MORPHOLOGY IN MINIMAL INFORMATION GRAMMAR

DANKO ŠIPKA

Adam Mickiewicz University

in: Singh, R. and S. Starosta (ed.), Explorations in Seamless
Morphology, Sage Publications, 2003, p. 328-338.

0. Introduction

In this paper I will present several solutions in the morphological generator within the machine translation system called NeuroTran®. While morphology within this system is not seamles, it still has much in common with this idea. It is namely so that the system used morphological boundries if they can contribute to its efficiency, and it changes them if they are a hurdle to it.

I will first present the MT system and its grammar rules, then the treatment of morphology within these grammar rules, and finally the minimization strategies, which at the same time show the role of morphological boundries in the system.

1. NeuroTran® amd MIG

NeuroTran® is a software program developed by Translation Experts Ltd. (More about the company is available at: http://www.tranexp.com). It is intended to "do things with words". It is typically post-fordist and utilizes its knowledge base in various ways depending upon the specific options selected. It includes a morphological generator and analyzer, a dictionary with lexical lists extraction capability, sentence parsing and translation capabilities, as well as quantitative and qualitative analysis tool. In its sentence translation mode, the software acts as a bilingual transfer system. Since certain minimization strategies rely on transferring features and equivalents from one language to another, one could say that the project as a whole has some properties of a multilingual system.[1]

Crucial solutions within NeuroTran® are rooted in certain human cognitive faacilities. When faced with the high complexity of their environment, people often reach to heuristics and schemata in an effort to reduce the resources required to process

[1] I will here present only linguistic and general layout of the project skipping thus matters of programming. which I am not involved in. The programming of NeuroTran® and its tool called Dictman is conducted by Translation Experts programmers Dr. Nenad Končar, Sł awomir Pawlowski, and Vladimir Šipka.

cognitively such data (Fiske and Taylor, 1991; Kahneman, Slovic and Tversky 1982). We must first recognize the high complexity of the tasks NeuroTran® is intended to perform and then attempt to make it operate in a manner similar to the way in which human beings actually process information. Furthermore, we must at least attempt to utilize the properties of human cognitive processing in the process of preparing the knowledge base for the program.

NeuroTran® is based upon three crucial ideas. The first is that one needs to minimize to the greatest extent possible all the information required by the software to function and, then, allow it to acquire new information by reading texts and communicating directly with the user. The program sucessfully accomplishes this by using artificial neural networks, that is to say, it starts as a "cognitive miser" and then uses schemata to assimilate new pieces of information while at the same time adapting and changing old ones according to the new information.

The second main idea behind NeuroTran® is that one needs to reduce the effort required by the lexicographer, again by requiring minimum information or input. In other words, the dictionary and program creators function as "cognitive misers" on behalf of the user.

Finally, all NeuroTran data needs to be reusable so that all knowledge bases and functions are usable in various situations and for various fields of endeavors well as for different languages.

NeuroTran® uses a set of formal rules we have named Minimal Information Grammar (MIG). These rules operate on the basis of a bilingual labeled lexical list (the list of equivalents with their respective grammar and usage labels together with frequency data, etc.) and a representative corpus for both languages in any given pairing. The architecture of MIG is subordinate to the fundamental ideas behind NeuroTran®.

The grammar was named minimal because it reduces the information required for the software to perform its functions at a high level of speed and accuracy. It does this by: (1) balancing information in both the rules and the data it operates under; (2) using different classes of rules (constructors, mutators, selectors, etc.) to manipulate existing linguistic material, and; (3) using artificial neural networks to provide new information as a direct result of the learning process which occurs anytime the software "reads" a text or communicates with a user. Existing information is virtually and continuously “recycled”. MIG operates with the following classes of rules:

a. *constructors* - use dictionary labels to construct all possible forms of a word

b. *mutators* - change already generated forms

c. *finders* - find the form or word required

d. *definers* - decide what is what

e. *coordinators* - determine how one form coincides with others

f. *choppers* - divide larger units into smaller ones

g. *binders* - unite smaller units into larger ones

h. *transformers* - replace one word or form with another, for example, by translating a word in one language into a word in the other

i. *counters* - keep track of all statistics

j. *doubters* - detect situations where there are more possibilities than would allow the program to proceed

k. *gamblers* - choose the solution that (based upon everything in the database) is the most probable even though other options remain viable

l. *teachers* - change existing information (rules and figures) after reading different texts and translations

m. *chatters* - ask the user when they need a piece of information or if user wants to change something

n. *conductors* - direct the order in which the rules are to be applied

Every rule consists of a head (stating the input of the rule) and a body (providing details of how the output is calculated). This is represented in the Table 2 using an example of the English to Serbo-Croatian translation transfer rule for number-gender coordination between the nominal head and its adjective modifier:

Rule structure	**Rule**	**Example**
<rule head> =>	ENGSCR GRM N[ADJECTIVE|PRONOUN] NOUN =>	big book -> velik knjiga
<rule body line 1>;	COPY(2>1:NUMBER,GENDER)	velik knjiga -> velika knjiga
<rule body line 2>;		
....		
<rule body line N>		

Table 1

The entry in the labeled list of equivalents has the following structure:

<entry><grammatical labels><usage labels><frequency data><collocation data>
<equivalent 1><grammatical labels><usage labels><frequency data><collocation data>
<equivalent 2><grammatical labels><usage labels><frequency data><collocation data>
...
<equivalent n><grammatical labels><usage labels><frequency data><collocation data>

The text corpus data are attached to the program as text with an index pointing from each form in the dictionary to each its starting and ending byte in the text.

2. MIG Morphology

Morphological paradigms within MIG are basic information used by neural network to initiate and investigate all possible solutions in translation, parsing, and qualitative analysis. In order to generate a paradigm, MIG uses labels attached to the lexical entries in main dictionary text, and then applies a set of primarily constructor-type rules to the word bearing the label. It is well known that morphological generator and analyzer are in fact two sides of the same coin. Systems seeking minimality can thus develop one deriving the other from it. The idea of MIG is to develop the generator and to derive the analyzer from it.

The following dictionary entry:

selo,a n; *[...]/village n; [...]*

is used to generate both its forms and the analyzer’s entries:

Rule	***Explanation***	***Generated forms***	***Generated analyzer’s entries***
SCR PARA *o,a n =>	Head of the rule. If you find an entry ending in *o,a n; do the following (=>)		
NOUN;	Declare it a noun		
NEUTER;	in neuter gender		
O1=(1->','-1);	Its stem is the part until the comma minus one character	sel	
SINGULAR;	Its singuar forms are as follows:		
NOM=O1+o;	Nominative is the stem plus ‘o’	selo	selo,NSN[...] /village n; [...]
GEN=O1+a;	Genitive is the stem plus ‘a’	sela	sela,GSN[...] /village n; [...]
DAT=O1+u;	...	selu	selu,DSN[...] /village n; [...]
[...]			
PLURAL;			
NOM=O1+a;		sela	sela,NPN[...] /village n; [...]
[...]			

Table 2

It is obvious that morphological rules enable arbitrary definition of the stem at any point, as it can be seen from the following Serbo-Croatian rule:

```
SCR PARA *ati,*em,* iv; =>
 VERB;
 TEMPLATE=INFINITIVE,1ST SG PRESENT,3RD ST
PRESENT_iv(IMPERFECTIVE);
 EXAMPLE=orati,rem,ru iv;
 ACTIVE;
  O1=(1->',');
 INFINITIVE=O1;
 PRESENT;
 AFFIRMATIVE;
  O1=(1->SAMEAS(1','+1))+(1','->2','-1);
  O2=(1->SAMEAS(2','+1))+(2','->1' ');
   SINGULAR;
    FIRST=O1+m;
    SECOND=O1+š;
    THIRD=O1;
   PLURAL;
    FIRST=O1+mo;
    SECOND=O1+te;
    THIRD=O2;
```

3. Minimizing Strategies within MIG Morphology

Every rule used in the generator consists of a head (stating the input of the rule) and a body (providing details of how the output is calculated) just like in any other MIG rule.

Labels in the knowledge base consist of endings of the words starting from the point which allows generation of the paradigm using minimal length of the corresponding rule. For example, in the Serbo-Croatian knowledge base the verb *slati,šaljem,šalju iv;* ('send') has a much longer tag than *orati,rem,ru iv;* ('plough') whereby both follow under the **ati,*em,* iv;* generator rule head.

The morphological generator thus does not follow morphological segmentation, but simply looks for the simplest solution. Furthermore, the system has a specific treatment of alternations, using the same minimal-effort approach. Finally, the system has the option of semi-automatic tagging, which is applied to the lexeme as a whole. I will discuss these three strategies in turn.

The fact that morphological segmentation is irrelevant can be exemplified using the Serbo-Croatian entires like *pile* ‘chick’, *tele* ‘calf’, etc. which have the following status in traditional morphology:

Case	**Segmentation**	**Explanation**
Nominative Singular:	**pil**-e	short stem
Genitive Singular:	**pilet**-a	long stem
Nominative Plural	**pilad**-0/**pilić**-i	supletive stems
Genitive Plural	**pilad**-i/**pilić**-a	supletive stems

MIG however follows the simplest solution, and thus defines the stem for this noun paradigm only once, using the following rule:

```
SCR PARA *e,eta n =>
 NOUN;
 TEMPLATE=NOM SG,GEN SG_N(EUTER);
 EXAMPLE=pile,eta n;
   NEUTER;
  O1=(1->',')-1);
   SINGULAR;
    NOM=O1+e;
    GEN=O1+eta;
    DAT=O1+etu;
    ACC=O1+e;
```

```
     VOC=O1+e;
     INS=O1+etom;
     LOC=O1+etu;
    PLURAL;
     NOM=O1+ad|O1+ići;
     GEN=O1+adi|O1+ića;
     DAT=O1+adima|O1+ićima;
     ACC=O1+ad|O1+iće;
     VOC=O1+adi|O1+ići;
     INS=O1+adima|O1+ićima;
     LOC=O1+adima|O1+ićima
```

The segmentation in MIG exibits the following differences in relation to the traditional one.

Traditional	**MIG**
pil-e	pil-e
pilet-a	pil-eta
pilad-0/pilić-i	pil-ad/pil-ići
pilad-i/pilić-a	pil-adi/pil-ića

This example clearly demonstrates that adopting morphological boundries different than traditional leads into a shorter description, which in turn minimizes the resources needed to create the generator.

Another minimization strategy is the combination of the constructor (presented in the Table 3) and the three mutators (presented in the Table 4) to generate the inflection for a whole range of Polish feminine nouns while at the same time accounting for a broad range of both stem and ending alternations. This combination of rules is sufficient for such diverse examples as *teczka* - GPl *teczek* (‘portfolio’), *noga* - GPl *nóg* (‘leg’);

kobieta - GSg *kobiety* (‘woman’) and *apteka -GPl aptek* (‘pharmacy’). The labels contained in dictionary entries require only basic information which cannot be inferred from the form of the lexeme. All other information is inferred from the form of the entry and dealt with by coordinating constructors and mutators.

Constructor rule	**Explanation**
POL PARA *a,V1,V2 f =>	Head: if the entry ending like this is discovered

NOUN;FEMININE;
O1=(1->','-1);
	Body: it is a feminine noun and its stem is the part preceding the comma with the final character deleted

SINGULAR;	
NOM=O1+a;	the Nominative Singular is constructed by adding ‘a’ to the stem
GEN=O1+V1;	In the Genitive Singular, the vowel after the first comma has been added to the stem
DAT=PAL(O1)+e;	In the Dative Singular, the mutator called PAL has been used
ACC=O1+ę;	
INS=O1+ą;	
LOC=PAL(O1)+e;	In the Locative Singular, the mutator called PAL has been used
VOC=O1+o;	
PLURAL;	
NOM=O1+V2;	In the Nominative Plural, the vowel following the second comma has been added to the stem
GEN=OU(KEK(O1));	In the Genitive Plural both, mutators OU and KEK have been applied to the stem
DAT=O1+om;	
ACC=O1+V2;	
INS=O1+ami;	
LOC=O1+ach;	
VOC=O1+V2	

Table 3

Mutator rule	**Explanation**
POL FUN PAL => PAL[O]=LAST[O][(t,d,r,sz,ż,rz,k,g,ch,ł,p,b,w,m,n,s,z,c)=>(ci,dzi,rz,si,zi,zi,c,dz,sz,l,pi,bi,wi,mi,ni,si,zi,ci)]	Function PAL. If the last character of the stem is one of the characters before the => sign, then the function changes it into the one after that sign. Otherwise, nothing happens
POL FUN OU => OU[O]=[O][(*KoK_)=>(*KóK_)]	Function OU. If the stem ends in a sequence: consonant-’o’-consonant, then this ‘o’ has been changed into ó
POL FUN KEK =>KEK[O]=[O][(*KK_)=>(*KeK_)]	Function KEK. If the stem ends in a sequence of two vowels, then ‘e’ has been inserted in between these two consonants

Table 4

The basic operating principle is that if the conditions are met for a mutator to be applied, then it changes the stem. If no such conditions are present, nothing happens. If we look at this rule applied to the Genitive Plural, we can see that (in the case of the entry *teczka,i,i, f*) the constructor generates the stem *teczk* -- in the Genitive plural, there are two consonants at the end of the stem and the mutator KEK inserts ‘e’ between them thereby providing *teczek* . The entry *noga,i,i f;* does not fulfill this criterion so there is no similar ‘e’ insertion. But the conditions to change ‘o’ into ‘ó’ are present so the mutator OU has been applied and the final Genitive Plural form becomes *nóg*. Finally, the entry *apteka,i,i f* does not fulfill either of the criteria so the stem remains unchanged and the Genitive Plural becomes *aptek*.

Rule	**Examples**				
POL PARA *a,V1,V2 f =>	kobieta	apteka	teczka	noga	koza

NOUN;FEMININE;
O1=(1->',''-1);
	kobiet	aptek	teczk	nog	koz

SINGULAR;					
NOM=O1+a;					
GEN=O1+V1;					
DAT=PAL(O1)+e;	kobieci+e	aptec+e	teczc+e	nodz+e	kozi+e
ACC=O1+ę;					
INS=O1+ą;					
LOC=PAL(O1)+e;	kobieci+e	aptec+e	teczc+e	nodz+e	kozi+e
VOC=O1+o;					
PLURAL;					
NOM=O1+V2;					
GEN=OU(KEK(O1));	kobiet	aptek	teczek	nóg	kóz
DAT=O1+om;					
ACC=O1+V2;					
INS=O1+ami;					
LOC=O1+ach;					
VOC=O1+V2					

Table 5

Finally the third minimization strategy used by the system consists of using one part of dictionary entries to tag the others semi-automatically. The idea is to look for any number of characters at the end of an entry to find the shortest string which allows unambiguous morphological tag assignment. It is important to stress that longer strings are applied prior to their shorter counterparts, as we can see from the following Serbo-Croatian:

```
SCR LABEL-ORDER =>
     [...]
     šov/,a m;
     lov/,a m;
     ov/,a,o--;
     tiv/,a m;
     hiv/,a m;
     iv/,a,o--;
     [...]
```

and Polish example:

```
POL LABEL-ORDER =>
     [...]
     ędzia/,iego,iowie ## m;
     zia/,zi,zie f;
     baja/,i fsg;
     aja/,i,e f;
     eacja/,i,e f;
     dacja/,i,e f;
     ykcja/,i,e f;
     tycja/,i,e f;
     cja/,i fsg;
     [...]
```

As it can be seen, this is applied to the lexeme as a whole, without taking into account any morphological boundries.

4. Conclusions

It has been demonstrated that in this system traditional morphological boundries have been used when they are needed and re-created when they are found not to be productive. This was the only possible way to achieve very practical goals of minimizing resources needed to create the system. At the same time, the goal of any scholarly research should be to find the shortest possible and the most accurate account of any phenomenon, which in turn means that morphological description should be short and accurate, be it seamless or not.

REFERENCES

Fiske S.T., Taylor S.E. 1991 *Social cognition*, New York: McGraw-Hill

Kahneman D., Slovic P., Tversky A. 1982 *Judgment under uncertainty: Heuristics and biases*, New York: Cambridge Univesity Press.

ABSTRACT

Minimal Information Grammar is a set of rules used by the MT system called NeuroTran. The grammar is named minimal for the fact that it tends to reduce the information needed for the software to perform its functions. It does so by balancing information in grammatical rules and knowledge base, both being components of the grammar, by using different classes of the rules (constructors, mutators, selectors, etc.) to manipulate the existing linguistic material, and finally by using neural networks to add new pieces of information form a learning process which is initiated any time when the software reads a text or communicates with users.

Morphological paradigms within MIG are basic information used by neural network to initiate and investigate all possible solutions in translation, parsing, and qualitative analysis. In order to generate a paradigm MIG uses labels attached to the lexical entries in main dictionary text, and then applies a set of primarily constructor-type rules to the word bearing the label.

Labels in the knowledge base consist of endings of the words starting from the point which allows generation of the paradigm using minimal length of the corresponding rule. The morphological generator thus does not follow morphological segmentation, but simply looks for the simplest solution. Furthermore, the system has a specific treatment of alternation, using the same minimal-effort approach. Every rule used in the generator consists of a head (stating the input of the rule) and a body (providing details of how the output is calculated).

The paper uses the examples from Slavic languages to show the advantages of this approach over traditional morphological segmentation.

Bio-Bibliographical Note

Danko Šipka has earned a Ph.D. in linguistics in 1989 and a Ph.D. in psychology in 1998. He was a Fulbright visiting scholar and a Humboldt post-doctoral fellow. He has published four books and more than hundred papers mostly on Slavic lexicology, lexicography, and natural language processing. At present he is an associate professor at the University of Poznan, Poland and a senior linguist at MRM, USA.