Teaching MT with XEpisteme

J. Gabriel Amores
Departamento de Lengua Inglesa
Universidad de Sevilla
Palos de la Frontera sn
41004 Sevilla
Spain
jgabriel@us.es

Abstract

This paper describes XEpisteme, a tool for the development of transfer–based Machine Translation systems. The tool is inspired in Lexical Functional Grammar (LFG), and is currently being used to teach MT at the Universities of Seville and Pompeu Fabra in Spain. The paper first outlines the teaching context and academic background in which the tool is being utilized. After a short description of the tool, the paper focusses on why this tool is especially adequate for the purpose of teaching classical, transfer–based MT.

1 Introduction

Our experience in teaching MT started some ten years ago, in parallel with research in transfer–based systems inspired in the linguistic framework of Lexical Functional Grammar (LFG) (Kaplan and Bresnan, 1982). Several courses on MT have been taught along these years to postgraduate students, as part of the PhD programme in English Language and Linguistics at the University of Seville. Students taking this course have sufficient background in general linguistics, morphology, generative syntax, and may have taken some courses in English–Spanish translation. Over the years, students have gradually arrived with more knowledge of computers in general, while their programming skills are still rudimentary or inexistent.

A number of programs have been used to illustrate (English to Spanish) MT systems, ranging from MicroCAT to (several versions) of Engspan's PAHO system (León, 2001), and earlier versions of Episteme (Quesada Moreno and de Amores Carredano, 2000). Although translation memories are also described in our courses, we have focussed on the linguistic aspects of transfer–based MT systems. The reason for this is that our PhD programme is designed for English linguistics students, rather than potential translation professionals, for whom a broader exposure to all aspects of the translation process would certainly be necessary.

2 Overview of XEpisteme

Episteme is a tool for the development of transfer–based MT systems. As such, it allows the user to specify source and target grammars and lexicons, and a bilingual module of lexical and structural transfer rules between any pair of languages. It has been developed entirely by the Julietta research group in NLP at the University of Seville. The core system is implemented in Ansi C, running under Linux. We have recently started to develop a first version of a GUI implemented in Java. Full descriptions of the lexical analysis, parsing and unification algorithms may be found in Quesada Moreno and de Amores Carredano (2000). The system is freely available for research purposes on an *as–is* basis upon request to the authors.

3 Teaching Advantages of XEpisteme

This section outlines some of the advantages which XEpisteme offers in a teaching scenario.

3.1 Linguistic Motivation: LFG

As pointed out above, the system is inspired in a well–known linguistic framework such as LFG. This poses the advantage that students may benefit from research carried out in LFG for a number of languages, and focus on those linguistic aspects which may be of particular interest from a translation or linguistic point of view.

The use of LFG as a suitable formalism for MT is not new. A number of previous MT prototypes have demonstrated the suitability of LFG for MT to a greater or lesser extent. See for example Kaplan et al. (1989), Kudo and Nomura (1986), Netter and Wedekind (1986), Sadler et al. (1990), Way (2001) among others. As far as I know, LFG is the only unification grammar formalism which has found its way in the development of a multilingual, transfer–based commercial MT system (Apptek, 2002).

Our approach to LFG–based MT is the classical one, in which the system generates a source c- and f–structure for each sentence. Next, the transfer module obtains a target f–structure from the source language f–structure. Finally, the generation module yields the corresponding c-structure and final string output in the target language.

Currently, we have source English and Spanish lexicons of about 70,000 base entries each, with morphological, subcategorisation and semantic codification. Our English grammar covers the core subcategorisation patterns of English verbs, adjectives and nouns. The system may handle the major features of the LFG formalism, such as *coherence*, *completeness*, *consistency*, *reentrance*, $=c$, coreference, existence restrictions, set inclusion, and lexical redundancy rules. At the time of writing, *functional uncertainty* had not been implemented yet.

An additional advantage of this tool is that it could also be used to teach LFG syntax, X–Bar syntax, or any other unification–based approach to grammar. There is no specific commitment to LFG. That is, Episteme's source module is basically equipped with a parser and a unification algorithm, so that other unification–based approaches may also be implemented.

3.2 Full Control

Perhaps the most interesting feature of XEpisteme is that the user has full control over the translation process: he may work on the analysis stage only, reach up to the (lexical or structural) transfer stage, or produce a full translation of either a text or a sentence.

Before attempting a translation, the system
checks the syntactic well–formedness of the lexical and grammar rules. Duplicated production rules and rules which have the same right–hand side are also automatically spotted by the system.

Additional transfer and generation modules may be developed in order to translate from an existing source language. Although this capability is not possible at present, the system may be easily modified so that it translates into more than one target language in one go.

3.3 Graphical Environment

The graphical interface allows the user to create and load translation environments (specific analysis, transfer and generation modules grouped together under a single project name), work on the grammar and/or dictionaries, and activate or deactivate translation options (trace, statistics, full parse/partial parse, graphical output, text input format, etc.) Figure 1 below shows an example of the c- and f-structures gener-

ated after translating a simple sentence from English into Spanish.

In addition, the interface lets the user make use of up to four panels in which he may distribute the different working modules. Thus, panel one may have control over the translation process: loading an enviroment, submitting a text for translation, edit the output, etc. Panel 2 may display the source grammar and lexicon. Panel 3 may do the same for the target language, and panel 4 may show the transfer rules.

A help button connects the user to a Web page which provides on–line descriptions of the functionality of the system.

3.4 Reusability

Another important advantage of XEpisteme is that it allows the user to *plug–and–play* different components. Thus, a single lexicon may be used with different grammars if we wish to develop a controlled language version of our general system.

The hierarchical organisation of the lexicon makes it possible to use the system in noun translation mode only for example, since the different lexical categories are stored in different files.

3.5 Does not require Programming Expertise

The system is transparent to the user in the sense that no programming skills are necessary. New language modules may be developed from scratch, including morphological rules and lexical redundancy rules. Grammar rules and lexical entries may be updated, created or modified without any programming knowledge. Nevertheless, the usual difficulty of having to learn the appropriate specification language is unavoidable.

3.6 Format preservation

So far, the system accepts Ascii and HTML files, whose format is preserved after translation. In the near feature we plan to expand this functionality for other existing formats such as .RTF and .XML.

4 Sample Modules

This section provides a brief description of the specification languages in which the lexicons, grammars and transfer rules are introduced in the system. Its purpose is to show that the syntax we have developed is simple and powerful at the same time.

4.1 Analysis Lexicon

The analysis lexicon is organised as a hierarchy of classes for which the user defines the relevant features. The definition below specifies the class of verbs in English, and the features that a verbal entry may have. This definition is declared just once for each syntactic category.

```
Everb (CAT:v,LU,MOR,pred,ggf,
       vtype,aktion,@Eagr,inf,
       gprep,pform,pforms,pass,
       tns_asp:(tense,perf,prog),
       subj:(role,form,semfeat,
             agr,head,arb,acc),
       obj:(role,form,semfeat,
            @Eagr),
       obj2:(role),
       pobj:(role,pcase,semfeat),
       scomp:(role,stype),
       acomp:(role),
       ncomp:(role),
       vcomp:(role,inf,
              subj:(arb,acc),
              tns_asp:(prog,tense)
              )
       )
```

Each class (called a `Shape` in Episteme) has a number of morphological rules associated with it. The rule below generates all forms for the regular verb *listen* in English. It should be pointed out that our system generates all possible entries at compilation time. So, in fact, no morphological analysis is actually performed.

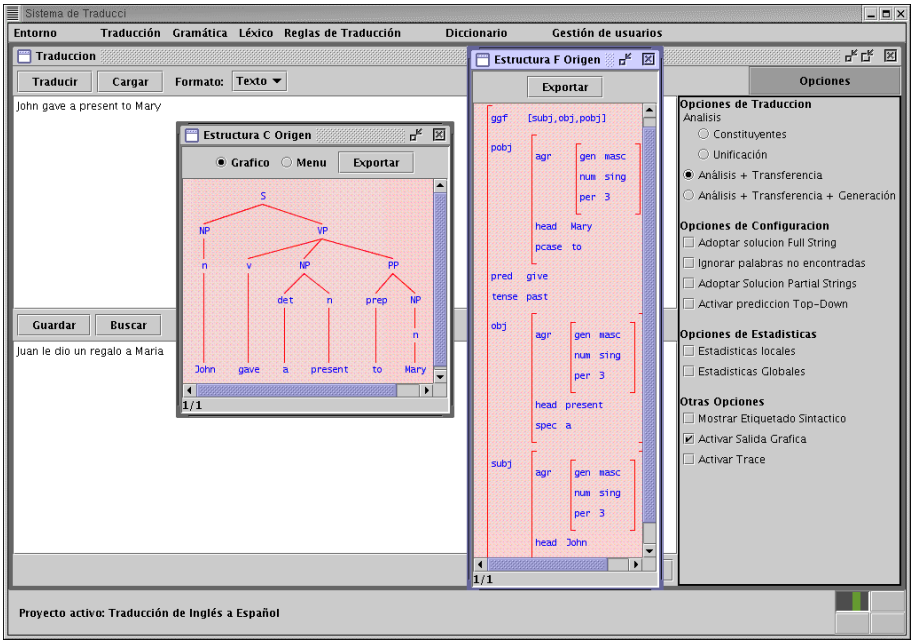

Figure 1: Graphical output for a simple sentence in XEpisteme's GUI

```
/*
* listen-ed
*/
RulePattern (MOR:[Vlisten])
RuleTarget {
(MOR:extract(base->MOR:,[Vlisten]),
           inf:bare)
(MOR:extract(base->MOR:,[Vlisten]),
        <PPl>,subj:(agr:^.agr^))
(MOR:extract(base->MOR:,[Vlisten]),
     tns_asp:(tense:pres),
     agr:(per:1|2,num:sing),
     subj:(agr:^.agr^))
(MOR:extract(base->MOR:,[Vlisten]),
     LU:strcat(base->pred:,s),
     tns_asp:(tense:pres),
     <3Sg>,subj:(agr:^.agr^))
(MOR:extract(base->MOR:,[Vlisten]),
     LU:strcat(base->pred:,ed),
     tns_asp:(tense:past),
     agr:(num:sing|plur,per:1|2|3),
     subj:(agr:^.agr^))
(MOR:extract(base->MOR:,[Vlisten]),
     LU:strcat(base->pred:,ed),
     tns_asp:(perf:yes))
(MOR:extract(base->MOR:,[Vlisten]),
     LU:strcat(base->pred:,ing),
     tns_asp:(prog:yes),
 MOR:null())  }
```

The rule described above has defined the top class for verbs in English. Subclasses of verbs may then be defined by means of `InputForms`. `InputForms` only need a subset of the features defined in the top class. In addition, subcategorisation information is specified at this point through the `ggf` feature. The rule below defines the subclass of transitive verbs in English.

```
IFverb_obj(LU,MOR)

 Everb (LU:base->LU,
        MOR:base->MOR,
        CAT:vt,
        ggf:[subj,obj],
        pred:base->LU)
```

The effect of this strategy is that entering a new transitive verb in the lexicon (for example, the verb *abandon*) comes down to adding a single line. The remaining features will be inherited from the previous `Shape` and `InputForm` defi-

nitions. The same strategy is followed in the generation lexicon.

```
(abandon,[Vlisten])
```

4.2 Analysis Grammar

Our analysis grammar rules conform to the classical LFG notation. In the rule below for transitive patterns, `@up` refers to the mother's (↑) feature structure, while `@self-N` points to the feature structures of the right–hand side constituents (↓) in the production. In addition, this rule checks for *completeness* and *coherence* locally, thus rejecting wrong parses as early as possible.

```
(8:VBAR->vt DP)
  {@up = @self-1;
   @up.obj = @self-2;
   @completeness(@sf-[subj]);
   @coherence(@sf-[subj]);  }
```

4.3 Transfer Rules

Lexical and structural transfer rules follow the classical target language selection based on conditions imposed on the input f–structure. The general syntax is the following:

```
LexicalTransfer feature
  (source-val =>
       target-val-1
       @when (condition-1)
       target-val-2
       @when (condition-2)
       ...
       target-val default)
```

As can be seen, more than one translation may be specified for each source lexical item.

4.4 Generation Grammar

The generation grammar operates recursively over the input f–structure, generating the appropriate constituents whenever possible (`@generate()` in the sample rule below), or calling the morphological generator when appropriate (`@synthesis()`). Generation rules may be triggered if specific conditions are met, similarly to what occurred with transfer rules.

```
GenerationBlock: pcase head
(7:PP -> p NP)
  {@self-1 = @synthesis(@up.pcase);
   @self-2 = @generate(@up);}
```

In conclusion to this section, it may be seen that XEpisteme poses the pedagogical advantage of stressing very relevant MT–related aspects such as modularity and the separation of linguistic data and computational algorithms, in a natural, and transparent way.

5 Future Work

In addition to keeping on expanding the linguistic coverage of existing modules, we plan to continue research in several directions:

- Implement an MS Windows version of the program. Since the system is implemented in Ansi C and Java, porting XEpisteme to the MS Windows platform should not turn too problematic.
- Relax source and target modules so that a 'direct MT' version of the system may be chosen. This is especially relevant in order to obtain some sort of translation in cases of no parsing or partial parsing.
- Specify translation for domain–specific texts.
- Link the transfer module to WordNet.
- Carry out research in the integration of transfer–based MT systems and existing translation memories.

6 Conclusion

This paper has described XEpisteme, a tool for the development of transfer–based MT systems inspired in LFG. XEpisteme is especially well suited to teach classical transfer–based MT since

it allows the user to create and load source, bilingual and target lexicons, as well as the corresponding source and target grammars for any pair of languages.

A graphical user interface shows the user the result of each stage in the translation process: analysis, transfer and generation. The strict division in modules and the organisation of the grammars and lexicons helps the student understand the importance of separating the linguistic data from the algorithms which manipulate those data.

Finally, the system is not limited to the translation of sample sentences. In fact, it is capable of translating Ascii and HTML files, in which the format will be preserved. Its implementation in standard C and Java programming languages ensures its portability to virtually any hardware platform.

References

Apptek. 2002. www.apptek.com.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical–functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, pages 173–281. The MIT Press, Cambridge, MA.

Ronald M. Kaplan, Klaus Netter, Jürgen Wedekind, and Annie Zaenen. 1989. Translation by structural correspondences. In *Proceedings of the 4th Conference of the European Chapter of the Association for Computational Linguistics*, pages 272–281, Manchester, April. UMIST.

Ikuo Kudo and Hirosato Nomura. 1986. Lexical–functional transfer: a transfer framework in a machine translation system based on lfg. In *Proceedings of the 11th International Conference on Computational Linguistics (COLING '86)*, pages 112–115, Bonn, August. University of Bonn.

Marjorie León. 2001. SPANAM©and ENGSPAN©for Windows©2000: An mt pioneer keeps up with technology. In Bente Maegaard, editor, *MT Summit VIII. Machine Translation in the Information Age*, pages 203–206, Santiago de Compostela, Spain, September.

Klaus Netter and Jürgen Wedekind. 1986. An LFG–based approach to machine translation. In *Proceedings of I International Conference on the State of the Art in Machine Translation in America, Asia and Europe (IAI-MT86)*, pages 199–209, Didweiler, Germany.

José Francisco Quesada Moreno and José Gabriel de Amores Carredano. 2000. *Diseño e Implementación de Sistemas de Traducción Automática [Design and Implementation of Machine Translation Systems]*. Secretariado de Publicaciones de la Universidad de Sevilla.

Louisa Sadler, Ian Crookston, Doug Arnold, and Andy Way. 1990. LFG and translation. In *Third International Conference on Theoretical and Methodological Issues in Machine Translation 11-13 June 1990*, pages 121–131, Linguistics Research Center, Austin, Texas.

Andy Way. 2001. Solving headswitching translation correspondences in LFG-DOT. In Miriam Butt and Tracy Holloway King, editors, *The Proceedings of the LFG '01 Conference*. CSLI Publications.