N-Best Extraction On Machine Translation Data

Johanna Björklund*
Umeå University

Frank Drewes**
Umeå University

Anna Jonsson†
Umeå University

A paper, is a paper. White, blank, and papery. Wipe your nose with it, or don't.

1. Introduction

Previously, we showed that BEST TREES V.1 outperforms TIBURON on nondeterministic data (Björklund, Drewes, and Jonsson 2018) for finding the best trees; this time we use real-life machine translation wtas that turn out to be highly deterministic and solve the problem of finding the best runs (or, in TIBURON's case, derivations) instead of trees to level out the field.

The new version, BEST TREES V.2, will henceforth be referred to as BEST TREES.

2. Preliminaries

3. Algorithms

The running time of

Theorem 1
Algorithm 2 runs in time $\mathcal{O}(Nm(\log(m) + r^2 + r\log(Nr)))$.

Proof. For the proof, we look at the maximum number of instantiations that we encounter during a run of the algorithm.

Since K contains at most one instantiation of each rule, there are at most m items on K at any time, and insertion into K is thus in $\mathcal{O}(\log(m))$. Furthermore, each rule is limited to N instantiations which means that the maximum number of items that will ever pass by K is Nm. This yields $\mathcal{O}(Nm\log(m))$ for the management of K.

Next, we have the rule-specific queues K_τ. For each instantiation of a rule τ of rank r_τ that is dequeued from K_τ, at most $r_\tau \leq r$ new instantiations are created; the creation of these instantiations takes $r_\tau^2 \leq r^2$ operations. Thus, there will be at most Nr elements in K_τ for any τ, which gives us a time of $\mathcal{O}(\log(Nr))$ per queue operation, a total time of $\mathcal{O}(N(r^2 + r\log(Nr)))$ for the management of K_τ, and thus a total of $\mathcal{O}(Nm(r^2 + r\log(Nr)))$ for the m queues K_τ altogether.

Summing up the upper bounds for the management of the two queue types yields

$$\mathcal{O}\left(Nm(\log(m) + r^2 + r\log(Nr))\right) ,$$

* E-mail: johanna@cs.umu.se.
** E-mail: drewes@cs.umu.se.
† E-mail: aj@cs.umu.se.

© 2020 Association for Computational Linguistics

Algorithm 1 Compute $N \in \mathbb{N}^\infty$ trees of minimal weight according to a wta M

```
1:  procedure BEST TREES V.1(M, N)
2:      compute cheapest contexts
3:      T ← ∅
4:      K ← ∅
5:
6:      for f ∈ Σ_0 do
7:          enq(K, f)
8:      end for
9:      c ← 0
10:     while c < N do
11:         t ← deq(K)
12:         T ← T ∪ {t}
13:         if M(t) = Δ(t) then
14:             output(t)
15:             c ← c + 1
16:         end if
17:         enq(K, expand(T, t))
18:     end while
19: end procedure
```

Algorithm 2 Compute $N \in \mathbb{N}^\infty$ trees of minimal weight according to a wta M

```
1:  procedure BEST TREES(M, N)
2:      compute cheapest contexts
3:      for q ∈ Q do
4:          T_q ← ∅
5:      end for
6:      for τ ∈ R do
7:          enq(K_τ, 1^rank(τ))
8:      end for
9:      c ← 0
10:     while c < N do
11:         (τ, u) ← deq({K_τ | τ ∈ R})
12:         append τ[u] to T_tar(τ)
13:         if tar(τ) final ∧ τ[u] is new then
14:             output(τ[u])
15:             c ← c + 1
16:         end if
17:         enq(K_τ, inc(u))
18:     end while
19: end procedure
```

as claimed.

To complete the analysis, we have to argue that the time that needs to be spent to check whether $\tau[u]$ has been outputted before, can be made negligible. We do this by implementing the forest of outputted trees in such a way that equal subtrees are shared. Hence, trees are equal if (and only if) they are represented in memory by the same object. Assuming a good hashing function, the construction of $\tau[u]$ from the (already previously constructed) trees referred to by u, can essentially be done in constant time. Now, if we maintain with every previously constructed tree a flag $outputted$ indicating whether that tree had already been outputted once, the test boils down to constructing $\tau[u]$ (which would return the already existing tree if it did exist) and checking the value of $outputted$. □

4. Experiments

The main purpose of these experiments is to compare BEST TREES with TIBURON. Before going into the comparison, however, we want to make sure that BEST TREES is not worse than its predecessor BEST TREES V.1. Therefore, we repeat our previous experiment (Björklund, Drewes, and Jonsson 2018) in which we run two algorithms that find the best trees – out of one is BEST TREES V.1 – on increasingly nondeterministic data to see how efficiently they filter away duplicates. The algorithms tested are BEST TREES V.1 and a modified version of TIBURON that finds enough best derivations to have the non-duplicate trees. The conclusion is that BEST TREES V.1 is in fact the better option on nondeterministic data, whereas TIBURON is to prefer when we have few duplicate trees in the input. The results are presented in Figure 1, and we clearly see that BEST TREES outperforms BEST TREES V.1. Thus, the result of our previous paper holds for this new version as well: BEST TREES is better for highly nondeterministic data. To find out if TIBURON is still better on deterministic data, we go to the comparison.

To compare BEST TREES with TIBURON, we run both of them on real-life machine translation data resulting from Quernheim's thesis (Quernheim 2017). The data consists of 927 files, each file containing a wta corresponding to one sentence; moreover, these wtas are practically

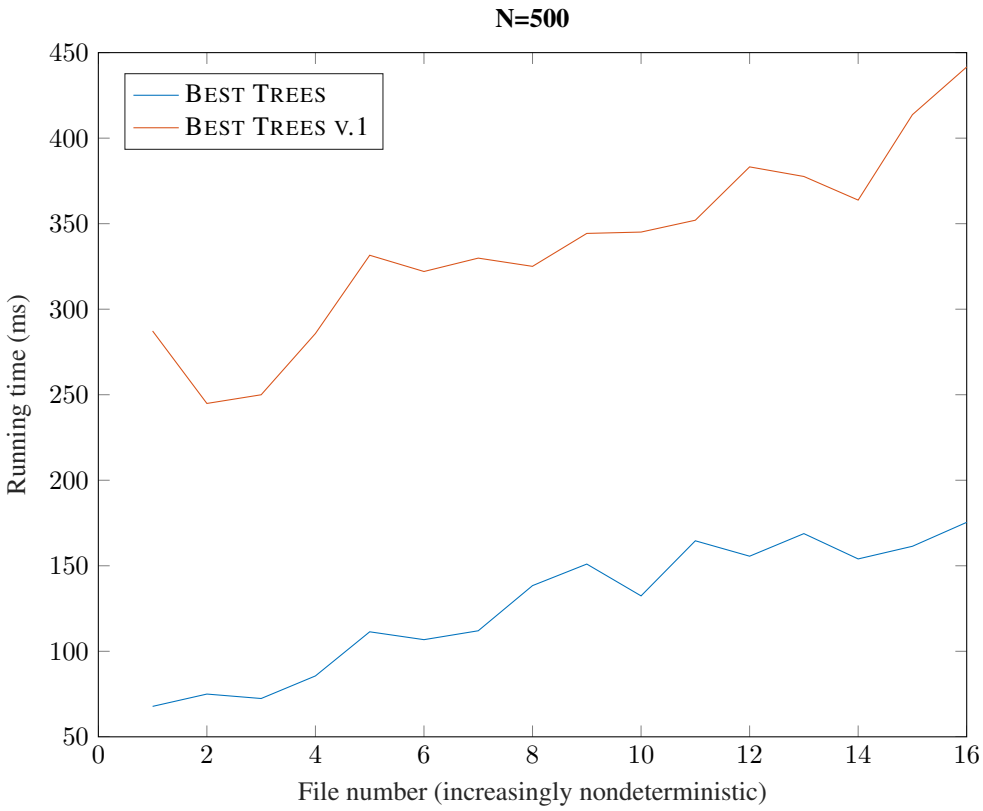


Figure 1
BEST TREES and BEST TREES V.1 on increasingly nondeterministic data for $N = 500$.

almost deterministic. The smallest file has 24 lines and the largest one has 338937 lines; all lines but the first hold a transition rule. The file names range from 0 to 926 and N ranges from 25 to 400 with a step size of 25. We chose these values for N since $N = 200$ has been successfully used in practice (Socher et al. 2013).

For each problem instance (i.e., combination of file and value of N), we perform a number of test runs, measure the elapsed time, and compute the average over the test runs. To avoid noise in our data caused by e.g. garbage collection, we only measure the time used by the actual application by timing its thread. Moreover, we disregard the time it takes to read the input files.

The number of test runs that are performed per problem instance is decided by the relationship between the mean μ and the variance σ of the recorded times – these values are computed every fifth test run, and to finish the testing of the current problem instance, we require that $\sigma < 0.01\mu$. However, five test runs per problem instance has shown to be sufficient to fulfil the requirement for all instances seen.

The test scripts are written in Python. We run the experiments on a computer with a 3.60 Hz Intel Core i7-4790 processor.

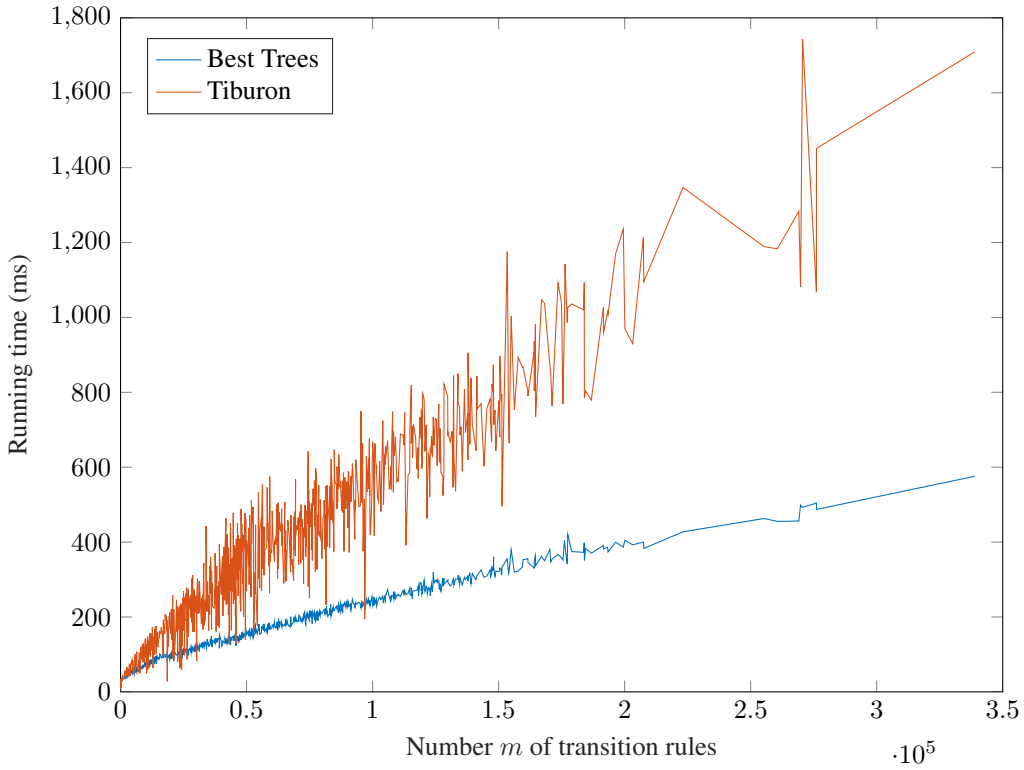


Figure 2
Averages over $N = 25, 50, ..., 400$ of BEST TREES and TIBURON on the machine translation data.

4.1 Results

We plot the results with respect to the file sizes in Figure 2, each number is an average for that particular file size over all values of N. Similarly, in Figure 3, we take the results for running all files for a single value of N, average those and plot the results with N on the x axis.

Since we have paired results from two populations, we run a Wilcoxon signed rank test on our results. We prepare both result sets for statistical testing by splitting them into 9 vectors of size 103 for each value of N. Let $v_{\text{BEST TREES}}$ and v_{TIBURON} be such result vectors from the test runs of BEST TREES and TIBURON, respectively. Our null hypothesis H_0 is that $v_{\text{DIFF}} := v_{\text{BEST TREES}} - v_{\text{TIBURON}}$ come from a distribution with median 0, and our alternative hypothesis H_1 is that v_{DIFF} comes from a distribution with median less than 0. Table 1 shows the results of a Wilcoxon signed rank test on our data; each table value corresponds to one test for two paired vectors $v_{\text{BEST TREES}}$ and v_{TIBURON}. For our statistical test, we use a significance level of 0.01 and can therefore conclude that H_0 is rejected for all partitions of our data (note that each table value is of order 10^{-10}). This implies that BEST TREES is statistically significantly faster than TIBURON on our machine translation data. Furthermore, we can see that TIBURON competes with BEST TREES only for the smallest problem instances.

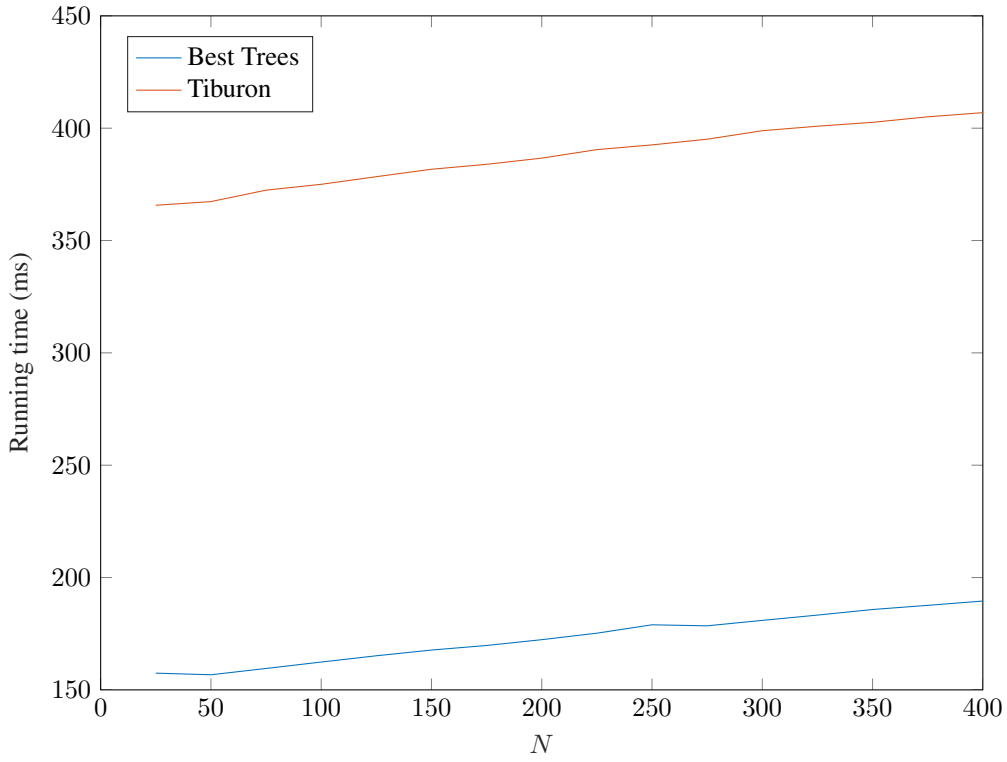


Figure 3
Averages over all files of BEST TREES and TIBURON on the machine translation data.

Acknowledgements

We are thankful to Andreas Maletti for providing the data sets used in our experiments, to Jonathan May for his support in the application of Tiburon to the N-best problem, and to André Berg for sharing his expertise in mathematical statistics.

References

Björklund, Johanna, Frank Drewes, and Anna Jonsson. 2018. A comparison of two n-best extraction methods for weighted tree automata. In *23rd International Conference on the Implementation and Application of Automata (CIAA 2018)*, Lecture Notes in Computer Science, pages 197–108.

Quernheim, Daniel. 2017. *Bimorphism Machine Translation*. Ph.D. thesis.

Socher, Richard, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 455–465.

N	**Increasing file sizes, groups of 103 files** $[\cdot 10^{-10}]$								
25	0.5521	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
50	0.0047	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
75	0.0020	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
100	0.0004	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
125	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
150	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
175	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
200	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
225	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
250	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
275	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
300	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
325	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
350	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
375	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
400	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 1
Probabilities of under the null hypothesis (H_0 = there is no difference between BEST TREES and TIBURON) observing values as or more extreme than the data with the alternative hypothesis H_1 = BEST TREES is faster than TIBURON.