On increasing the syntactical parse efficiency

Borbála Katalin Benkő

Department of Telecommunications
Budapest University of Technology and Economics, Hungary
bbenko@hit.bme.hu

Abstract

The efficiency of a syntactical parser can be measured according to speed and correctness factors. One reason of the inefficiencies is the border problem, which is when a constituent is mistakenly assigned to a neighboring one.

This paper introduces a formal tool that reduces the border problem, thus increasing the speed of the parse and decreasing the ratio of wrong results. The basic idea of the "strong rules" model is that natural languages contain definitely unambiguously parsable structures (humans always prefer this parse). With marking the borders of such structures, the defective expansion of the surrounding constituents can be limited.

Implementation issues and other application areas of the "strong rules" model are also discussed.

1. Introduction

Syntactical parsers are mainly used for information extraction, machine translation and in other high-level text processing systems [1]. These applications require *fast* and *unambiguous* parse results (e.g. a translator is unusable if there are 300 translations for a single sentence).

Ambiguity reduction is usually achieved via weighting and filtering the symbols: a weight value is assigned to each of them; symbols with too small weights are dropped. An example is the probabilistic approach, where weights represent probabilities [2],[3]. Although the models include algorithms for the initial value assignment [4], in practice this is one of the weakest points.

Using the weighting-filtering point of view we can handle several language phenomena but not all the problems. The strong rule model was designed to handle such an unsolved problem. Although at first sight, "strong rules" seem to be a variation of the probabilistic approach, detailed description shows that they are orthogonal.

1.1. The border problem

It can be observed that parsers spend considerable amount of time building incorrect parse trees; even if part of these wrong trees is filtered soon after creation. The motivation of the "strong rules" is to reduce the time spent on building wrong trees wherever possible.
This paper deals with one specific situation, a phenomenon that is called the "border problem". Solving the border problem means that the wrong parses are excluded, so both the result's precision increases and the process becomes faster.

The border problem is when

- There is an ambiguous constituent (e.g. word) in the sentence: it has correct and wrong parses.
- At the unification phase, the tree containing wrong parse doesn't stop at the real border of the structure; it ranges into a neighboring structure.

The border problem is crucial in some agglutinating languages with free word order. In these languages, it is easy to see the relationships inside of a structure but a full parse is required to determine the borders (e.g. the border between two NPs).

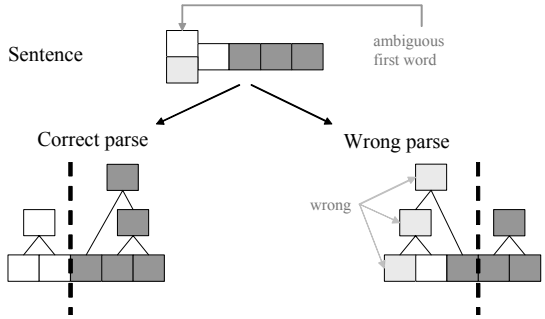

Figure 1: The border problem

Figure 1 shows an example for the border problem. The first word of the sentence has ambiguous parse (the wrong one is marked gray). The correct parse says that the first two and the last three symbols form constituents. In the alternative parse, the wrong tree expands into the second structure, also resulting in a full coverage. The problem is that not only the inner structure of a constituent is corrupted but also the border between them is moved.

1.2. Effects of the border problem

The border problem degrades both efficiency factors.

- There are more symbols in the parse table, so the parse process requires more time (speed factor).
- In several cases, the expanding wrong parse also results a full coverage. The ratio of wrong trees in the result will be higher, which degrades the correctness factor.

Although, the border problem is not the only reason of the efficiency cut, it is a considerable part. Measurements show that reducing the border problem results significant performance growth (see Section 5).

1.3. Solution possibilities

There are two main solution possibilities for the border problem.

The probabilistic approach tries to solve it with assigning a high value to the correct parse and giving only a low score

to the wrong tree. Using a sufficient cutting level, the wrong parse is discarded. The main difficulty of this approach is that the cutting level must be chosen carefully: with a level too low also the correct parses may be discarded; with a level too high also the wrong parse is kept, increasing the number of symbols in the parse table. Since the probability value assignment uses only inner information, this is a quite unsolvable task. In several cases, the real probability of the structure also depends on the neighboring symbols; which is not considered by the probabilistic approach. Furthermore, there is no generic possibility to implement mutual exclusion.

Another approach is to detect and mark the borders of the structures where possible. The definitely detected borders are marked: they cannot be crossed by the surrounding ambiguous parses. So, the spread of the adjacent ambiguity is stopped. In the classical models, there is no possibility for border marking; the "strong rules" model is based on this idea.

2. The "strong rules" idea

Adding and managing border marks in a classical parsing model would be very a complex and time-consuming task. The strong rule model marks the border of a structure so that it hides the constituents, only the unified symbol is left in the parse table. The definitely correctly parsed structures become indivisible units; so the adjacent unambiguous structures cannot partially expand into them, causing ambiguity roll-on. This general definition can be specialized different ways:

- Strong rules (definite rules): the constituents unified by a strong rule become invisible, that is, no rule can access them in the future.
- Strong symbols: instead of the rules, the symbols are marked as strong. A strong symbol hides its inner subsets of the same type (e.g. a 4-word NP hides its 2-word inner part that is also an NP)

Both models have been implemented and evaluated. The parser HumorESK of MorphoLogic Ltd. [5] uses strong symbols; SRP (Strong Rule Parser, previously Piranha) of BUTE [6] is built on the strong rule model.

Theoretically, the strong symbol model can also be derived from the strong rule model so that the recursive rules that generate the given symbol (e.g. NP) are marked strong.

Strength marks should be used with care. Appropriately used strong symbols/rules make the parse process faster and increase the correctness of the results; inappropriately used strong elements may loose correct parses or even cause the sentence to be unparsable (e.g. overlapping strong elements).

2.1. Strong symbols

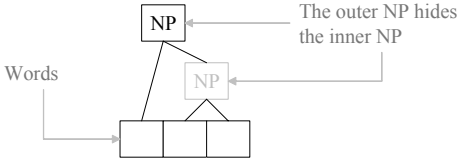

Figure 2: Strong symbol

Strong symbols are motivated by the quasi-recursive structure of Hungarian NP-s and VP-s. In Hungarian, if an adjacent symbol can be unified with an NP, the correct parse is always[1] the unified version; even if non-unification also results in full coverage.

Always-expanding symbols are marked strong, which means, that the wider symbols hides its narrower constituents, like the outer NP hides the inner NP in Figure 2.

Implementation issues

Strong symbols can be implemented so that a normal CFG execution (e.g. a CYK) is periodically interrupted by a cleanup mechanism. The cleanup checks whether newly generated strong symbols have inner subsets of the same type and hides such subsets (removes them from the parse table). The cleanup process can be very expensive; its execution should be optimized.

HumorESK uses a layered grammar: the rules are partitioned into successive layers (e.g. NP layer, VP layer, compound sentence layer), the parser is restarted at each layer. The cleanup process runs only at the end of the layers.

2.2. Strong rules

Strong rules are motivated by the border problem. Strong rules suppose that there are structures in the language that are always definitely parsable (humans always prefer this parse even if formally there are other possibilities); and mark them as undividable units.

For example, in Hungarian, the postpositional NP is such a structure: the postposition always belongs to the preceding NP[2]; even if morphological ambiguity lets other possibilities.

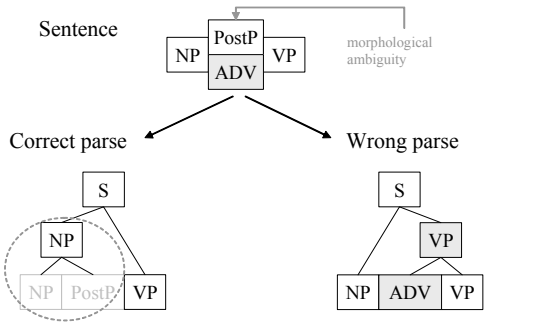

Figure 3: Strong rule for postpositional NP

Often, the postposition has the same word form as the adverb part of a phrasal verb (Figure 3); so, there are two formally correct possibilities for a full coverage, but humans always prefer the NP-PostP version. If we mark the rule that unifies the NP and the PostP as strong, it hides the unified symbols (hiding is marked as light gray lines). So, the wrong parse gets disabled: it fails since the first word of the sentence is hidden (by the strong rule) and without this word no full coverage is resulted. (The figure shows the wrong parse in the case of a non-hidden first word.)

[1] Exceptions are very rare and can be handled with specific rules.

[2] The word order might be unclear for the readers not familiar with the Hungarian language: 'behind the chair' is expressed as [the][chair][behind-of]

Strong rules can be used to describe definite or preferred grammatical structures (e.g. NP-PostP in Hungarian) or proverbs and sayings.

Implementation model

The simplest implementation model is to add a visibility field to the symbols in a normal CFG execution model. Strong rules set the visibility value to false. This model is the basis of the detailed one (see Section 4).

3. Application areas of the "strong rules"

There are several application areas for the strong rules.

3.1. Strong rules for the border problem

First of all, strong rules can be used as a solution for the border problem. Hiding the unified constituents defines the borders of the structure, so the spread of the surrounding ambiguity is stopped.

3.2. Strong rules for mutual exclusion of rules

Strong rules can also be used for ranking rules that are applicable for the same set of symbols.

Suppose that there are two rules: a general one and a specific one. The goal to achieve is mutual exclusion: if the specific rule is applicable the general one should not be applied. (In classical models there is no possibility for that.) This problem often occurs when working with proverbs: the proverb-recognizer rule should disable the word-by-word processing rules.

Making the specific rule strong implicitly gives a solution. If the specific rule cannot be applied, obviously the general rules are executed. As the specific rule is applicable, it hides the unified symbols, so the general rules will have nothing to be applied on.

3.3. Strong rules for inner structure disambiguation

In several cases, only partial information of the parse tree is used by the caller (e.g. by an information extraction system). Often, the order of the rule applications is indifferent: e.g. it doesn't matter if the noun is first unified with the preceding adjectives and only after the succeeding postposition or vice versa; both situations show the same amount of information, the same dependency graph (Figure 4). In such cases, it is needless to generate both parse trees.

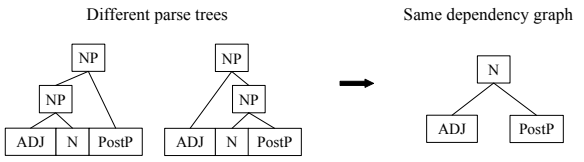

Figure 4: The parse tree is indifferent

Marking one of the irrelevant ordered rules strong automatically discards the other tree.

3.4. Strong rules to decrease the number of symbols in the parse tree

The parse speed is usually proportional to the number of symbols in the parse table. Decreasing the symbol number speeds up the whole process.

The strong rules keep the number of symbols low by hiding the unified constituents. It is useful to implement the visibility property so that the hidden symbols are also physically removed from the effective parse table.

With strong rules, it is possible that the number of symbols in the parse table decreases. Figure 5 shows an example for that (3→2). The strong Rule#1 hides two unified constituents but creates only one new while disables the other rule. At special cases it may happen that there are fewer (visible) symbols at the end of the parse than at the beginning.

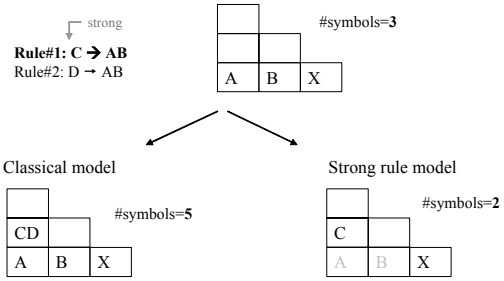

Figure 5: Number of symbols in the parse table

4. Implementation issues

The simple implementation model didn't include neither conflict resolution nor consistency check.

4.1. Conflict resolution

Figure 6 shows an example for the conflict between multiple strong rules. Let us suppose that the noun has got two different morphological parses (e.g. in English "play" as fun and "play" as drama). As the strong-marked *NP→N+PostP* rule is applied for the first parse, it hides the PostP; there nothing remains to unify with the second parse of the noun. The desired behavior would be that both NP-as are generated and all the three symbols are hidden.

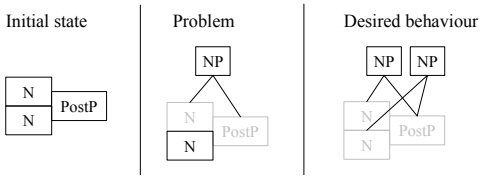

Figure 6: Conflict situation

The conflict is solved if the symbols are not directly hidden by the strong rule. A wait-counter is defined for each symbol; this counter registers the potential unification partners (the neighbors). When a potential unification partner disappears or gets unified, the wait counter decreases. When a new partner appears, it increases. Symbols become invisible as the wait counter reaches zero; say there is no more possibility to early hiding.

The wait counter can be managed using Dijkstra's P and V primitives. The following algorithm solves the problem for 2-membered CFG rules with multiple first elements (the parse goes from left to right, *r* is the wait counter)

- Initially, *r* equals to the number of symbols at the previous position. For the symbols at the first position of the sentence, *r* is 1.
- During unification, the new symbol acquires the symbols at the next position (P primitive)
- For strong rules, at unification, the first constituent releases the second one (V primitive)
- For any symbol, if the wait counter reaches zero, the symbol gets invisible (and gets moved away from the effective parse table.). The invisible symbol releases everybody.

Using the P and V primitives makes it possible to use the strong rules in concurrent processing systems, too.

4.2. Consistency

Inconsistency is when a symbol that has just been hidden by a strong rule was already used by a non-strong one. If the grammar fulfills a few formally checkable requirements, no consistency check or restore is needed. In this case the strong-rules are executed first and non-strong rules come only in the second round (for more details see [7] and [8]).

5. Evaluation results

Both strong rule models have been implemented and empirically evaluated. Basically, two efficiency aspects have been examined: the speed factor that has been measured with the number of symbols in the parse table and the correctness factor (if the results contain the correct parse).

We compared the strong rule model (SRP), the strong symbol model (HumorESK) and a classical model (CYK).

For the correctness measurement, a news corpus of 10000 sentences was used. For the speed measurement, a random part of the whole corpus consisting of 100 (compound) sentences was used. The average length of a simple sentence was 6.77 words, this resulted in 10.63 initial symbols in the parse table (due to the morphological ambiguity).

5.1. Number of symbols

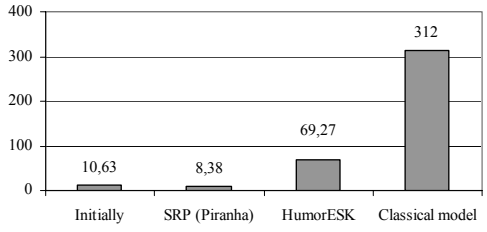

Figure 7: Number of symbols

The measurements examined the number of symbols at the end of the NP level. The reason for choosing this point was that the detailedness of the parse trees was very different at the higher levels; so it would be inappropriate to compare them.

Both strong models decreased the number of symbols considerably. In the case of the strong symbol model a decrement of 1 magnitude, in the case of the strong rule model a decrement of 2 magnitudes was experienced. The strong symbol model was found to contain less visible symbols at the end than at the beginning of the parse.

5.2. Correctness

Generally, the number of parse trees in the results decreased while the correct ones were kept. Both strong models were found to contain the correct parse tree in nearly all cases. HumorESK had a performance of 95%, Piranha 92%. The uncovered cases come from misspelled or morphologically misparsed words. The strong rules model is more sensitive for the wrong morphological parses.

6. Conclusions

The border problem is a significant factor in making the syntactical parse more efficient.

The "strong rules" model seems to be a suitable solution for the border problem while it has got several other application areas. This model helps to keep the number of symbols low while the correct parses are kept and several wrong parses are disabled.

The "strong rules" approach can be combined with other models (e.g. probabilistic) since they consider different aspects of the language.

7. Abbreviations

ADV	Adverb
CFG	Context Free Grammar
CYK	Cocke-Younger-Kasami algorithm
N, NP	Noun, Nominal Phrase
PostP	Postposition
VP	Verbal Phrase

8. References

[1] D. Jurafsky, J. H: Martin, *Speech and Language processing*. 2000, Prencice Hall, New Jersey

[2] Roark, Brian, Charniak, *Measuring efficiency in high-accuracy, broad-coverage statistical parsing*. 2000, In Proceedings of the COLING-2000 Workshop on Efficiency in Large-scale Parsing Systems

[3] C. D. Manning, H. Schutze*, Foundations of Statistical Natural Language Processing*. 1999,The (MIT) Press

[4] Paul Smolensky, *The Initial State and `Richness of the Base' in Optimality Theory*. 1996, John Hopkins University

[5] HumorESK parser of the MorphoLogic Ltd, http://www.morphologic.hu

[6] SRP (Piranha) parser of the Budapest University of Technology and Economics, http://piranha.hit.bme.hu/srp

[7] B. K. Benkő: *Sentence-level parsing algorithms for Hungarian,* 2003, Budapest University of Technology and Economics (in Hungarian)

[8] B. Benkő, T. Katona, P. Varga, *Processing Hungarian texts for information retrieval*, 2003, In Proc. of Hungarian scientific students' conference (in Hungarian)