MAITE MELERO *
ARIADNA FONT LLITJOS **
Microsoft Research, Redmond *
Language Technologies Institute, Carnegie Mellon University, Pittsburgh **
United States of America
E-Mail: maitem@microsoft.com
E-Mail: aria@cs.cmu.edu

Construction of a Spanish Generation Module in the framework of a general-purpose, Multilingual Natural Language Processing System

Abstract

Our aim is to present a Generation Grammar for Spanish in development. This grammar is part of a multilingual, general-purpose Natural Language Processing (NLP) System developed at Microsoft Research and is intended to be used in a future English-Spanish Machine Translation (MT) application. It is implemented in G, a programming language close to C, well suited for the description and manipulation of natural language.

The generation module takes as input a deep syntactic, head-argument structure called Logical Form (LF) and outputs a syntactic Immediate Dependency-Linear Precedence (ID/LP) tree, identical to the tree obtained by the analysis grammar. The surface sentence is then automatically produced from this tree.

In the design of the generation module, we have to take into account the sequence of the rules and the availability of the information at every point. Also, since generation is the last step of the translation process, it may be able to provide robustness to the whole system by dealing with incomplete or inconsistent structures.

1. A Large Scale, Broad-coverage, Multilingual MTS

At Microsoft Research a large scale, broad-coverage, multilingual[1] NLP System is currently being developed. The platform used for the development of the different linguistic applications is NLPWIN, a linguist friendly programming environment with powerful debugging tools and a system of nightly build runs and verification tests.

The Machine Translation system, called WindowsMT, may be considered a hybrid of linguistic and statistical knowledge, and thus it benefits from both approaches. On one hand, the analysis and generation modules for each language are manually encoded by grammarians using the G language[2]. On the other hand, the transfer technique, called MindMeld, is done in an automatic way through alignment of source and target language.

A nice feature of this system is how it deals with ambiguity. As far as lexical ambiguity is concerned, it packs all the senses and subcategorization patterns and even different syntactic categories of a given word into a single record, which will only be disambiguated later if needed. This saves from having to deal with multiple syntactic trees due to lexical ambiguity. As for syntactic ambiguity, all the trees (and nodes) have a probability attached to them, and so the system will only output the parsed tree with the highest probability, and that is the tree used to create the corresponding Logical Form or LF.

The LF is a deep syntactic representation shared across all languages that aims at neutralizing surface discrepancies. LFs are the ultimate output of the Analysis module, what gets transferred from source to target language in the alignment process, and the input to the Generation module.

One of the highlights of this MT approach is the use of MindNet and MinMeld to provide deep example-based processing. MindNet is a lexical knowledge database automatically derived from processing text. In the transfer process, called MindMeld, the alignments are between source LF and target LF, not between source sentence and target sentence like Translation Memories. The alignment of rich LF structures enables a much more precise alignment and matching as well as a unified treatment of lexical and syntactic translation.

At runtime, WindowsMT produces an LF for the input source sentence, uses MindMeld to obtain best matches to the LFs in MindNet, and stitches together linked target LF pieces. Finally, it sends stitched target LFs to the target Generation module, first looking up in the bilingual dictionary all the nodes that haven't been matched against any source LF node.The whole MT system relies then on the felicity of

[1]It includes 7 languages so far: Japanese, Chinese, Korean, French, German, Spanish and English.
[2]G is primarily based on C, with some lisp-like list operators.

the alignments and on the result of MindMelding, which are mostly done automatically, and may yield defective LFs. For this reason, the Generation module should be capable of handling inconsistent or incomplete LFs, whenever possible, adding to the total robustness of the system.

2. The Generation Process

The input to the generation process are Spanish LF structures and the output are syntactic ID/LP trees that in most cases reproduce the tree that would be obtained by the analysis grammar. The terminal or lexical nodes of this generated tree contain a lemma and enough morphological information for the morphological rules to generate the actual words. Since all word order information is implied in the ID/LP tree, the surface sentence may be automatically produced from it.

Generation takes place in a two-step process that we will illustrate with an example.

(i) *No me gusta que llueva en verano.*

```
gustar1 ({Verb} +Pers3 +Sing +Pres +Neg +Indicat)
   |_Tsub----llover1 ({Verb} +Pers3 +Sing +Pres
                    +Subjunc +I0  +Wthr)
               |_en------verano1 ({Noun} +Masc
                            +Pers3 +Sing +Count)
   |_Tind----yo1 ({Pron} +Fem +Masc +Reflex +Pers1
                    +Sing +FindRef +Anim +Humn)
```

fig 1

In a first step, the LF structure (fig 1) is projected into a **basic syntactic tree** that keeps the argument relations present in the LF structure (fig 2). The semantic head is the first child of the parent node and the rest of the arguments follow as post-modifiers of the head[3]. An intermediate level of nodes is created as parent of each terminal node, except for the head. All generation rules will then apply to this structure on a top-down, depth-first manner, removing, creating or moving nodes as needed.

```
REC607   REC608*  "gustar"
         REC609   REC610*  "llover"
                  REC611   REC612*  "verano"
         REC613   REC614*  "yo"
```

fig 2

3. The Spanish Generation Module

The rules in the Spanish generation module apply sequentially and are distributed in three main blocks.

- There is a first block of **pre-generation rules** that operate on the semantic object (LF structure), adding or removing features and Semantic nodes, even before the basic syntactic tree is actually build. These rules have two main goals:

  - prepare the LF in order to facilitate the construction of the syntactic tree[4].
  - fix possibly deficient or inconsistent LFs

- The **generation rules** proper constitute the second block of rules. They modify the basic syntactic tree described above and perform all the linguistic manipulations needed to generate a Spanish sentence. The main generation rules are described in point 3.

- The last block is a set of **post-generation rules** that are in charge of checking the wellformedness of the generated output. They take care of:

  - some changes in the surface word order, like moving some modifiers to the front, if certain conditions are met[5];
  - several agreement checks: verb and subject, noun and its modifiers, clitics and full complements, etc.

[3]For languages like Japanese, where the verb tends to go at the end, the arguments are placed in a pre-modifier position.

[4]As, for instance, in copulative LFs where the attribute is the semantic head of the construction, the verb is restored as the head.

[5]Most of the ordering decisions are taken in the generation module itself.

- certain euphonic changes like having a masculine article in front of feminine nouns beginning with a stressed ***a***[6], or using the apocopated form of the adjective when it precedes the noun[7].

4. An overview of some Generation Rules

These rules constitute the body of the Spanish generation and they apply on the basic syntactic tree in the order in which they are presented here.

Since the generated syntactic tree is built sequentially and it acquires information in an incremental way, it is crucial to take into account

- the order in which the rules apply
- the order in which the nodes are consolidated[8]

As noted above, the rules apply to the tree top-down depth-first so that they explore a branch completely before starting with a new one. In the example in fig 2, the numbering of the records (REC607 to REC614) shows the order of the nodes on which the rules will apply.

This has many implications that will affect the design of our module; e.g. the subject is not yet consolidated when we are trying to figure out which values for person and number should we give to the verb.

There are two rules that determine the type of the node: one that builds phrasal (or non-terminal) nodes[9] and applies first and a second one that consolidates head (or terminal) nodes, and basically percolates all the information present in the parent node. The first one is more critical because all decisions on node type or what information we put in it are taken there.

Determination of node type is performed on the basis of different types of information, mostly semantic: what type of argument the node is (e.g. Attribute for Adjectives); what kind of arguments it has (e.g. Dobj (or deep object) for verbs). The rule defaults to NP when no conditions for the rest of node types are met.

Three types of information are used to compute the generated tree:

1) semantic or deep-syntax information coming from the Logical Form structure
2) lexical information encoded in the Spanish dictionary
3) syntactic information already computed by the generation rules

The use of one or other will depend on its availability at a given point in the construction of the generated tree. If the LF that results from alignment is complete and consistent then this is the kind of information we are going to use first. If it is not, then we will need to figure out how to compute it.

For example, to get the subcategorization information of the verb, instead of looking it up at the dictionary, which would give us all subcategorization possibilities of the different senses of the entry, we first look at what is in the LF. Hopefully, what it is there is the disambiguated subcategorization information that corresponds to the structure that is being generated. If for whatever reasons (problems in alignment, etc.), the information is not there, or it is contradictory, then it is retrieved from the dictionary.

Morphological information, which is required to generate the actual words, is also taken from the LF, if present; if it is not, defaults are applied for the different features (as: 3rd person, singular, masculine or present tense). However the morphological bits are subject to change in the process of application of the rest of the rules, as for instance when the agreement with the subject is checked.

In the node type rule we are already able to set the agreement values for the adjective, which are taken from the parent (NP or VP)[10]. We are also able to take some reordering decisions. Thus, if the adjective is marked as being pre-nominal in LF[11], it is put in front of the noun.

Clitics are a special kind of NPs that use Syntactic Function information to compute Case. SFs are calculated on the sentence node by a different generation rule. SF is also used to generate preposition “a” marking of Indirect Objects and human Direct Objects.

[6]el agua, un águila

[7]una gran ocasión.

[8]We will use the term “consolidate a node” when all generation rules have applied to this node. In particular, “consolidate a terminal node” will mean add all “bits” or features needed by the morphology rules.

[9]SENT, VP, NP, AJP, etc...

[10]Although, as a measure of robustness, the agreement is rechecked in the last block of rules.

[11]This information has to come from the LF and not from the dictionary because adjectives like ”único” may be both pre- and post-nominal, but the two positions are not interchangeable. Each has a different sense (with a different translation in English : *single* and *unique*).

Syntactic Functions are primarily assigned on the basis of the Semantic Roles present in LF, although the information about the category of the head sometimes has to be used in order to disambiguate between different SFs that share the same Semantic Role.

We are not going to go into details as to how coordination is represented in LF, but in the basic tree, it has a flat representation, as shown in fig 3.

(ii) *A Beatriz y a Marta les gusta el cine.*

```
REC549   REC550*  "gustar"
          REC551   REC552*  "cine"
          REC553   REC554*  "Beatriz"
          REC555   REC556*  "Marta"
```

fig 3

In a first step, syntactic functions are assigned to all coordinates independently. After that, the coordination rule applies. In the case of non-top coordination (i.e. when the coordinates are arguments, as in (ii)), it creates a parent node with the conjunction as head. See in fig 4, how the node NP16 has been created on top of REC553 and REC551.

```
DECL4   REC550*  "gustar"
          NP16     REC553   REC554*  "Beatriz"
                   CONJ4*   "y"
                   REC555   REC556*  "Marta"
          REC551   REC552*  "cine"
```

fig 4

In the case of top coordination (i.e. coordination of main verbs, see fig 5), the rule will need to create a parent for the first coordinate.

(iii) *Tome papel y escriba.*

```
REC461   REC462*  "tomar"
               REC463   REC464*  "usted"
          REC465   REC466*  "papel"
          REC467   REC468*  "escriba"
```

fig 5

In either case the rule gathers all coordinated elements (separates them with commas if they are three or more) and puts them under the top coordinated node. The values for person, gender and number for the new top NP nodes are also computed here.

There are three rules that build complex verbal groups: one for canonic passives of the form: *ser* + past participle[12]; one for progressive forms with *estar* + gerundive; and one for perfective forms with *haber* + sing-masc participle. These rules have to apply in this order to effectively deal with the combination of the three phenomena (reverse of their surface order: *ha estado siendo visto*)

Modal verbs and negation disappear from the argument structure in LF and are turned into an attribute of the main predicate. They need to be recreated in the generation module.

There is a rule for each of the following SFs: Subject, Object and Indirect Object.

The Subject rule reorders the subject (places it in front or after the verb) depending on the type of the verb. When the subject is a personal pronoun and it is not on focus, it is dropped.

The rules for the Object and the Indirect Object reorder them, taking into account whether they are clitics or full complements. They also generate clitic doubling in the case of topicalization as in: *A Juan lo vi ayer*. The clitic shares the agreement values with the full complement but this cannot be checked until the post-generation rules.

There is a rule for each type of subordinated clause: modifier subclauses, complement clauses, infinitival clauses and relative clauses.

[12]The generation of a canonical or a reflexive passive depends entirely of the LF that results from alignment, and thus it is a decision that generation doesn't need to make.

The rule that generates a Complement Clause applies on VP nodes that are arguments of the Parent (e.g. Quiero *que venga*). One of two conditions has to be true for the rule to apply: either the VP has some tense value in the LF, or the main verb subcategorizes for a that-clause.

This rule also deals with obviation: if the verb is a control verb and the subject of the subordinated clause coincides with the main subject, the VP is marked +Inf, and nothing else is done with it, in this rule. This is to avoid generating something like: *Ella quiere que ella venga.*

The Complement Clause rule performs also the following operations:

-decides whether to generates a *que* or a *si* complementizer depending on the type of the main verb.

-changes mood into subjunctive if it is a subject or the main verb takes subjunctive as complement.

-drops the pronominal subject of the clause except in potentially ambiguous cases: *María quiere que él vaya.*

-reorders the COMPCL in the main clause in relation with the verb and the other arguments.

The rule that generates an Infinitival Clause applies on VP nodes that are marked Inf in the LF or that have been marked Inf by the previous application of the Complement Clause rule, in the case of obviation. It may also apply on non-marked VPs that are objects to verbs subcategorizing for an infinitival complement.

The same rule that generates a Relative Clause on VP nodes that are attributes to non-verbal nodes, may alternatively generate a Participial Clause if the VP is not tensed (*La chica que fue seleccionada* vs*. la chica seleccionada*).

5. References

[Jensen et al. 1993] Karen Jensen, George E. Heidorn and Stephen D. Richardson (editors). Natural Language Processing: The PLNLP Approach. 1993. Kluwer, USA.

[Richardson 2000] Steve Richardson. The evolution of an NLP System. NLP Group Microsoft Research. June 2 LREC'2000 Athens, Greece.