Proceedings

of the

ESSLLI Student Session 1999

Amalia Todirascu (editor)

Combining Chart-Parsing and Finite State Parsing

I. Aldezabal, K. Gojenola, M. Oronoz
Informatika Fakultatea, 649 P. K. Euskal Herriko Unibertsitatea,
20080 Donostia (Euskal Herria)
E-mail: jipgogak@si.ehu.es

Abstract

This paper presents the development of a parsing system that combines a unification-based partial chart-parser with finite state technology. It is being applied to Basque, an agglutinative language. In a first phase, a unification grammar is applied to each sentence, giving a chart as a result. The grammar is partial and gives a good coverage of the main elements of the sentence, but the output is ambiguous. After that, the resulting chart is treated as an automaton to which finite state disambiguation constraints and filters can be applied to choose the best single analysis. The system has been tested on two different applications: the acquisition of subcategorization information for verbs, and the detection of syntactic errors.

0.1 Introduction

This paper presents a project for the development of a parsing system that combines a unification-based partial chart-parser with finite state technology. The system is being applied to Basque, an agglutinative language, with free order among sentence components.

In a first phase, a unification grammar is applied to each sentence, giving a chart as a result. The grammar is partial but gives a complete coverage of the main elements of the sentence, such as noun phrases, prepositional phrases, sentential complements and simple sentences. It can be seen as a shallow parser [2, 3] that can be used for subsequent processing. However, it contains both morphological an syntactic ambiguities, giving a huge number of different interpretations per sentence.

After that, the resulting chart is treated as an automaton to which finite state disambiguation constraints and filters can be applied, so that unwanted readings or information can be discarded or parts of a sentence can be chosen, depending on the application. The system has been tested on two different applications: the extraction of subcategorization information for a given verb, and the detection of syntactic errors.

In this manner, we combine constructive and reductionistic approaches to parsing: in a first phase the unification-grammar obtains syntactic units, usually with many different interpretations, while in a second phase the analyses are restricted according to the context and the kind of application.

The remainder of this paper is organized as follows. In Section 2 we present a description of the chart-parser. Section 3 describes the finite state parser and its combination with the chart. Section 4 specifies the applications of the system, and shows some preliminary results.

0.2 The chart parser

0.2.1 Previous work

The system relies on different wide-coverage tools that have been developed for Basque:

- The Lexical Database for Basque [4]. It is a large repository of lexical information with about 70.000 entries (60.000 lemmas), each one with its associated linguistic features: category, subcategory, case, number, definiteness, mood and tense, among others. As this database is the basis of the syntactic analyzer, we must say that there is no information related to verb subcategorization.

- A morphological analyzer [4]. The analyzer applies Two-Level Morphology for the morphological description and obtains, for each word, its segmentation(s) into component morphemes. The module is robust and has full coverage of free-running texts in Basque.

- A morphological disambiguator based on both the Constraint Grammar formalism [13, 5] and a statistical tagger [9]. This tool reduces the high word-level ambiguity from 2.65 to 1.19 interpretations, but still leaves a number of different interpretations per sentence.

0.2.2 The syntactic grammar

Basque being an agglutinative language, linguistic information like case, number and determination are given by means of morphemes appended to the last element of a noun/prepositional phrase. Following the most extended linguistic descriptions for Basque, we took morphemes (see Figure 1) as the basic units upon which syntactic analysis is based, departing from the traditional use of the grammatical word as the syntactic unit, as is done in non agglutinative languages like English. Although the figure only shows the category of each lexical/syntactic unit, there is a rich information for each of them, encoded in the form of feature structures. Moreover, after the syntactic units are obtained, the analysis tree is not used any more.

The PATR-II formalism was used for the definition of the syntactic rules. There were two main reasons for this election:

- Simplicity. The grammar is not linked to a linguistic theory, like LFG or HPSG, as there has not been a broad description for Basque using those formalisms [1]. Moreover, their application would require information not available at the moment, such as verb subcategorization. PATR-II is more flexible at the cost of extra writing, as it is defined at a lower level. As we will explain later, we plan to use this analyzer in the process of bootstrapping lexical information.

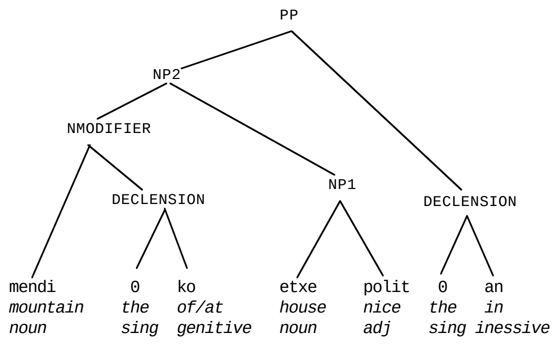

Figure 1: Parse tree for *mendiko etxe politan* ('in the nice house at the mountain')

- The formalism is based on unification. This is useful for the manipulation of complex linguistic structures for the representation of grammatical constituents. We must stress that the grammar uses good linguistic granularity, in the sense that we keep all the linguistically relevant morphosyntactic information, very rich compared to most chunking systems. This fact will enable us to use it as a general tool for different applications.

The grammar at the moment contains 120 rules. There is an average number of 15 equations per rule, some of them for testing conditions like agreement, and others for structure building. The main phenomena covered are:

- Noun phrases and prepositional phrases. Agreement among the component elements is verified, added to the proper use of determiners.
- Subordinate sentences, such as sentential complements (completive clauses, indirect questions, ...) and relative clauses.
- Simple sentences using all the previous elements. The rich agreement between the verb and the main sentence constituents (subject, object and second object) in case, number and person is verified.

The components found by the parser are very reliable, as only well-formed elements are obtained, which are necessary for a full sentence interpretation. Due to the variety of syntactic structures found in real sentences, the grammar is limited in the sense that only a small fraction of the sentences will receive a full analysis. However, constructing a complete grammar would be a costly enterprise. For that reason, this grammar is applied bottom-up, resulting in a parser that obtains pieces of analyses or chunks, and the resulting chart can be used for subsequent processing.

Charts have been used before as a source of information in [11], where the information contained in the chart is inspected when no complete parse

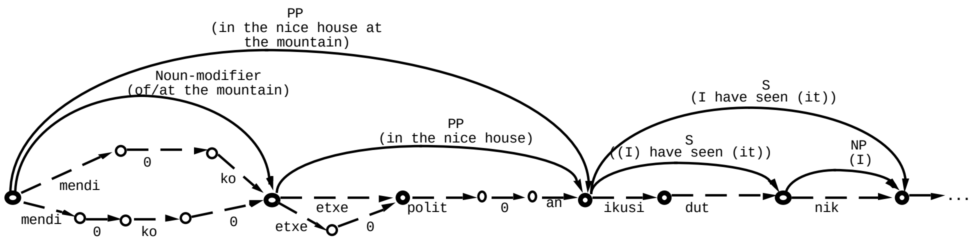

Figure 2: State of the chart after the analysis of *Mendiko etxe politan ikusi dut nik* ('I have seen (it) in the nice house at the mountain')

is found, with the aim of correcting a syntactic error. In fact, this system assumed that every correct sentence gets a complete analysis, while we consider that quite often the grammar will not be able to parse the whole sentence. Even in the case when we want to detect a grammatical error, we do not assume that the corrected sentence would give a complete parse.

0.3 The Finite State Parser

In recent years the field of finite state systems has gained much attention [12, 13]. A characteristic that all finite state systems have in common is that they work on real texts dealing with units bigger than the grammatical word, using more elaborated information, but without reaching the complexity of full sentence analysis.

As the output of our chart parser is ambiguous, we obtain a large number of potential analyses (concatenations of chunks covering the whole sentence). We needed a tool for the selection of patterns over the final chart, and finite state technology is adequate for this task. Currently we use the Xerox Finite State Tool (XFST[1]). As a link between both parsers, the chart must be converted into an automaton, which can then be processed by XFST (see Figure 2). In the figure, dashed lines are used to indicate lexical elements, while plain lines define syntactic units (those obtained by the chart-parser, see Figure 1). The bold circles represent word-boundaries, and the plain ones delimit morpheme-boundaries. As before, each arc in the figure is represented by its morphosyntactic information, in the form of a sequence of feature-value pairs.

In the different applications, we have used some common operations:

- Disambiguation. Two kinds of ambiguity were considered: morphosyntactic ambiguity left by the Constraint Grammar and stochastic disambiguation processes and that introduced by the chart parser. As

[1]http://www.rxrc.xerox.com/research/mltt/fsSoft/docs/fst-97/xfst97.html

whole syntactic units can be used, this process is similar to that of CG disambiguation with the advantage of being able to reference syntactic units larger than the word.

- Filtering. Depending on each application, sometimes not all the available information is relevant. For example, one difficult kind of ambiguity, noun/adjective, as in *zuriekin* (*zuri*='white', 'with the whites'/ 'with the white ones') can be ignored when acquiring verb subcategorization information, as we are mainly interested in the syntactic category and the grammatical case (prepositional phrase and commitative, respectively), the same in both alternatives.
- Extracting parts of a sentence. The global ambiguity of a sentence is considerably reduced if only part of it is considered. For instance, in the case of extracting verb subcategorization information, some rules examine the context of the target verb and define the scope of the subsentence to which the previous operations will be applied.

Among the finite state operators used (see Figure 3[2]), we apply composition, intersection and union of regular expressions and transducers[3]. We use both ordinary composition and the recently defined *lenient composition* [10]. This operator allows the application of different eliminating constraints to a sentence, always with the certainty that when some given constraint eliminates all the interpretations, then the constraint is not applied at all, that is, the interpretations are "rescued". As the operator is defined in terms of regular relations it can be composed with other automata.

0.4 Applications

0.4.1 Acquisition of subcategorization information

We are interested in automatically obtaining verb subcategorization information from corpora [7, 8]. Our objective is to enrich the verb entries contained in the lexical database with information about the main constituents (noun phrases, prepositional complements and subordinate sentences) appearing with each verb.

The chart parser analyzes the main units in each sentence and then finite state rules carry out the following tasks (see Figure 3):

- Recognition of the relevant subsentence. The mean number of words per sentence is 22, ranging from one to six subsentences. The problem is that of delimiting the part corresponding to the target verb.

[2]The .o. and & operators denote respectively the composition and intersection of regular languages and relations.

[3]In fact, Figure 3 represents a simplification of the system, as each operation in the figure corresponds to the composition of a considerable number of smaller rules.

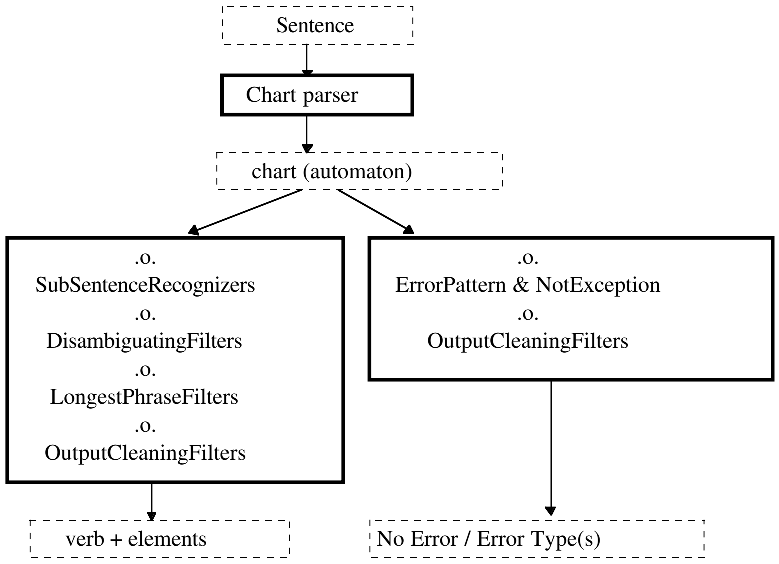

Figure 3: Different applications of the system

- Disambiguation. Some heuristics are used to solve the remaining ambiguities, related, among others, to different types of subordination.
- Selection of the longest NP/PPs. This heuristic proved to be very reliable to exclude multiple readings of a single NP/PP.
- Cleaning the output. The result of the previous phase contains a lot of morphsyntactic information, but most of it is superfluous for this application, as we are mainly interested in the grammatical case, number and subordination type of each element.

The design of the finite state rules is a non-trivial task when dealing with real texts. Nevertheless, the declarativeness of finite state tools improves significantly our previous version of the system implemented by means of ad hoc programs, difficult to correct and maintain. Moreover, XFST also improves efficiency. About 350 finite state definitions were made for this application.

As a first experiment, 500 sentences for each of 5 different verbs were selected. Figure 4 shows the cases that have mostly appeared around each of the selected verbs. When there was more than a single analysis for a subsentence, it was not taken into account. This eliminated about 5% of the sentences. This is not a problem as long as the corpus is big enough [7]. The absolutive case (the one that usually represents both the object of the transitive verbs and the subject of the intransitive ones) is not included since it is by far the most frequent case in all verbs, therefore it is not relevant

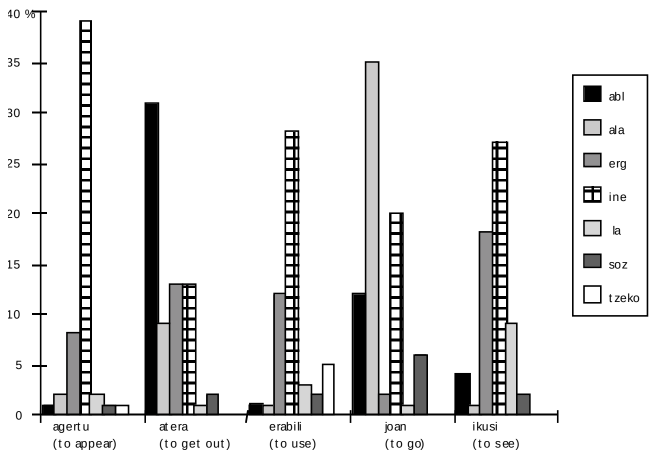

Figure 4: Frequency of elements for each verb

to characterize the different behaviour of the verbs. On the other hand, we find cases that seem to have close relationship to a given verb, such as the ablative case ('from') in the *atera* verb ('to go out'), the alative case ('to') in the *joan* verb ('to go') and the inessive case ('in') in the *agertu* verb ('to appear'). Both the ablative and alative cases usually represent respectively the source and the goal of an action, which means that both represent a movement. As for the inessive case, it represents the spatio-temporal coordinates of an entity or event. That is why it appears in all the verbs. Finally, there are cases that show an exclusive tendency toward a given verb, such as the completive subordinate *-la* ('that') in the *ikusi* verb ('to see'), and the adverbial subordinate *-tzeko* ('for', 'so that').

All these properties make clear that verbs take other relevant cases apart from the ones representing the object and subject of the sentences, and therefore it does not strike us as implausible to assume that they should be taken into account in the main structure of the verbs. In other words, the system can help us in deciding whether a given phrase can be considered as an argument for the treated verb.

Added to this experiment, we are also working on getting statistical patterns from the different combinations of the elements appearing in each sentence. We will apply the system to 8.000 sentences (about 200.000 words) corresponding to 10 new verbs. In this way, we expect to obtain patterns that will let us group all verbs in different classes. It will be also a way to verify the suggestions and similarities got from the above experiment.

Regarding evaluation, there are not already existing resources (in the form of annotated corpus or dictionaries) with subcategorization information to automatically compare with the results of the tool, as in [7, 8]. As a first test we manually evaluated the results after applying our tool to 50 sentences (with an average of 26 words per sentence). Table 1 shows the results.

Concerning the recognition of subsentences to which each verb applies, it is done correctly 82% of the times. Sometimes the subsentence is easily recognizable (for example, when delimited by punctuation marks), while other cases are difficult due to the nested structure of complex, long sentences. We also counted the number of times that the verb is correctly returned with all its relevant syntactic components (without examining their role as arguments or adjuncts). This was performed correctly in 70% of the sentences. After examining the global error rate (30%, that is, the total number of errors including the incorrectly recognized subsentences), we found room for improvement: half of the errors were due to the previous disambiguation process (either the incorrect selection was chosen or there was more than an alternative), while a third of errors were caused by the incompleteness of the unification grammar. Although these problems will always be error sources, we feel that after adding some simple and general linguistic rules they will correct most of the errors.

Subsentence recognition		**Syntactic elements**	
Correct	Incorrect	Correct	Incorrect
82%	18%	70%	30%

Table 1. Results of the system applied to 50 sentences.

0.4.2 Syntactic error detection

The system has been tested on errors in real texts written by learners of Basque. As a preliminary experiment, we chose errors related with date-expressions for different reasons:

- The context of application is wide and well-defined in Basque, as well as the most common errors. In Basque, date-expressions like 'Donostian, 1995eko maiatzaren 15ean' (Donostia, 15th of July, 1995) require that some of the elements are inflected, sometimes according to another contiguous element.

- As it is relatively easy to obtain test data we eliminate one of the main problems when dealing with syntactic errors in real texts: finding erroneous sentences for each phenomena.

In order to recognize 6 different error types, we needed more than 100 definitions of finite state patterns for their treatment. We have used a corpus consisting of 70 dates (including correct and incorrect ones) for the definition of the patterns. Apart from the six kinds of errors, we also tried to account for the most frequent combinations of errors. At the moment we are in the process of getting new corpora with erroneous sentences for testing. Although the date error recognition is complex in Basque, the tool provides again a flexible and systematic way to solve it. We plan to extend the system to other types of errors, like agreement between the main components of the sentence, which is very rich in Basque, being a source of many errors.

0.5 Conclusions

This work presents the development of a system which combines:

- A syntactic grammar for Basque. It covers the main components of the sentence. Due to the agglutinative nature of Basque, the introduction of the syntactic level greatly improves the ability to treat the wealth of information contained in each word/morpheme. The recognized components can be used for posterior processing.

- Finite state rules. They provide a modular, declarative and flexible workbench to deal with the resulting chart, making use of the available information for different tasks, such as syntactic error detection or the acquisition of subcategorization information.

This combination results very adequate for shallow parsing applications to languages like Basque, with limited grammatical resources compared to other languages. The free order of constituents, among other aspects, would make the design of a full grammar a very complex task. Moreover, the partial grammar is enough for the intended applications. The consideration of the chart as the interface between both subsystems also adds to the simplicity of the combined tool. Finally, we must emphasize two main conclusions:

- The stratified partial parsing approach provides a powerful way to consider simultaneously information at morpheme, word and phrase level, adequate for agglutinative languages where the grammatical word is not the starting point of the analysis.

- The unification grammar and the finite state system are complementary. The grammar is necessary to treat aspects like complex agreement and word order variations, currently unsolvable using finite state networks, due to the exponential growth in size of the resulting automata [6]. On the other hand, regular expressions, in the form of automata and transducers, are suitable for operations like disambiguation and filtering.

Acknowledgements

This work was partially supported by the Basque Government, the University of the Basque Country and the CICYT. Thanks to Gorka Elordieta for his help writing the final version of this document.

Bibliography

[1] J. Abaitua. Complex predicates in Basque: from lexical forms to functional structures. *PhD Thesis*, Univ. of Manchester, 1988.

[2] S. Abney. Part-of-Speech Tagging and Partial Parsing. In *Corpus-Based Methods in Language and Speech Processing*, Kluwer, Dordrecht, 1997.

[3] S. Ait-Mokhtar, J-P. Chanod. Incremental Finite-State Parsing. In *Proceedings of ANLP-97*, Washington, 1997.

[4] I. Alegria, X. Artola, K. Sarasola, M. Urkia. Automatic morphological analysis of Basque. In *Literary & Linguistic Computing, Vol. 11*, 1996.

[5] I. Alegria, J. M. Arriola, X. Artola, A. Diaz de Ilarraza, K. Gojenola, M. Maritxalar, I. Aduriz. Morphosyntactic disambiguation for Basque based on the Constraint Grammar Formalism. In *RANLP*, 1997.

[6] K. Beesley. Constraining Separate Morphotactic Dependencies in Finite-State Grammars. In *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing*, Ankara, 1998.

[7] M. Brent. From Grammar to Lexicon: Unsupervised Learning of Lexical Syntax. In *Computational Linguistics, vol. 19(2)*, 1993.

[8] T. Briscoe, J. Carroll. Automatic Extraction of Subcategorization from Corpora. In *Proceedings of ANLP-97*, Washington, 1997.

[9] N. Ezeiza, I. Alegria, J.M. Arriola, R. Urizar, I. Aduriz. Combining Stochastic and Rule-Based Methods for Disambiguation in Agglutinative Languages. In *Proceedings of COLING-ACL-98*, Montreal, 1998.

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

[11] C. Mellish. Some Chart-Based Techniques for Parsing Ill-Formed Input. In *Proceedings of EACL-89*, 1989.

[12] E. Roche, Y. Schabes. Finite-State Language Processing. MIT, 1997.

[13] A. Voutilainen. A syntax-based part-of-speech analyser. In *Proceedings of EACL-95*, Dublin, 1995.