APPLICATION OF FINITE-STATE TRANSDUCERS TO THE ACQUISITION OF VERB SUBCATEGORIZATION INFORMATION

I. Aldezabal, M. Aranzabe, A. Atutxa, K. Gojenola, M. Oronoz, K. Sarasola

Department of Computer Languages and Systems
Informatika Fakultatea, 649 P. K., Euskal Herriko Unibertsitatea,
20080 Donostia (Euskal Herria)

jipgogak@si.ehu.es

Abstract

This paper presents the design and implementation of a finite-state syntactic grammar with the objective of extracting information about verb subcategorization instances from newspaper texts. After a partial parser has built basic syntactic units such as NPs, PPs, and sentential complements, a finite-state parser performs syntactic disambiguation and filtering of the results, in order to obtain a verb occurrence together with its associated syntactic components.

1 Introduction

This paper presents the design and implementation of a finite-state syntactic grammar with the objective of extracting information about verb subcategorization instances from newspaper texts, with the aim of automatically obtaining verb subcategorization frames. The system is being applied to Basque, which has as its main characteristics being agglutinative and having basically constituent-free order. After a partial parser has built basic syntactic units such as NPs, PPs, and sentential complements, a finite-state parser obtains a verb occurrence and its associated syntactic components.

Concerning the acquisition of verb subcategorization information, there are proposals ranging from manual examination of corpora [Grishman et al. 1994] to fully automatic approaches. For example, [Briscoe and Carroll 1997] describe a grammar-based experiment for the automatic extraction of subcategorization frames with their associated relative frequencies, obtaining 76.6% precision and 43.4% recall.

The rest of the paper is organized as follows. Section 2 presents the basic parsing system we have implemented, detailing its main components. Section 3 examines the finite-state-grammar applied to the extraction of subcategorization information. Finally, Section 4 presents some evaluation results.

2 Overall syntactic process

Figure 1 shows the architecture of the system. First, the system performs morphological analysis based on two-level morphology [Koskenniemi 1983; Alegria et al. 1996] and disambiguation using the Constraint Grammar formalism [Karlsson et al. 1995]. As a second step, a partial parser is applied [Abney 1997], which recognizes basic syntactic units including noun phrases, prepositional phrases and several types of subordinate sentences. Currently we can employ two different partial parsers, one of them using a unification grammar [Aldezabal et al. 2000] and the other one based on the Constraint Grammar formalism [Aduriz et al. 1997]. Although the two parsers are based on very different formalisms, they both obtain similar results, in terms of coverage and ambiguity.

After the partial parser has obtained the main syntactic components of the sentence, there are multiple readings for each sentence (see Figure 3), as a result of both morphological ambiguity (1.19 interpretations per word-form after morphological disambiguation) and syntactic ambiguities introduced by the partial parser. Although the partial parser is useful for several non-trivial applications like the detection of syntactic errors, it is still incomplete, lacking important aspects like subcategorization information. For this reason, a finite-state parser performs syntactic disambiguation and filtering of the results, with the aim of obtaining subcategorization information that will be used to enrich the lexical database.

3 A finite-state grammar for the acquisition of verb subcategorization instances

These are the main operations performed by the finite-state parser:

- Clause recognition. To extract verb subcategorization information, some rules examine the context of the target verb and define the scope of the clause to which the disambiguation operations will be applied.
- Disambiguation. Morphological and syntactic ambiguities are the cause of obtaining multiple readings per sentence, where some of them can be discarded. As whole syntactic units can be used, this disambiguation process is similar to that of Constraint Grammar (CG) disambiguation with the advantage of being able to reference syntactic units wider than the word, which must be defined in a roundabout manner in the word-based CG formalism.

  A special kind of disambiguation specific to the task of extracting verb subcategorization information is the filtering of several noninteresting syntactic tags. For example, the noun/adjective ambiguity present in *zuriekin* ('with the whites' (*zuri* as a noun) / 'with the white ones' (adjective)) can be ignored when acquiring verb subcategorization information, as we are interested in the syntactic category and the grammatical case (prepositional phrase and commitative, respectively), the same in both alternatives.

Figure 2 shows the application to a sentence, in the context of analyzing the verb *esan* (to say). The result has been simplified in the figure, because both the input and output are presented as text, rather than as an automaton containing feature-value pairs that represent syntactic components. Taking a sentence as input, the clause corresponding to the target verb is delimited first. Then, after several disambiguation steps, the syntactic units that appear with the verb are selected, each with its associated syntactic features.

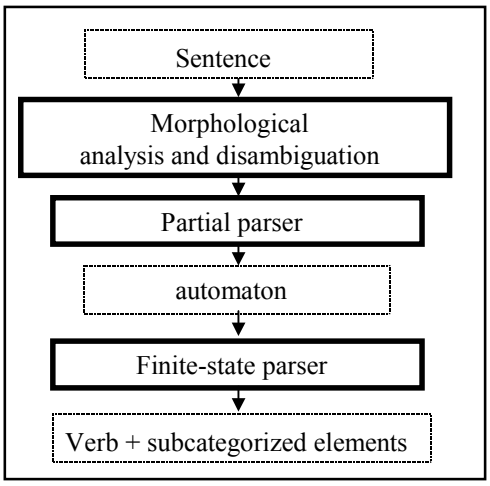

Figure 1. Overview of the system.

Bertara joandako guardia zibilak ere gauza bera esan zuen atzo eman zuten prentsaurrekoan adierazi zenez.
(The civil guard that went there also said the same thing as it was explained in the press conference they gave yesterday.)

↓

(guardia zibilak) (ere) (gauza bera) (esan zuen)
(the civil guard) (too) (the same thing) (said)
ergativeNP(*guard*, singular, definite) nominativeNP(*thing*, singular, definite) target(*say*)

Figure 2. Example of an input sentence and its corresponding output.

To implement the finite-state grammar we have applied several operations on regular expressions and relations, among them composition and replacement, using the *Xerox Finite State Tool* (XFST) [Karttunen et al. 1997]. We use both ordinary composition and the *lenient composition* operator [Karttunen 1998]. This operator allows the application of different eliminating constraints to a sentence, always with the certainty that when some constraint eliminates all the interpretations, then the constraint is not applied at all, that is, the interpretations are 'rescued', making the system robust. The operator was first proposed to formalize Optimality Theory constraints in phonology. As Karttunen points out, it also provides a flexible way to enforce linguistic or empirical constraints in syntactic disambiguation, as is done in [Oflazer 1999]. In the next subsections we will describe in detail the different parts of the finite-state grammar.

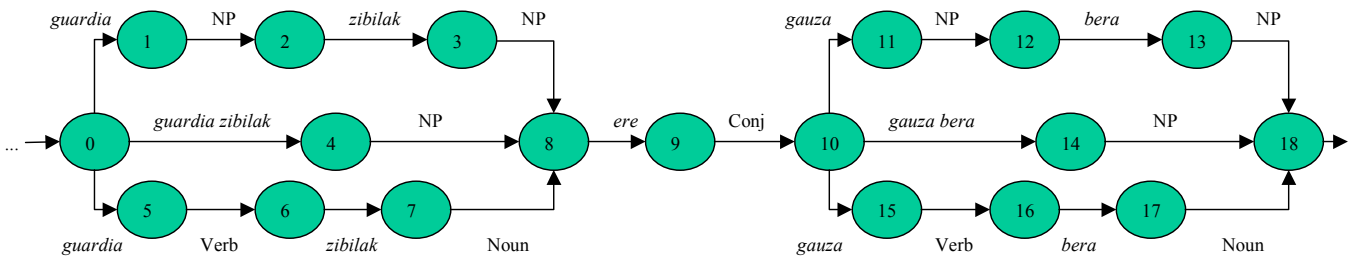

Figure 3. Automaton corresponding to a part of the sentence[1] in Figure 2: '… *guardia zibilak ere gauza bera* ….'

3.1 Clause recognition

In order to extract verb subcategorization information, a group of rules will examine the context of the target verb and define the scope of the clause associated to the target verb. The global ambiguity is considerably reduced if only the clause corresponding to the target verb is considered. In the following steps the disambiguation operations will be applied to this clause.

```
define MarkLeftSubordinate [NP | PP | NegativeParticle]*
                           TargetVerb
                            @-> "{{" ...
                            || SubordinateModifier _ ;
```

Figure 4. Example of a rule for marking sentence boundaries2.

The rule in Figure 4 will insert a clause boundary to the left of the target verb just before one or more intermediate NPs or PPs if there is a subordinate modifier to their left, which in Basque cannot be part of the clause corresponding to the target verb. For example, in the sentence of Figure 2, the delimiter "{{" will be inserted to mark the beginning of a clause ("*Bertara joandako {{ guardia zibilak ere gauza bera esan ...*"), because the verb *joandako* contains a subordinate modifier that is out of the scope of the target verb.

Twenty-two rules have been defined for the task of delimiting boundaries, which are sufficient to obtain high precision. The compilation of the previous rule produces a transducer with 482 states and 12,221 arcs. As most of the rules are as complex as the one presented in Figure 4, they cannot be composed into a single automaton, because of its prohibitive size. For that reason, the clause filters are applied sequentially.

3.2 Disambiguation: longest match selection

A usual kind of ambiguity is produced when there are alternative readings of a sentence where some of the syntactic units can be either independent or can also be included inside bigger units. Figure 5.a is a simplification of the automaton in Figure 3, which shows an example where there are several arcs giving NPs[3], while some others cannot be analyzed (they have been simplified in the picture, marked as "other"). This kind of unanalyzed element are relatively frequent and correspond to punctuation marks, unknown words or portions not covered by the partial grammar. After examining a set of example sentences, we decided to apply two heuristics [van Zanten et al. 1998]:

- Take the readings with NPs covering most of the sentence (maximal projections). Although at first glance it could seem that eliminating any "other" interpretation could suffice, this would not work, because it would eliminate all the readings, as there is one "other" path (from state 2 to 3) that must be followed by all readings. Figure 5.b shows how after applying this heuristic the number of readings is reduced from nine to four.

---

[1] The automaton has been simplified, because only the string and its syntactic category have been drawn. Actually each unit also contains its corresponding morphosyntactic tags, such as case, number, or subordination type.

[2] A rule of the form A @-> B ... C || D _ E will insert the tags B and C to either side of A in the context D A E.

[3] Although we will illustrate the problem using NPs, the method has been equally applied to other kinds of syntactic units, such as verb chains, PPs or sentential complements. In fact, we treat both NPs and PPs as NPs, because in Basque they have the same syntactic structure, differing only in syntactic case.

- Take the longest NPs (minimal number of maximal projections). The number of maximal projections is minimized in order to obtain a preference for more extended linguistic analyses over a series of smaller ones. Figure 5.b shows four readings, but the alternative containing only NP3 and NP6 will be usually preferred (Figure 5.c).

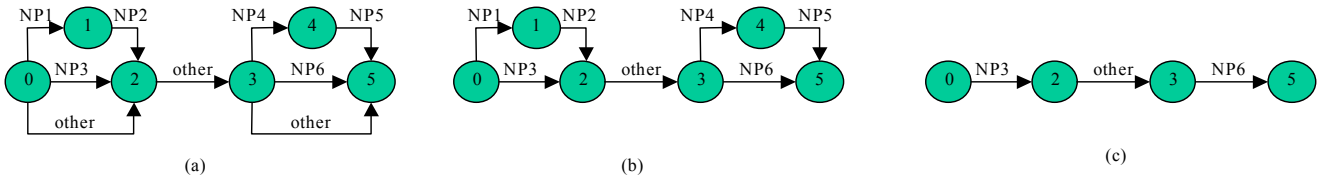

Figure 5. Application of disambiguation constraints. (a) Initial automaton. (b) Remaining alternatives after selecting the maximal projections. (c) Remaining alternative after selecting the minimal number of NPs.

The heuristics have been applied in a reductionistic manner, discarding the readings that do not meet the required constraints:

- Selection of maximal projections, that is, readings where most of the input sentence is covered by syntactic units. We applied the following constraints sequentially to obtain the desired result:

```
* Take the reading(s) that contains no "other" arcs.
* Take the reading(s) containing at most one "other" arc.
* Take the reading(s) containing at most two "other" arcs.
*  ...
* Take the reading(s) containing at most N "other" arcs[4].
```

  As the composition of the constraints with any sentence would give a null output because no reading can meet all of them, the constraints will be sequentially applied using the lenient composition operator (.O. in the *XFST* tool), so that the reading(s) with the minimum number of "other" arcs will be selected[5]:

```
define Sentence0 Sentence  .O. ~$Other
define Sentence1 Sentence0 .O. ~[[$Other]^1]
define Sentence2 Sentence1 .O. ~[[$Other]^2]
...
define SentenceN SentenceN-1 .O. ~[[$Other]^N]
```

  After applying these constraints to the automaton of Figure 5.a, the result would be that of Figure 5.b.

- To select the readings with the longest NPs we apply a similar approach, selecting the readings with the minimum number of NPs. Again, the lenient composition operator must be used. In this way, the result will be the reading in Figure 5.c.

These two sets of restrictions must be applied sequentially, because otherwise the results would not be the intended ones. For example, if we first applied the second heuristic to the automaton given in Figure 5.a, it would incorrectly take two readings (one of them would be composed by NP3 plus two "other" arcs) when the desired result should contain at least two NPs.

The heuristics for the selection of the longest match perform satisfactorily (see Section 4 for a preliminary evaluation). In a first comparison to English, one could ask if the heuristics would imply that some patterns, such as V NP PP, would always be reduced to V NP, missing the pattern NP PP. Due to the characteristics of Basque syntax this problem is not present, as any PP linked to an NP appears before it and uses the special suffixes *-ko* and *-en*, while those PPs linked to the main verb use special case markers. Obviously, this kind of problem has been avoided in part due to the specific work of extracting subcategorization information, and will have to be dealt with when developing a full syntactic analyzer.

---

[4] The value N must be a reasonable constant. With long sentences, a value of N = 20 is enough.

[5] The following operators are used: ~ (complement), $ ("contains" operator) and "X^>Y" (Y or more repetitions of X).

3.3 General disambiguation constraints

We have also developed a number of constraints that try to reduce several types of ambiguity. For example, a common case of ambiguity between *NP-Subject-Ergative-Singular* and *NP-Object-Nominative-Plural* (as in *gizonak*-the man(Subject)/the men(Object)) can be resolved using information about agreement when the verb is finite. Another ambiguous situation concerns the distinction in subordinated sentences between an indirect interrogative clause and a relative clause, which many times can be resolved due to the presence of an interrogative pronoun. These constraints must also be applied using the lenient composition operator, so that no constraint will give a null result.

```
define ApplyAgreement1 ~$[VerbNominativePlural
                           ?*
                           NPNominativeSingular
                          ];
```

Figure 6. Example of a disambiguation rule using verb agreement.

For example, the rule in Figure 6 eliminates sentence readings containing a singular NP in nominative case and a finite verb that asks for a plural nominative NP, due to agreement in number. The constraint will be applied after the correct clause has been delimited, because the presence of more than one verb would make the restriction highly inaccurate. There are sixteen constraints treating agreement, which can only resolve part of this kind of ambiguity (for example, the constraints can not be applied to sentences with nonfinite verbs).

4 Evaluation

The resulting finite-state grammar contains more than 300 rules. It has been applied to a corpus of 111,000 sentences from newspaper texts, totaling two million words. When dealing with unrestricted texts there are several extra difficulties added to the problem of ambiguity, such as multiword lexical units, unknown words, proper names, spelling errors and long sentences (as each sentence contains an instance of the target verb together with other main or subordinate clauses, delimiting the exact boundaries of the clause corresponding to the target verb is a difficult task).

For evaluation we took a set of previously unseen 150 sentences [Aldezabal et al. 2000]. We measured precision (the number of correctly selected elements / all the elements returned) and recall (the number of correctly selected elements / all the elements present in the sentence), applied to each instance of the target verb and its corresponding complements/adjuncts. Although there is always a balance between precision and recall, we tried to maximize precision, sometimes at the cost of lowering recall. We consider the results satisfactory, with 87% precision and 66% recall. We examined the causes of the errors manually, concluding that about half of them can be improved by simple refinements of the lexicon and the grammars (both the partial grammar and the finite-state grammar), while the others would require qualitative changes to the grammars, as the inclusion of information about subcategorization.

5 Conclusion

In the literature, finite-state language processing has been applied to a variety of tasks, ranging from morphology to syntax. This work presents the application of a finite-state grammar to the specific task of detecting verbs and their associated sentence components. The application works at a level higher than morphology, as previously built syntactic units must be dealt with, but only with a restricted subset of syntax: the relation of verbs with their adjuncts and complements. The task is complex because many problems must be taken into account, such as ambiguity and determination of clause boundaries, among others.

The finite-state grammar provides a modular, declarative and flexible workbench to deal with the graph of syntactic components. It establishes the application of empirical, corpus-oriented facts about morphological/syntactic ambiguity, versus the more general facts on linguistic well-formedness encoded in the previously applied partial grammar. The system is being used in the process of acquiring verb subcategorization instances, obtaining 87% precision and 66% recall over a set of previously unseen 150 sentences, and it has been applied to a corpus of two million words.

Although the finite-state grammar was specifically designed for that concrete application, many rules express general linguistic facts and are reusable, so that we plan to include them in a more general syntactic finite-state grammar, together with the subcategorization information that is being acquired.
We are currently working on future extensions of this work:

- As the finite-state grammar selects instances of environments in which each verb appears, we are at the point of refining these environments by automatically eliminating the possible adjuncts, with the objective of distinguishing complements and adjuncts. In order to perform this task, we have used the *chi-square* statistical measure, which allows deciding whether the association between each element in the environment and the verb is strong or not. In the latter case, we will consider the element an adjunct that will be deleted from the verb instance. This way, the number of putative patterns can be cleaned and reduced.

  The next step will consist in applying *Bayes* to group subcategorization instances into frames. We are right at the point of evaluating the performance of the statistical measures. Note that we do not assume the existence of any previously defined set of possible subcategorization frames. Our aim is to get them from raw data. It is important to do so because no Basque dictionary includes subcategorization information apart from the traditional transitive/intransitive distinction.

- In our finite-state grammar we have used the lenient composition operator for the selection of the longest match, following the proposal defined in [Karttunen 1998] for Optimality Theory in phonology. Recently, [Gerdemann and van Noord 2000] have demonstrated how a *matching* approach to the same problem, without using lenient composition, improves the results, as it eliminates the need for an upper limit on the counting, while at the same time minimizes the resulting transducer. The application of this new approach to syntax is a problem that deserves further research.

Acknowledgements

This research is supported by the Basque Goverment, the University of the Basque Country and the Interministerial Commision for Science and Technology (CICYT). Thanks to Gorka Elordieta for his help writing the final version of the paper.

Bibliography

[Alegria et al. 1996] Alegria I., Artola X., Sarasola K., Urkia M. 1996. Automatic morphological analysis of Basque. Literary and Linguistic Computing. 11 (4). Oxford University.

[Abney 1997] Abney S. P. 1997. Part-of-Speech Tagging and Partial Parsing. S. Young eta G. Bloothooft, editoreak, Corpus-Based Methods in Language and Speech Processing, Kluwer, Dordrecht.

[Aduriz et al. 1997] Aduriz I., Arriola J.M., Artola X., Díaz de Ilarraza A., Gojenola K., Maritxalar M. 1997. Morphosyntactic disambiguation for Basque based on the Constraint Grammar Formalism. Conference on Recent Advances in Natural Language Processing, Bulgaria.

[Aldezabal et al. 2000] Aldezabal I., Gojenola K., Sarasola K. 2000. A bootstrapping approach to parser development. Proceedings of the International Workshop on Parsing Technologies, Trento..

[Briscoe and Carroll 1997] Briscoe T., Carroll J. 1997. Automatic Extraction of Subcategorization from Corpora. ANLP'97, Washington.

[Ezeiza et al. 1998] Ezeiza N., Alegria I., Arriola J.M., Urizar R., Aduriz I., 1998. Combining Stochastic and Rule-Based Methods for Disambiguation in Agglutinative Languages. COLING-ACL 1998, Montreal.

[Gerdemann and van Noord 2000] Gerdemann D., van Noord G. 2000. Approximation and Exactness in Finite State Optimality Theory. Proceedings of the Fifth Workshop of the ACL Special Interest Group in Computational Phonology (SIGPHON), COLING 2000, Luxembourg.

[Grishman et al. 1994] Grishman R., Macleod C., Meyers A. 1994. Comlex Syntax: Building a Computational Lexicon. COLING 1994, Japan.

[Karlsson et al. 1995] Karlsson F., Voutilainen A., Heikkila J., Anttila A. 1995. Constraint Grammar: A Language-independent System for Parsing Unrestricted Text. Mouton de Gruyter.

[Karttunen et al. 1997] Karttunen L., Chanod J-P., Grefenstette G., Schiller A. 1997. Regular Expressions For Language Engineering. Natural Language Engineering.

[Karttunen 1998] Karttunen L. 1998. The Proper Treatment of Optimality in Computational Phonology. Proceedings of the International Workshop on Finite State Methods in Natural Language Processing, Ankara.

[Oflazer 1999] Oflazer K. 1999. Dependency Parsing with an Extended Finite State Approach. ACL'99, Maryland.

[Roche and Schabes 1997] Roche R., Schabes Y. 1997. Finite-State Language Processing. MIT Press.

[Samuelsson and Voutilainen 1997] Samuelsson C., Voutilainen A. 1997. Comparing a Linguistic and a Stochastic Tagger. ACL-EACL'97, Madrid.

[van Zanten et al. 1998] van Zanten G.V., Bouma G., Sima'an K., van Noord G., Bonnema R.. Evaluation of the NLP Components of the OVIS2 Spoken Dialogue System. In: van Eynde, Schuurman and Schelkens (eds), Computational Linguistics in the Netherlands 1998, Rodopi Ámsterdam.