BORBÁLA KATALIN BENKŐ
Technical University of Budapest
Budapest, Hungary
bbenko@hit.bme.hu

Increasing the syntactical parse efficiency using "strong rules"

1. Introduction

Syntactical parsing is a very important sub-task of the high-level text processing, such as content extraction, intelligent search and machine translation. [1] The result returned by the syntactical parser is the basis of the further manipulations: transformations, deduction and data extraction.

From the application's point of view, the parser is required to be efficient: thus the parse process must be fast and the result must be unambiguous and correct.

- Speed is especially important in quasi real time applications. For example, at a "daily news processing system" the parsing phase must end in less than 24 hours; otherwise it cannot be called "daily".
- High ambiguity makes the results unusable. Nobody needs a translator that gives 300 translations for a single sentence, just because the syntactical parser of the source language returned 300 different parse trees.
- Correctness is strongly related to the ambiguity. In several cases, the results contain both correct and incorrect trees. The incorrect trees result in false facts in the system, so the deduction is corrupted.

Traditionally, the efficiency factors are improved via weighing and filtering. [2][3] A weight value is assigned to the symbols; symbols with too low values are dropped. The user application may consider only the trees with high values. In the widely used probabilistic approach the weight values represent probabilities. [4][5]

In spite of the probability assignments and filtering, parsers still spend considerable amount of time on building wrong trees. The speed decrement is not only the problem; the wrong partial parse may get to the results, causing loss of correctness and increasing the ambiguity. The weighting-filtering solution that the classical methods offer to increase the efficiency is not enough. This paper presents a formal model that increases both efficiency factors (speed and ambiguity-correctness) by considering such language aspects that were not considered by the classical models.

1.1 Weaknesses in the classical–probabilistic approach

The Chomsky-based rule model and the probabilistic approach provide no solution for the following problems:

- *Mutual exclusion*. There is no possibility to define that rules mutually exclude each other. This is an elementary problem that we face when working with sayings and proverbs. In this case, the proverb-specific rule must disable the word-by-word parsing rules.
- *Rule weighting based on information taken from the adjacent symbols*. In several cases, the adjacent symbols also influence the probability of the current structure. It is impossible to describe in the probabilistic model since it considers only inner information. This is especially important at the border problem (see in Section 2.1).
- *Dynamic filter levels*. Constant filter levels are not sufficient. For example, at a saying, the word-by-word parse should be dropped; while the same parse is kept if it is not a saying (e.g. one word is different).

1.2 Goals

The goal of the original research was to improve the parse efficiency – namely to speed up the parse, to reduce the parse ambiguity and to improve the correctness – by identifying and disabling inefficiency sources. The efficiency leak is caused by several factors, this paper deals with one significant part the border problem.

2 The "strong rules" model

2.1 The border problem

The border problem is when a constituent is mistakenly assigned to a neighboring one, crossing the border between the structures. The border problem occurs when

- There is an ambiguous symbol that has both correct and wrong parses.
- At the unification phase, the tree containing the wrong parse doesn't stop at the border of the structures but it ranges into the neighboring one.

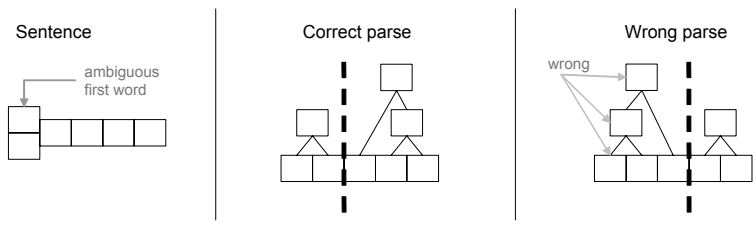

Figure 1: Border problem: the wrong parse crosses the border

Figure 1 shows an example for the border problem. The first word of the sentence is ambiguous; it has one correct and one wrong parse (gray). In the correct parse, the first two and the last three words form constituents. In the wrong parse, the border between the constituents is crossed; maybe, also resulting a full coverage.

The border problem causes multiple efficiency loss:

- *Speed loss*: the parser spends time on building wrong trees.
- *Ambiguity-correctness loss*: the wrong parse cannot be eliminated, on the higher levels more and more symbols will contain the wrong parse in the parse table.

The border problem cannot be solved in the general case but it can be reduced in special cases.

2.2 The "strong rule" idea

All natural languages contain definitely unambiguously parsable structures; say humans always prefer this parse even if the morphological ambiguity lets other possibilities. If we mark the borders of these definitely parsable structures so that adjacent trees are not allowed to cross the borders, the border problem is reduced: the adjacent wrong trees cannot partially extend into the correct structure. This is the basic idea of the "strong rules".

Managing border marks doesn't fit into the chomskian rule model, it raises several questions about conflict resolution and partially overlapping border marks. Instead of marking the borders, the strong rule model uses an alternate solution: the constituents of the definitely parsed structure get invisible and so inaccessible. Hiding is advantageous, as

- The constituents are hidden, so the adjacent wrong trees cannot access them for unification.
- The number of symbols in the parse table decreases; which fastens up the parse process.

The idea of the "strong rules through constituent-hiding" can be concretized in two different ways: as strong rules or strong symbols. Both models have been implemented and evaluated, see in Section 4.

2.3 Strong rules

The strong rule version uses classical context free rules with the extension that some of the rules are marked strong. When a strong rule is applied, it hides the unified constituents and leaves only the newly created unified symbol in the parse table.

Beyond the language-specific situations, the strong rules are also very useful when working with sayings and proverbs. Figure 2 shows an example for a language-specific situation. In Hungarian, the PostP (postposition) of an NP (nominal phrase) has often got the same form as the adverb part in a phrasal verb. This causes a theoretically ambiguous parse while humans always prefer the PostP version[1]. If we mark the Rule#1 in Figure 2 strong, it hides the unified NP and PostP (dashed circle at the "Correct parse" part). So the wrong parse is disabled since there will be no visible symbol in the first position (in the figure, the wrong parse is drawn in the case of a non-strong Rule#1).

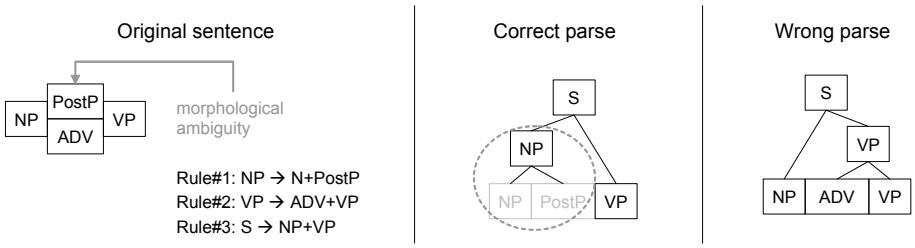

Figure 2: Strong rule in a language-specific case

2.4 Strong symbols

In the strong symbol model, not the rules but the symbols are strong. If a symbol (e.g. a NP) is strong, it hides all constituent of the same type. Figure 3 shows an example where a 3-word NP hides its 2-word NP constituent.

The strong symbols were motivated by the always-expanding behavior of the Hungarian NPs. If a constituent can be unified with an NP, it is always unified with it.

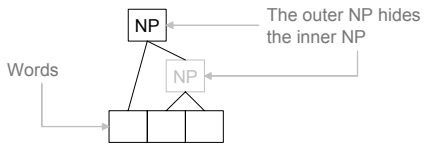

Figure 3: Example: NP is a strong symbol

2.5 Implementation

Strong rules can be integrated into a normal rule execution model (e.g. CYK – Cooke-Younger-Kasami) with adding a visibility field to each symbol; the state of the visibility field is modified when a rule is executed. As the visibility field is set to false the symbol gets inaccessible.

[1] In Hungarian the postposition directly follows the noun, e.g. *"next to the mouse"* means *"az egér mellett"* [the][mouse][next-to]

The implementation must contain inconsistency handling and conflict resolution.

Inconsistent state is when a symbol that is just unified by a strong rule was previously used by a non-strong rule. At inconsistent states, the consistency must be restored via rollback of that previously mistakenly applied non-strong rule. This consistency restore step may be very expensive so rollbacks should be prevented or at least the number of rollbacks should be optimized.

- *General case – optimizing the number of consistency checks*. The consistency check is not needed to run after each rule application. It is better when a normal rule execution model (.e.g. CYK) is used, periodically interrupted with the consistency check. If the parser uses layered grammar, it is sufficient to do a consistency check at the end of the layers only.
- *Special case – preventing the inconsistency*. If the grammar fulfills a few formally checkable conditions, the consistency saving rule execution can be achieved. The rules are divided into two subsets s1 and s2 so that s2 only non-strong rules, and there is no backward relationship to s1 (there is no rule in s1 that unifies symbols coming from s2). In this case, after running the rules of s1 there is no more need to prevent inconsistency, since there is no more strong rule that can be run after a non-strong one. The generalized version of the inconsistency prevention has got a very efficient implementation model.

Conflict situation is when multiple strong rules are applicable for a given position in the sentence. Conflict is e.g. when a noun has got two correct but different morphological parses and the strong ADJ+NP rule is applicable for both versions. An efficient solution is presented in [6], based on Dijksta's P and V primitives.

3 Other issues of the strong rule model

3.1 Application areas

Strong rules can be used in several application areas.

Parse speedup. The original goal and the main application area of the strong rules is to speed up the parse. This is achieved (1) through decreasing the number of symbols in the parse table[2] and (2) via reducing the number of wrong trees.

General ambiguity reduction. Strong rules mark certain symbols as definitely parsed, so no further ambiguity is accepted on these symbols. Reduces the total ambiguity, too.

Mutual exclusion between rules. Let us consider the situation that there are two rules: a general one and a special one. The desired behavior is: if the special rule is not applicable the general one should be applied; if the special rule is applicable the general one should be disabled. As mentioned before, this situation is typical when working with sayings and proverbs.

The mutual exclusion problem is solved if we mark the special rule "strong": When it is not applicable, the strength mark doesn't take effect; as it is applicable, it hides the unified constituents so the general rule is disabled.

Inner structure disambiguation. >From the application's point of view, often, only a part of the parse tree is important. For example, the application uses dependency tree, and multiple parse trees result the same dependency tree (see in Figure 4). In this case, it is useless to generate all the trees, since the second one doesn't hold additional information.

If we mark one of the rules "strong", only that tree will be produced.

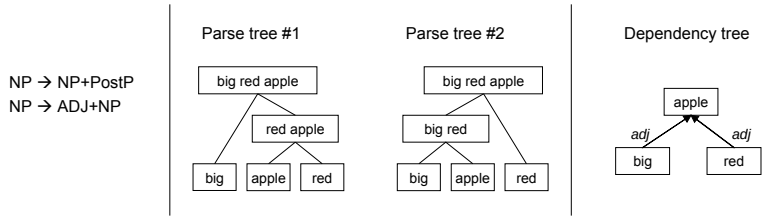

Figure 4: Multiple parse trees result the same dependency tree

3.2 Problems and non-application areas

While appropriately used strong rules significantly improve the parser; inappropriate usage causes efficiency loss. The strength marks must be handled with care. Figure 5 shows an example where a rule (the second one) is mistakenly marked strong; as a result, the overlapping strong rules exclude the possibility of a full coverage: the symbol C is hidden by the mistakenly strong second rule, so the sentence symbol S cannot be generated

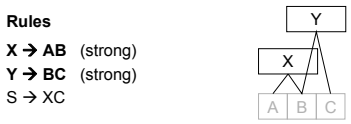

Figure 5: Overlapping strong rules make the sentence non-parsable

The whole model is based on the idea that the strong-marked rule is *always definitely correctly* applied. If this condition is true, no partial overlap problem can occur.

[2] The hidden symbols are removed from the effective parse table.

Strong rules are not suggested to use in *mostly correctly* applied cases – or, the exceptions must be handled using other rules or tricks. The *always correct* condition also holds for the previously mentioned application areas "mutual rule exclusion" and "inner structure disambiguation".

4 Experimental evaluation

Both the strong rules and the strong symbols model have been implemented and empirically evaluated. The parser SRP (Strong Rule Parser) [7][8][9] of the Budapest University of Technology and Economics uses strong rules; HumorESK (Hungarian Morphology Extended with Syntactical Knowledge) [10][11][12] of the MorphoLogic Ltd is based on strong symbols. The two systems were compared with each other and with a reference engine (a classical, Cooke-Younger-Kasami based parser).

Two comparisons were carried out. The first one examines the strong rules' or symbols' effect on the parse speed; the second one examines the correctness factor. Random subsets of a Hungarian short news corpus of 10000 sentences were used for the comparison.

4.1 Parse speed

At the time of the speed comparison, the three parsers ran in considerably different hardware and software environments[3], so it is inappropriate to measure the parse time. The speed of the parse is proportional to the number of symbols in the parse table; so these values are used.

The compound sentences of the corpus were divided into simple sentences for SRP and for the reference engine. In average, one simple sentence consisted of 6.77 words; which resulted in 10.63 initial symbols in the parse table due to the morphological ambiguity.

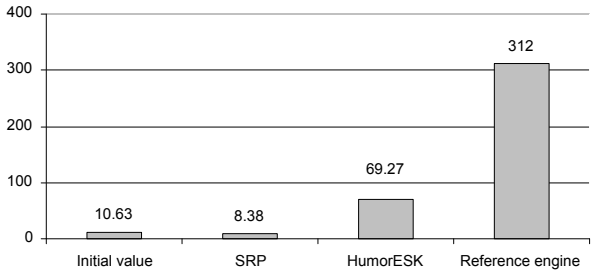

Figure 6: Number of symbols in the parse table

Figure 6 shows the measurement results. Both strong models show significant decrement compared to the reference engine. The strong rule model decreased the number of symbols to 6% of the original value (which is 2 magnitudes!); the strong symbol model shows a decrement of 53% (1 magnitude). The result may be strange at the first sight, because SRP contains less visible symbols at the end than in the beginning. This can be explained by the percentage of strong rules (70% of the rules in SRP were strong) and the radical symbol-number-decreasing property of the model.

4.2 Correctness/ambiguity

At the second measurement, the acceptability of the result was examined. The answer of the parser was declared to be acceptable if it contained at least one correct parse tree. At this measurement, no reference engine was used; because the reference engine continuously exceeded the time limit for parsing one sentence.

Both strong parsers were found to be very acceptable; SRP has the value 92%, HumorESK has 95%. The uncovered cases come from misspelled or morphologically mis-parsed words or from the time limit. The strong symbol model is more sensitive for the wrong morphological parses.

Although there were no measurements, it can be said that the total number of results decreased; while the correct trees were kept. High percentage of the cases that were uncovered by the reference engine got a correct coverage. In general, both the ambiguity factor was reduced (by eliminating several wrong/similar parses) and the correctness was increased (by reducing the number of wrong parses and covering the uncovered cases).

5 Conclusions

The authors of this paper tried to find methods to improve the efficiency of the syntactical parsers, namely the speed and the ambiguity/correctness factors. The border problem was identified as considerable source of the inefficiencies. The design goal of the strong rule model was to reduce the negative effects of the border problem.

Analysis shows that the strong rules have several interesting properties and alternative application areas, such as applicability for mutual rule exclusion, which is a problem insolvable in the classical parsing models.

Empirical evaluations confirmed that the appropriate usage of strong rules result significant efficiency improvement.

[3] HumorESK ran on Pentium4/Windows, SRP on Pentium2/Unix, reference engine on Pentium3/Windows

6 References

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

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

[3] 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

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

[5] C. Chelba, D. Engle, et al, *Structure And Performance Of A Dependency Language Model*, 1997, in Proc. Eurospeech '97

[6] B. K. Benkő: *On increasing the syntactical parse efficiency*, 2004, In Proc. of The Seventh All-Ukrainian International Conference on Signal/Image Processing and Pattern Recognition, Kyev

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

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

[9] B. K. Benkő, T. Katona, P. Varga, *Processing Hungarian texts for information retrieval*, 2003, In Proc. of the XXX. Hungarian National Scientific Students' Conference (in Hungarian)

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

[11] G Prószéky, *Syntax As Meta-morphology.* 1996, in Proceedings of COLING-96,

[12] G. Prószéky, *Lexical Information and Decisions in Parsing*. 1999, in Cristea, Dan, et al (Eds.) 4th Eurolan Summer School on Human Language Technology, Technical Report 99-02, Iasi, Romania.