THEORIE-TAG 2023 A. Lin, G. Zetzsche (Eds.): Theorietag 2023, Kaiserslautern, 4. – 6.10.2023
RPTU & MPI-SWS Kaiserslautern, Technical Report, pp. 41–45.

Error-Correcting Parsing – This Time We Want All!

Florian Bruse Stefan Kablowski Martin Lange

Theoretische Informatik / Formale Methoden, Universität Kassel
{florian.bruse, martin.lange}@uni-kassel.de

The Problem and its Motivation. A well-known problem in the theory of formal languages is that of error-correcting parsing: given a, say, context-free language L over some alphabet Σ, and a word $w \in \Sigma^*$, compute a word $v \in L(G)$ s.t. $\Delta(w,v)$ is minimal, where $\Delta : \Sigma^* \times \Sigma^* \to \mathbb{R}^{\geq 0}$ is some fixed distance metric. Here we are are solely concerned with the Levenshtein metric [9] which measures distance between two words as the minimal number of insertion, deletion or replacement operations on letters that turn one of them into the other.

It has been shown that error-correcting parsing (for context-free languages) is conceptually not much more difficult than ordinary parsing; there are solutions based on Earley's parser or the CYK algorithm which, given some w, compute a parse tree for some $v \in L(G)$ and a minimal sequence of edit operations that turn w into v, cf. [1, 10].

The requirement of minimality in the formulation of error-correcting parsing is important. Suppose a file contains a syntactically misshaped Java program. There are of course many ways to edit it to become a correct one, for instance by deleting it entirely and inserting the infamous hello-world program. Such a correction is not minimal in general, though. Minimality allows us not to introduce a notion of semantical distance between correctly formed and ill-formed programs; instead we simply assume that the faulty program originated from a correct one, and the process that introduced syntactic errors is governed by statistical laws so that the program that is obtained by applying a minimal correction is most likely the original one.

There are, however, applications of parsing in which the reason for some w not belonging to L cannot be found in some "original" $v \in L$ which has been modified to w under laws of statistics. Consider the following scenario. The curricula of natural sciences secondary-education classes typically contain experimental lessons whose purpose it is to teach pupils the principles of scientific discovery and reasoning. They are given a research question and are asked to formulate a matching hypothesis and then to (in-)validate it using some experimental setup. The Theoretical Computer Science / Formal Methods group at the Univ. of Kassel is involved in the development of a digital learning tool that initiates adaptive learning by giving feedback on each step of the process, from formulating a hypothesis to checking its (semantic) correctness [7]. The set of syntactically correct hypotheses can easily be formalised by a context-free grammar, and it is not hard to imagine that hypotheses formulated by some 8th-grade pupil are not always grammatically correct. However, here it is neither right nor helpful to automatically apply a minimal correction in order to continue with the semantical checks, for the following two reasons.

- The cause of syntactical incorrectness may be more than merely a typo; it could be that the pupil has not fully understood the grammatical structure of hypotheses yet.
- For learning purposes, it is better to present the pupil with some feedback on why his/her formulation is misshaped and let them correct it.

This leads to the following problem UECP of *universal* error-correcting parsing.

given:	a context-free language L over some alphabet Σ, and a word $w \in \Sigma^*$
compute:	the set of *all* minimal corrections ρ s.t. $\rho(w) \in L$

As it turns out, in order to suit the application sketched above, we also need a more relaxed notion of minimality. Consider, for example, the following attempt at formulating a hypothesis w.r.t. the research question "*Does temperature influence yeast growth?*"

yeast grows it is warm

An obvious way to correct this would be to insert the word *when* in position 2, forming "*yeast grows when it is warm.*" So the edit distance to a correctly formed hypothesis is 1, and there are also other ways to execute a single edit to form a correct sentence with potentially different meaning, for instance inserting *because*, *if*, *and*, etc. However, maybe the author of this pre-hypothesis has a different idea of causality and actually tried to state

if yeast grows then it is warm

or they actually rightly predicted another aspects of the influence between temperature and yeast growth but failed to formulate

yeast grows unless it is hot

at edit distance 2. So while these do not reside at an edit distance of *minimal length*, the former should definitely be considered to be a minimal correction in the sense that it results from a minimal set of edit operations (here: insertions only) that create a valid sentence. The notion of minimality that is formally defined below, does not capture the latter, though. Note that the two edit operations – inserting "*unless*" at position 2 and replacing "*warm*" with "*hot*" at position 5 – are independent; they can be carried out in any order (with appropriate adjustments to the index positions). There is one particular order, namely the one stated here, applying the insertion before the replacement, which leads to an intermediate word in the language, namely "*yeast grows unless it is warm.*" This is why we do not consider this correction to be minimal – there is an order of its edit operations which produces a word in the language before all edits are being carried out.

A Theory of Minimality in Corrections. A *deletion*, resp. *insertion operation* is written $a{\downarrow_i}$, resp. $a{\uparrow_i}$ for $a \in \Sigma$, $i \in \mathbb{N}$. A *replacement operation* is written $a/_i b$ for $a, b \in \Sigma$, $i \in \mathbb{N}$. An *(edit) operation* is either of these three.

Each operation α induces a partial map of type $\Sigma^* \to \Sigma^*$, straight-forwardly realising the effect of deleting, inserting or replacing a symbol at a particular position in a word. A *correction* is a (possibly empty) sequence $\rho = (\alpha_1, \ldots, \alpha_m)$ of operations. The effect of the application of a correction to a word, or simply *correcting* the word, is explained by a homomorphic extension of the effect that singular edit operations have: $\rho(w) := \alpha_m(\ldots\alpha_1(w)\ldots)$.

Two corrections ρ, ρ' are *equivalent* if $\rho(w) = \rho'(w)$ for all $w \in \Sigma^*$. A correction ρ is *normalised* if

$$\rho = (a_1/_{i_1}b_1, \ldots a_n/_{i_n}b_n, c_1{\downarrow}_{j_1}, \ldots, c_m{\downarrow}_{j_m}, d_1{\uparrow}_{h_1}, \ldots, d_k{\uparrow}_{h_k}) \tag{1}$$

for some n,m,k s.t. $i_1 > \ldots > i_n$, $j_1 > \ldots > j_m$ and $h_1 \geq \ldots \geq h_k$.

It is possible to define rules of a rewrite system $\rightarrow$ operating on pairs of edit operations that are sound w.r.t. equivalence. For instance, we would have

$$\rho, \mathit{unless}{\uparrow}_2, \mathit{warm}/_5\mathit{hot}, \rho' \quad \rightarrow \quad \rho, \mathit{warm}/_4\mathit{hot}, \mathit{unless}{\uparrow}_2, \rho'$$

for any ρ, ρ'. Likewise, some combinations cancel each other out like a deletion of a letter following its insertion, and other pairs can be shortened; e.g. a deletion followed by an insertion of a different letter at the same position can be rewritten into a replacement. We leave it as an exercise to formulate up to $3 \cdot 3 \cdot 2 = 18$ rules covering the cases in which an operation of one of the three types is followed by another at either the same or the succeeding position in a word.

Proposition 1 *Every correction ρ is equivalent to a normalised ρ' s.t. $\rho \rightarrow^* \rho'$.*

Hence, it suffices to only consider normalised corrections henceforth. We write $\rho' \preceq \rho$ if ρ' is a subsequence of ρ. We say that ρ is a *$\preceq$-minimal correction* (for some CFG L and a word w), if $\rho(w) \in L$ and $\rho'(w) \notin L$ for every $\rho' \prec \rho$. We write $\mathcal{C}_L(w)$ for the set of normalised ρ s.t. $\rho(w) \in L$, and $\mathcal{C}_L^{\mathsf{min}}(w)$ for the set of $\rho \in \mathcal{C}_L(w)$ that are $\preceq$-minimal. The following result is important in order to make UECP well-defined.

Proposition 2 *Let L be a CFL, $w \in \Sigma^*$. Then* (I) *$\mathcal{C}_L(w)$ is a context-free language over a finite alphabet of edit operations, and* (II) *$\mathcal{C}_L^{\mathsf{min}}(w)$ is finite.*

Computing Minimal Corrections: Theory and Practice. Prop. 1 can be used as a basis for a simple but highly inefficient enumeration procedure for solving UECP [5]: given L and w, enumerate all corrections ρ in normal form and check for each of them whether

- $\rho(w) \in L$ by computing $\rho(w)$ straight-forwardly and then using a standard parsing algorithm for CFLs, and
- then compute the necessarily finitely many $\rho' \prec \rho$ and equally check $\rho'(w) \notin L$ for all of them.

Part (II) of Prop. 2 ensures that this procedure can be terminated at some point.

There is, however, a better way to solve UECP by internalising the construction of (minimal) corrections into the parser's work. Just as an ordinary (non-error-correcting) context-free parsing, the problem opens itself up to a solution using dynamic programming as corrections for a language L and word w can be built from corrections for the subwords of w and potentially different languages. It is not clear, though, whether minimality can be maintained in such a modular way, too.

There is a conceptually simple way to extend the CYK algorithm [11, 6, 12, 3] to UECP. Given a CFG G and a word $w = a_0 \ldots a_{n-1}$, we maintain, likewise, a table T of entries for each subword represented by a pair (i,j) with $i \leq j$. However, unlike the original CYK algorithm which only stores a set of nonterminals A in entry (i,j) s.t. $A \Rightarrow^* a_i \dots a_j$, we store a set of pairs of nonterminals and minimal corrections (A,ρ) s.t. $A \Rightarrow^* \rho(a_i \dots a_j)$. We then just need the following amendments, resp. adjustments.

- A table entry $T(i,i)$ is filled with pairs (A,ε) whenever $A \to a_i$ as in CYK, and additionally
  - with pairs $(A, b/_i a_i)$ whenever $A \to b$,
  - with pairs $(A, a_i{\downarrow}_i)$ whenever $A \to \varepsilon$.
- We get $(A,\rho) \in T(i,j)$ for $j \geq i$, whenever $A \to BC$ and there are h with $i \leq h < j$, ρ', ρ'' s.t. $(B,\rho') \in T(i,h)$, $(C,\rho'') \in T(h+1,j)$ and ρ is the normalisation of $\rho'\rho'''$ where ρ''' results from ρ'' by shifting all indices by the number of insertion operations minus the number of deletion operations in ρ'.

  Note that this can potentially add multiple entries with the same nonterminal in a table entry. Whenever $(A,\rho),(A,\sigma) \in T(i,j)$ and $\rho \preceq \sigma$ then (A,σ) is removed from $T(i,j)$.
- At last, note that so far, no insertion operations are generated. We first observe that a sequence of insertions operating consecutively on a word can be ordered and grouped into parts that consecutively insert letters at the same position. In the special case of the word to apply them to being ε we easily see that sequences of insertions of the form $a{\uparrow}_0$ for some $a \in \Sigma$ suffice to turn ε into any target word. It then only remains to see that it suffices to pre-compute, for any nonterminal A, a set I of pairs of nonterminals A and minimal pure insertion corrections $\sigma = b_0{\uparrow}_0 \dots b_{\ell-1}{\uparrow}_0$ s.t. $A \Rightarrow b_{\ell-1} \dots b_0$. These can be pre-computed once and then used in the following way to additionally fill table entries (i,j) with $i \leq j$.
  - Whenever $A \to BC$, $(B,\rho) \in T(i,j)$ and $(C,\sigma) \in I$, then add (A,ρ') to $T(i,j)$ where ρ' is the normalisation of $\rho \cdot \sigma'$ and σ' is obtained from σ by setting all position indices to $j+1$ plus the difference of insertions and deletions in ρ as above.
  - Whenever $A \in BC$, $(C,\rho) \in T(i,j)$ and $(B,\sigma) \in I$, then add (A,ρ') to $T(i,j)$ where ρ' is the normalisation of $\sigma \cdot \rho''$ and ρ'' is obtained from ρ by shifting all indices by $|\sigma|$.

Tests run with an OCaml implementation of this algorithm are promising in that it is possible to compute sets of minimal corrections for grammars with dozens of rules. In order to avoid costly normalisation in the grammar we build on a CYK variant that does not require Chomsky normal form [8]. The benchmarks also show, however, that sets of minimal corrections need not be small, and that in the light of the targeted application described above, it may be useful to further relax the notion of $\prec$-minimality s.t. that nature of computing more than just one correction is sufficiently retained.

We also aim to investigate the possibility to build a solution for UECP based on the Earley parser [4, 2] to see whether this would lead to a more efficient solution than the CYK-based one.

References

[1] A. V. AHO, T. G. PETERSON, A minimum distance error correcting parser for context-free languages. *SIAM Journal on Computing* **1** (1972) 4, 305–312.

[2] J. AYCOCK, R. N. HORSPOOL, Practical Earley Parsing. *The Computer Journal* **45** (2002) 6, 620–630.

[3] J. COCKE, J. T. SCHWARTZ, *Programming Languages and Their Compilers*. Courant Institute of Mathematical Sciences, New York, 1970.

[4] J. EARLEY, An Efficient Context-Free Parsing Algorithm. *Communications of the ACM* **13** (1970), 94–102.

[5] S. KABLOWSKI, *Computing All Minimal Corrections for a Word to Match a Context-Free Description*. B.sc. thesis, Univ. of Kassel, Germany, Faculty of Electr. Eng. and Comp. Sci., 2022. `https://www.uni-kassel.de/eecs/tifm/abschlussarbeiten/abg`

[6] T. KASAMI, *An efficient recognition and syntax analysis algorithm for context-free languages*. Technical Report AFCRL-65-758, Air Force Cambridge Research Laboratory, Bedford, Massachusetts, 1965.

[7] M. KASTAUN, M. MEIER, N. HUNDESHAGEN, M. LANGE, ProfiLL: Professionalisierung durch intelligente Lehr-Lernsysteme. In: *Bildung, Schule, Digitalisierung*. Waxmann-Verlag, 2020, 357–363.

[8] M. LANGE, H. LEISS, To CNF or not to CNF? An Efficient Yet Presentable Version of the CYK Algorithm. *Informatica Didactica* **8** (2009).

[9] V. I. LEVENSHTEIN, Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady* **10** (1966) 8, 707–710. Doklady Akademii Nauk SSSR, V163 No4 845-848 1965.

[10] S. RAJASEKARAN, M. NICOLAE, An Error Correcting Parser for Context Free Grammars that Takes Less Than Cubic Time. In: *Proc. 10th Int. Conf. on Language and Automata, Theory and Applications, LATA'16*. LNCS 9618, Springer, 2016, 533–546.

[11] I. SAKAI, Syntax in universal translation. In: *Proc. Int. Conf. on Machine Translation of Languages and Applied Language Analysis*. 1961.

[12] D. H. YOUNGER, Recognition and parsing of context-free languages in time n^3. *Information and Control* **10** (1967) 2, 372–375.