KNOWLEDGE ENGINEERING: PRINCIPLES AND TECHNIQUES
Proceedings of the International Conference on Knowledge Engineering,
Principles and Techniques, KEPT2007
Cluj-Napoca (Romania), June 6–8, 2007, pp. 96–103

NATURAL LANGUAGE GENERATION: APPLICATIONS FOR ROMANIAN LANGUAGE

CIPRIAN-IONUT DUDUIALĂ [1]

Abstract. Natural Language Generation (NLG) and Foreign Language Writing Aid (FLWA) are two important tasks of Natural Language Processing (NLP), which deal with obtaining natural language from a machine representation system and building computer programs that assists a non-native language user in writing decently in a target language, respectively. This paper uses both NLG and FLWA. Suppose a person wants to translate a sentence from English to Romanian language, but he/she does not speak Romanian. The first thing to be done is to take a dictionary, find the corresponding words, put them together and form the sentence, but a lot of disambiguities might arise. Using an affix grammar to construct the Romanian language grammar and a semantic which gives us information about the words we use to build a sentence, we can construct, starting from a set of words, correct sentences from sintactic and semantic point of view. This paper deals only with short sentences.

1. Introduction

Translating a sentence from English to Romanian means more than translating word with word using a dictionary . We risk to obtain phrases that make no sense.

First of all, the words order in a phrase is different. *"Is Dory online?"* is translated *"Dory este online?"* due to the fact that in Romanian language the subject appears before the predicate even when a question is constructed. Next, in English, for some tenses of verbs, the same form is used for different persons, while Romanian language uses different forms. In addition, a dictionary does not contain the verb forms for every tense and person. Another ambiguity can be generated by an adjective that determines a noun, since in English the adjectives have only one form. They do not depend on the genre or number of the substantive, but Romanian language uses different forms for adjectives.

This paper takes into consideration only short propositions containing a subject, a predicate and, if needed, some complements. The subject and the complements can be followed by at most one attribute. Conjuctions like *"si"* or *"sau"* and

Key words and phrases. natural language generation, affix grammar.

punctuation marks will be omitted. This means the propositions cannot have a multiple subject.

2. Theoretical Support

The main idea comes from [1], where models for source code generators are created. Starting from a language grammar and having some logical relations and simple statements as input, valid source code for a given programming language is obtained. In what follows, the simplest model from that paper is analyzed.

Let $G = \{\{S, C\}, \{\Delta, |-, \varphi, \Omega, a, b, c_1, c_2, (,), ", "\}, P, S\}$ be the proper grammar with the set of productions P:

$$\begin{aligned} S &\rightarrow (S,S)||-(C,S)|\Delta(C,S,S)|\varphi(C,S)|\Omega(S,C)|a|b \\ C &\rightarrow c1|c2 \end{aligned}$$

In the grammar definiton, the fundamental control structures are encoded: *"(,)"* means *"concatenation"*, *"|−"* means *"if whith one branch"*, *"Δ"* means *"if with two branches"*, *"φ"* means *"while"* and *"Ω"* means *"repeat"*. Let a and b be two fundamental structures and logical expressions. Then:

- $a < b$, if a always appears before b in the generated programs;
- $a > b$, if a always appears after b in the generated programs.

In the following example, a and b are fundamental structures, while *c1* and *c2* are logical expressions:

	a	b	c_1	c_2
a	-	$<$	$<$	$<$
b	$>$	-	$<$	$<$
c_1	$>$	$>$	-	$<$
c_2	$>$	$>$	$>$	$>$

Table 1. Source code generators - semantic example.

The matrix is anti-symmetric, thus the defined model is consistent. For the above example, the set of words produced by applying exactly one production which does not involve only terminal symbols is reduced to S={(a, b), (a, c1), (b, c1), (a, c2), (b, c2)}. The programs are equivalent with the Pascal programs:

```
a;  REPEAT a   REPEAT b   REPEAT a   REPEAT b
b;  UNTIL c1;  UNTIL c1;  UNTIL c2;  UNTIL c2;
```

A similar model solves our problem. An appropriate grammar helps building correct sentences from syntactical point of view. More than that, keeping records about the forms of the words (substantives, adjectives, verbs, pronouns, adverbs and prepositions) for singular and plural and for all three genres is useful to solve the ambiguities that might appear.

3. Romanian Language Grammar

This section introduces the Romanian language grammar, which is a simple grammar used to illustrate some examples. It can be enriched with other productions to reflect all forms of the sentences that can be created using Romanian words. An affix grammar with restrictions [2] is used, which is in fact a context free grammar with finite set-valued features, acceptable to linguists. A simple example is $G = \{\{A, B, C\}, \{a, b, c\}, P1, A\}$, with the set of productions *P1*:

$$\begin{aligned}
param &:: one; two. \\
A &\rightarrow (\{param :: one\}|\{param :: two\}), B(param) \\
A &\rightarrow (\{param :: one\}|\{param :: two\}), C(param) \\
B(one) &\rightarrow a \\
B(two) &\rightarrow b \\
C(param) &\rightarrow c
\end{aligned}$$

The restrictions are introduced through some parameters. In our case, *"param"* is the parameter, while *"one"* and *"two"* are the parameter values and are introduced by *"::"*. For grammar G, the conditions appear before rewriting and the following expressions are generated:

$$\begin{aligned}
A \rightarrow B(one) \rightarrow a \qquad & A \rightarrow B(two) \rightarrow b \\
A \rightarrow C(one) \rightarrow c \qquad & A \rightarrow C(two) \rightarrow c
\end{aligned}$$

which means that *C(param)* can be applied for all values of *param*. Suppose now that the conditions are after the rewriting, that is:

$$\begin{aligned}
param &:: one; two. \\
A &\rightarrow B(param), (\{param :: one\}|\{param :: two\}) \\
A &\rightarrow C(param), (\{param :: one\}|\{param :: two\}) \\
B(one) &\rightarrow a \\
B(two) &\rightarrow b \\
C(param) &\rightarrow c
\end{aligned}$$

In this case, $A \rightarrow B(param)$ is applied first. For all the values that param can take according to the condition imposed to this production, we will rewrite *B(param)*. Applying the same principle for $A \rightarrow C(param)$ we obtain:

$$\begin{aligned}
A &\rightarrow B(param) \rightarrow B(one) \rightarrow a \\
A &\rightarrow B(param) \rightarrow B(two) \rightarrow b \\
A &\rightarrow C(param) \rightarrow C(one) \rightarrow c \\
A &\rightarrow C(param) \rightarrow C(two) \rightarrow c
\end{aligned}$$

Next, the affix grammar for the Romanian language is created to simplify things a little from notation point of view. Let

$$\begin{aligned} GR \;=\; & \{\{SR, Prop, Substantiv, Subst\_Atr, Atribut, Predicat, Adverb, \\ & ListComplAdv, ListCompl, Compl\}, \{adjectiv(gen, nr), prepozitie, \\ & substantiv(gen, nr, caz, tip\_articol), pron\_pos(pers, genp, nrp, gen, nr), \\ & pronume(pers, gen, nr, caz), verb(pers, nr, timp), adverb, \\ & verb\_gerunziu, verb\_infinitiv\}, PR, SR\} \end{aligned}$$

be the Romanian language grammar. The parameters and their values for GR are:

$$\begin{aligned} nr,\ nrp &:: sing, pl. \\ gen,\ genp &:: masc, fem, neutru. \\ pers &:: I, II, III. \\ timp &:: prezent, viitor, perf\_comp, imperf, mmcperf, conj, cond\_opt. \\ tip\_subiect &:: subst, pron. \\ tip\_articol &:: hot, nehot. \\ caz &:: N, Ac, D, G. \end{aligned}$$

The set of productions PR is defined as follows:

$$SR \;\rightarrow\; (\{tip\_subiect :: subst\}|\{tip\_subiect :: pron\}), Prop(tip\_subiect)$$

$$\begin{aligned} Prop(subst) \;\rightarrow\; & ((\{gen :: masc\}|\{gen :: fem\}|\{gen :: neutru\})\&(\{nr :: sing\}| \\ & \{nr :: pl\})\&(\{tip\_articol :: hot\}|\{tip\_articol :: nehot\})), \\ & substantiv(gen, nr, N, tip\_articol)\ Predicat(III, gen, nr) \end{aligned}$$

$$\begin{aligned} Prop(subst) \;\rightarrow\; & ((\{gen :: masc\}|\{gen :: fem\}|\{gen :: neutru\})\&(\{nr :: sing\}| \\ & \{nr :: pl\})), Subst\_Atr(gen, nr, N) \\ & Predicat(III, gen, nr) \end{aligned}$$

$$\begin{aligned} Prop(pron) \;\rightarrow\; & ((\{gen :: masc\}|\{gen :: fem\})\&(\{nr :: sing\}|\{nr :: pl\})\& \\ & (\{pers :: I\}|\{pers :: II\}|\{pers :: III\})), \\ & pronume(pers, gen, nr, N)\ Predicat(pers, gen, nr) \end{aligned}$$

$$\begin{aligned} Subst\_Atr(gen, nr, caz) \;\rightarrow\; & (\{tip\_articol :: hot\}|\{tip\_articol :: nehot\}), \\ & substantiv(gen, nr, caz, tip\_articol)\ Atribut \end{aligned}$$

$$\begin{aligned} Subst\_Atr(gen, nr, caz) \;\rightarrow\; & (\{tip\_articol :: hot\}|\{tip\_articol :: nehot\}), \\ & substantiv(gen, nr, caz, tip\_articol) \\ & adjectiv(gen, nr) \end{aligned}$$

$$Subst\_Atr(gen, nr, caz) \rightarrow (\{tip\_articol :: hot\}|\{tip\_articol :: nehot\})\& (\{genp :: masc\}|\{genp :: fem\})\& (\{nrp :: sing\}|\{nrp :: pl\})\& (\{pers :: I\}|\{pers :: II\}|\{pers :: III\})), substantiv(gen, nr, caz, tip\_articol) pron\_posesiv(pers, genp, nrp, gen, nr)$$

$$Atribut \rightarrow ((\{gen :: masc\}|\{gen :: fem\}|\{gen :: neutru\})\&(\{nr :: sing\}| \{nr :: pl\})\&(\{caz :: Ac\}|\{caz :: G\})\&(\{tip\_articol :: hot\}| \{tip\_articol :: nehot\})), [prepozitie] substantiv(gen, nr, caz, tip\_articol)$$

$$Atribut \rightarrow ((\{gen :: masc\}|\{gen :: fem\})\&(\{nr :: sing\}|\{nr :: pl\})\& (\{pers :: I\}|\{pers :: II\}|\{pers :: III\})), prepozitie pronume(pers, gen, nr, Ac)$$

$$Predicat(pers, gen, nr) \rightarrow (\{timp :: prezent\}|\{timp :: viitor\}|\{timp :: conj\}| \{timp :: perf\_comp\}|\{timp :: mmcperf\}| \{timp :: imperf\}|\{timp :: cond\_opt\}), verb(pers, nr, timp)\ [ListComplAdv(gen, nr)]$$

$$ListComplAdv(gen, nr) \rightarrow [ListComplAdv(gen, nr)]Adverb(gen, nr) [ListComplAdv(gen, nr)]|ListCompl$$

$$Adverb(gen, nr) \rightarrow adjectiv(gen, nr)|adverb|verb\_gerunziu$$

$$ListCompl \rightarrow Compl|Compl\ ListCompl$$

$$Compl \rightarrow prepozitie\ verb\_infiniv$$

$$Compl \rightarrow ((\{gen :: masc\}|\{gen :: fem\}|\{gen :: neutru\})\&(\{nr :: sing\}| \{nr :: pl\})\&(\{caz :: Ac\}|\{caz :: D\})\&(\{tip\_articol :: hot\}| \{tip\_articol :: nehot\})), [prepozitie] substantiv(gen, nr, caz, tip\_articol)$$

$$Compl \rightarrow ((\{gen :: masc\}|\{gen :: fem\}|\{gen :: neutru\})\&(\{nr :: sing\}| \{nr :: pl\})\&(\{caz :: Ac\}|\{caz :: D\})), [prepozitie] Subst\_Atr(nr, caz)$$

$$Compl \rightarrow (((\{gen :: masc\}|\{gen :: fem\})\&(\{nr :: sing\}|\{nr :: pl\})\& (\{pers :: I\}|\{pers :: II\}|\{pers :: III\})\&(\{caz :: Ac\}|\{caz :: D\})), [prepozitie]pronume(pers, gen, nr, caz)$$

4. Simple Model for NLG Using Romanian Language Grammar

Using a semantic the amount of computations needed to determine a correct proposition is reduced. Each word used is considered an element of a set and a relation between the words is defined. The phrase: *"I always read the weather forecast in the newspapers"* is translated into Romanian *"Eu citesc intotdeauna previziunile meteo in ziare"*. Translating word by word, using a dictionary, the user obtains the set of words $S = \{eu, intotdeauna, a\ citi, vremea, previziuni, ziare\}$ and establishes, for example, the following relations between them:

	eu	a citi	intotdeauna	vremea	previziuni	ziare
eu	-	-	-	-	-	-
a citi	-	-	Pr	-	Pr	Pr
intotdeauna	-	C	-	-	-	-
vremea	-	-	-	-	A	-
previziuni	-	C	-	Wd	-	-
ziare	-	C	-	-	-	-

Table 2. NLG - simple model example.

In the above table, each entry has the following meaning:

- '-' → the 2 elements are not related
- 'Pr' → the word from the line is the predicate determined by the complement from the column
- 'C' → the word from the line is the complement which determines the verb from the column
- 'A' → the word from the line is the attribute which determines the word from the column
- 'Wd' → the word from the line is the word determined by the attribute from the column

Observe that the predicate of the proposition can be easily determined, since for sure one line has one or several of the entries with value *'Pr'*. Note also that a semantic has at least two words due to the fact that a subject and a predicate are needed for any proposition. Anyway, another question needs an answer: how can be determined the person and the number of a pronoun, for example? For that a database - like a dictionary (DEX) - that gives us information about words is used.

5. Extension of the Simple Model

The paper purpose is not to find new methods for NLG. It proves that using the simple idea of the models created in [1] and given a set of words correct sentences can be constructed using only those words. In our case, not only the semantic is important: creating propositions that make sense cannot be done at random. Not any adjective can be used to describe a substantive and not any adverb can be used together with a certain verb. Thus, the database needs some links between words showing if it makes sense to use them together - for example, specifying that *"munte"* can be used with the adjective *"imens"*, but makes no sense to be used with adjectives like *"scund"*. For the models defined in this paper, some problems appear for substantives in *dative* case if the links are not defined. All the forms of a substantive or adjective are also need - for different genres and numbers - and all the forms of a verb - for different tenses and persons.

Next, to reduce the number of propositions generated, the words order in the phrase is specified. We can say: *"Raul curge lin la vale"* or *"Raul curge la vale lin"* and so on. A semantic of the form of [Table 3] indicates which word can be in front of another and which can not be.

Definition: Let a and b be two words. Then:

- $a < b$, if a always appears before b in the generated propositions;
- $a > b$, if a always appears after b in the generated propositions;
- $a = b$, if a and b can appear in any order in the generated propositions.

Definition: A **simple semantic** is a table defining the order of words in a phrase and having the form of [Table 3].

		raul	a curge	la	vale	lin
sg	raul	-	$<$	$<$	$<$	$<$
prez	curge	$>$	-	$=$	$=$	$=$
-	la	$>$	$=$	-	$<$	$>$
sg	vale	$>$	$=$	$>$	-	$>$
adj	lin	$>$	$=$	$<$	$<$	-

Table 3. NLG - simple semantic example.

The above table shows that *"raul"* is always the first word, while *"lin"* always appears before *"vale"*. So, only *"Raul lin curge la vale"* is generated. Not specifying that *"lin"* is an adjective means it can also be considered an adverb, thus *"Raul curge lin la vale"* could also be generated. Two important issues can be deduced: (i) the prepositions can be introduced into the model, which helps avoiding inappropriate use of prepositions and (ii) some adjectives can be used as adverbs and the sense of the proposition is changed, depending on the interpretation given

to that adjective. Also, for example, the word "muncitor" can be used as adjective or substantive. Thus, specifying its function might be useful.

Another semantic that specifies which word is determined by another one can be introduced:

		raul	a curge	la	vale	lin
sg	raul	-	$<$	$<$	$<$	$< d$
prez	curge	$>$	-	$=$	=d	$=$
-	la	$>$	$=$	-	$< d$	$>$
sg	vale	$>$	d=	$> d$	-	$>$
adj	lin	$d >$	$=$	$<$	$<$	-

TABLE 4. NLG - semantic example.

In [Table 4] the letter *"d"* appears before the order sign. *"a d* b"*, with *"*"* from $\{>, <, =\}$ means a determines b, while *"a *d b"* means a is determined by b. Several combinations are possible: a substantive determined by an adjective or another substantive (attributes), for example. Moreover, a preposition appears before the word determined. Obviously, preposition means before a position, thus in the line corresponding to each preposition we have exactly once " $< d$", while in the corresponding column we have exactly once " $> d$". If this condition is not satisfied, we say that the semantic is *inconsistent*. If a word determines more than one other word the semantic is also considered *inconsistent*. The probability that in a short proposition a word is used twice is very small, but if this happens that word is added twice in the semantic.

6. Conclusions

This paper presents a way of generating senteces using a given set of Romanian words. An affix grammar for the Romanian language is introduced to ensure that the sentences are correct from syntactical point of view. Finally, in order to reduce the number of sentences generated and and to indicate the words order in a sentence, three different semantics are defined.

References

[1] Vasile Cioban, Ciprian Duduiala - *"A Case Tool Proposal for Source Code Generators"*, Proceedings of the Symposium *"Colocviul Academic Clujean de Informatica"*, pag. 147-153, 1-2 June 2006.

[2] http://www.cs.ru.nl/agfl/ - official site of AGFL formalism developed between 1991 and 1996 by the Computer Science Department of the Radboud University of Nijmegen.

(1) Centre for Mathematical Medicine and Biology, School of Mathematical Sciences, University of Nottingham, UK

E-mail address: cipriduduiala@yahoo.com