format to XML format (3).* It has as inputs the AVT morphological configurator and the morphological configurator DTD. It has as output the morphological configurator in XML format (4).
- *The morphological knowledge acquisition (5).* It has as inputs the XML morphological configurator (4), the particular morphological knowledge (POS from the lexicon and rules and exceptions taken from the grammars) and the Inflected POS DTD. It has as output the detailed morphological knowledge in XML format (6).
- *The particular model generation (7).* Using the detailed XML morphological knowledge (6) different particular models can be generated (8).

3 AVT Morphological Configurator

A Morphological Configurator is a formal description of the morphological structure of a language. It is based on AVT (Attribute Value Tree). We have not enough space here to give all the syntax of the AVT representation. We will indicate only few hints about this representation.

The Morphological Configurator contains an AVT that has associated to each node attributes (category) and attribute values. Each attribute has associated one or many values. Each value can have zero, one or many attributes.

Each attribute has associated some features like:

- inflection: indicates if the category is inflected or not;
- treatment: indicates the name of a procedure that must make the treatment of the category in a Natural Language Processing System (NLP).

Each value has also associated some features like:

- lemma: indicates if the category value will be associated to the POS lemma;
- lexicon: indicates if the category value is associated to a POS that will be an entry in the lexicon (if lexicon = entry) a supplementary entry in the lexicon (if lexicon = supplement) or not an entry in the lexicon (if lexicon = no);

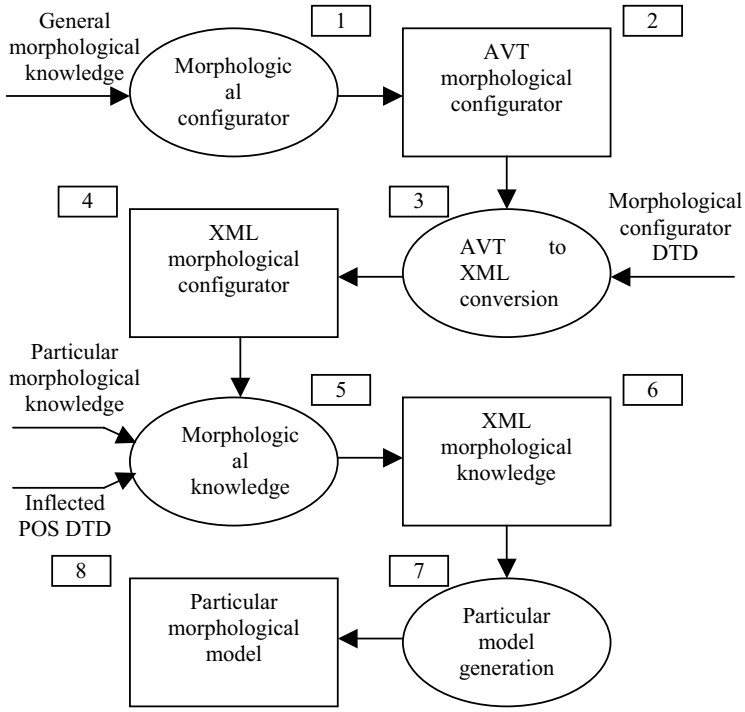

Fig. 1. The General Model

- indicative: indicates if the category value is associated or not to a POS that will be used by an automatic treatment in a NLP to generate the other inflected forms of this POS;
- treatment: indicates the name of a procedure that must make the treatment of the category value in a NLP.

An attribute can have associated a label (the definition) that can be used later in a place where it will be substituted by its definition. An attribute list associated to an attribute value can have also a label that can be used later in a place where it will be substituted with its definition. Using this mechanism (similar with the HPSG reentrancy) the AVT description is much more compact. For example, the AVT for Romanian language morphological configurator is about four times smaller.

The Morphological Configurator contains also a label dictionary section that gives in different languages the categories, category values, some abbreviations and comments. Therefore the AVT description can be prepared to be used in different exploitation systems implemented in different languages.

4 XML Morphological Configurator

The AVT morphological configurator can be converted to an XML format compatible with a specific DTD. We present here this DTD named Configurator.

```
<?xml version="1.0" encoding = "UTF-8"?>
<!ELEMENT configurator    (attribute, labelDictionary*)>
<!ELEMENT attribute       (name, value*)>
<!ELEMENT name            (NMTOKEN)>
<!ELEMENT abbreviation    (NMTOKEN)>
<!ELEMENT value           (name, attribute*)>
<!ELEMENT labelDictionary (label)+>
<!ELEMENT label           (name, translation, abbreviation)>
<!ELEMENT translation     (#PCDATA)>
<!ELEMENT abbreviation    (#PCDATA)>
<!ATTLIST configurator
          sourceLanguage NMTOKEN #REQUIRED>
<!ATTLIST attribute
          inflection     (yes | no)          #REQUIRED
          treatment      CDATA               #IMPLIED
          comment        CDATA               #IMPLIED>
<!ATTLIST value
          lemma          (yes | no)          #REQUIRED
          lexicon (entry | supplement | no)  #REQUIRED
          hint           (yes | no)          #REQUIRED
          comment        CDATA               #IMPLIED>
<!ATTLIST labelDictionary
          usingLanguage  NMTOKEN             #REQUIRED>
```

The tag significance can be easily deduced from the previous section.

5 XML Morphological Knowledge

Based on the Inflected POS DTD and XML morphological configurator a linguist can generate the XML morphological knowledge using the particular morphological knowledge (specific to a certain language). This XML morphological knowledge is compatible with a specific DTD. We will present now such an inflected POS DTD named Wordlist that describes the structure of the information about each word from lexicon.

```
<!ELEMENT wordlist        (word+, labelDictionary*)>
<!ELEMENT word            (form, (attribute, value)+)>
<!ELEMENT form            (#PCDATA)>
```

```
<!ELEMENT attribute       (NMTOKEN)>
<!ELEMENT value           (NMTOKEN)>
<!ELEMENT labelDictionary (label)+>
<!ELEMENT label (name, translation, abbreviation)>
<!ELEMENT name            (NMTOKEN)>
<!ELEMENT translation     (#PCDATA)>
<!ELEMENT abbreviation    (#PCDATA)>
<!ATTLIST form
          lexicon (entry | supplement | no) #REQUIRED>
<!ATTLIST wordlist
          source_language NMTOKEN #REQUIRED>
<!ATTLIST attribute
          inflection      (yes | no)          #REQUIRED
          treatment       CDATA               #IMPLIED
          comment         CDATA               #IMPLIED>
<!ATTLIST value
          lemma           (yes | no)          #REQUIRED
          lexicon (entry | supplement | no)   #REQUIRED
          hint            (yes | no)          #REQUIRED
          comment         CDATA               #IMPLIED>
<!ATTLIST labelDictionary
          usingLanguage NMTOKEN               #REQUIRED>
```

The filling of the XML file with the particular morphological knowledge for each POS is not at all an easy task. It can be realized with a special application that works partially automatic (for the regular rules) and partially manual (for exceptions).

6 LIR Model

Starting from the information that we have in the XML morphological knowledge we can generate different natural language processing models that we will use in different linguistically purposes. We will present here very shortly such a model that we will name LIR because it is based on three main structures: the Lexicon, the Inflection rule collection and the Root collection.

a) The Lexicon contains usually the word lemmas, but also other supplementary forms. We will consider that the lemma is the form of the word on which applying some transformations we will obtain all the other inflected forms of the word. Each lemma will have associated all the corresponding attributes - value pairs. From the previous section results how the lexicon entries can be deduced.

b) The Inflection Rule Collection is a set of rules that explain how each inflected form of the word can be obtained from its lemma. Thus the Inflection Rule Collection will contain a rule for each lemma. Many lemmas can have

associated the same rule. Each rule will contain an elementary rule for each inflected form. Each elementary rule contains a set of transformations that applied to the lemma give the inflected form. There are many possibilities to describe these elementary rules like the two level model [9]. We will present here an XML DTD that can be used to describe the structure of the Inflection Rule Collection.

```
<?xml version="1.0" encoding = "UTF-8"?>
<!ELEMENT inflectionRules (rule+, labelDictionary*)>
<!ELEMENT rule (inflection+)>
<!ELEMENT inflection ((attribute, value)+,
                      ((insert | delete | replace | add)+)?)>
<!ELEMENT attribute       (NMTOKEN)>
<!ELEMENT value           (NMTOKEN)>
<!ELEMENT insert          (#PCDATA)>
<!ELEMENT delete          (EMPTY)>
<!ELEMENT replace         (#PCDATA)>
<!ELEMENT add             (what)>
<!ELEMENT what            (word)>
<!ELEMENT word            (#PCDATA)>
<!ELEMENT labelDictionary (label)+>
<!ELEMENT label (name, translation, abbreviation)>
<!ELEMENT name            (NMTOKEN)>
<!ELEMENT translation     (#PCDATA)>
<!ELEMENT abbreviation    (#PCDATA)>
<!ATTLIST inflection_rules
          source_language NMTOKEN           #REQUIRED>
<!ATTLIST insert
          from            (begin | end)     #REQUIRED
          position        CDATA             #REQUIRED>
<!ATTLIST delete
          from            (begin | end)     #REQUIRED
          to              (begin | end)     #IMPLIED
          position        CDATA             #REQUIRED
          length          CDATA             #REQUIRED>
<!ATTLIST replace
          from            (begin | end)     #REQUIRED
          to              (begin | end)     #IMPLIED
          position        CDATA             #REQUIRED
          length          CDATA             #REQUIRED>
<!ATTLIST add
          where           (before | after)  #REQUIRED>
<!ATTLIST word
          exclusive       (yes | no)        #REQUIRED>
<!ATTLIST label_dictionary
```

```
using_language   NMTOKEN          #REQUIRED>
```

The transformations described by this DTD consist of:

- The insertion (insert) of some characters after a number (position) of lemma's characters numbered from (from) the beginning (begin) or from the end (end) of the lemma.
- The deletion (delete) of some characters in the lemma. The first position (from) is indicated by the character number (position) numbered starting from the beginning (begin) or the end (end) of the lemma. The second position (to) is indicated also by the number of characters (length) numbered starting from the beginning (begin) or the end (end) of the lemma. If the attribute to is missing, then the attribute length indicates the length of the string to be deleted.
- The replacement (replace) of some characters (the attributes from, to, position, length have the same meaning as in case of delete.)
- The adding (add) of some new words (word). These words can be added (where) before the lemma (before) of after the lemma (after). The exclusive attribute indicates if the corresponding word can be followed (yes) by a non specified sequence or not (no).

c) The Root Collection. By root we understand here a character sequence common for many inflected forms of a word. A root can be defined in many ways. We consider generally that the common root of set of words is the longest common sequence of characters between the words of the set. A word can have many roots if the inflected forms are very different. For example, for the French verb aller we have for indicative present the inflected forms: (je) vais, (tu) vas, (il) va, (nous) allons, (vous) allez, (ils, elles) vont. If we consider a minimum length of a root to one character, then there will be the roots: v- (for the inflected forms vais, vas, va, vont) and all- (for the inflected form allez). If we consider a minimum length of a root to two characters, then there will be the roots: va- (for the inflected forms vais, vas, va), and all- (for the inflected form allez) and vont (for the inflected form vont). The Root collection contains all the roots considered for all the inflected forms that are in the lexicon. For each root there is a pointer to the corresponding word in the lexicon.

In a speller the root collection can be used as follows. We consider different possible roots for the word that we want to check. Each generated root is searched in the root collection. For each found root in the root collection we search the corresponding word in the lexicon. For each different form found in the lexicon we take the inflected rules form the Inflection Form Collection and we generate with these rules the inflected forms that can be generated. If the checked word is found among these generated inflected forms, then the word is a correct form.

In an annotation process of a text we proceed in the same way but we can keep also from the Inflection Form Collection the corresponding AVT

trees associated to the inflected forms that match the checked word. We will obtain one or more interpretations of the checked word.

7 Conclusions

The General Model and the LIR Model we presented here are based on three basic data structures: AVT morphological configurator, Morphological Configurator XML DTD and Inflected POS XML DTD. These data structures are conceived to be used in a computer natural Language processing system. In such a system it the linguistic information belongs to a language and the exploitation user interface belongs possibly to another language. We presented also an Inflection Rule DTD that can be used to store the inflection rules and also the corresponding AVT associated with each inflected form. The AVT morphological configurator formalism was introduced in a more general language GRAALAN (Grammar Abstract Language) that are designated to describe the linguistic knowledge used in machine translation systems. A complete AVT morphological configurator for Romanian language (that is a high inflected language) was realized. The Romanian AVT morphological Configurator (version 3) contains more than 800 category nodes and more that 1400 category value nodes.

References

1. Diaconescu, Stefan (2002) Natural Language Understanding using Generative Dependency Grammar, in Proceedings of the twenty-second Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence (SGES), (ES2002), Liverpool, UK
2. EAGLES (1996) EAGLES Recommendation on Subcategorisation, EAGLES DOCUMENT EAG-CLWG-SYNLEX, Version of 31st August 1996
3. Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation, 6 October 2000.
4. IANA-LANGCODES (Internet Assigned Numbers Authority) Registry of Language Tags, ed. Keld Simonsen et al. (See http://www.isi.edu/in-notes/iana/assignements/language).
5. IETF RFC 1766 IETF (Internet Engineering Task Force). RFC 1766: Tags for the Identification of Languages, ed. H. Alvestrand. 1995. (See http://www.ietf.org/rfc/rfc1766.txt).
6. ISO 639 (International Organization for Standardization). ISO 639:1988 (E). Code for the representation of names of languages. [Geneva]: International Organization for Standardization, 1988
7. ISO 3166 (International Organization for Standardization). ISO 3166-1:1997 (E). Codes for the representation of names of countries and their subdivisions – Part 1: Country codes [Geneva]: International Organization for Standardization, 1997
8. Koskenniemi, K. (1983) Two-level morphology: A general computational model for word-form recognition and production, Publication No. 11, Department of General Linguistics, University of Helsinki.