

PAPER

raxtax: A *k*-mer-based non-Bayesian Taxonomic Classifier

Noah A. Wahl^{1,*}, Georgios Koutsovoulos¹, Ben Bettisworth¹
and Alexandros Stamatakis^{1,2,3}

¹Biodiversity Computing Group, Institute of Computer Science, Foundation for Research and Technology Hellas, 100 Nikolaou Plastira, 70013 Heraklion, Crete, Greece, ²Computational Molecular Evolution Group, Heidelberg Institute for Theoretical Studies, Schloss-Wolfsbrunnengasse 35, 69118 Heidelberg, Baden-Württemberg, Germany and ³Institute for Theoretical Informatics, Karlsruhe Institute of Technology, Kaiserstraße 12, 76131 Karlsruhe, Baden-Württemberg, Germany

*Corresponding author. nwahl@ics.forth.gr

FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

Abstract

Taxonomic classification in biodiversity studies is the process of assigning the anonymous sequences of a marker gene (barcode) to a specific lineage using a reference database that contains named sequences in a known taxonomy. This classification is important for assessing the complexity of biological systems. Taxonomic classification faces two inherent challenges: first, accuracy is critical as errors can propagate to downstream analysis results; and second, the classification time requirements can limit study size and study design, in particular when considering the constantly growing reference databases. To address these two challenges, we introduce **raxtax**, an efficient, novel taxonomic classification tool that explores common *k*-mers between all query sequence/reference sequence pairs. We also introduce two novel uncertainty scores which take into account the fundamental biases of reference databases. We further provide recommendations for interpreting the best reported matches. We validate **raxtax** on three widely used empirical reference databases and show that it is 2x–35x faster than competing state-of-the-art tools while being equally accurate. In particular, **raxtax** exhibits increasing speedups with growing query and reference sequence numbers compared to existing tools (for 50,000 and 1,000,000 query and reference sequences overall, it is 1.4 and 1.9 times faster, respectively), and therefore alleviates the taxonomic classification scalability challenge.

Key words: taxonomic classification, barcoding, biodiversity, phylogenetics, evolutionary biology

1. Introduction

Biodiversity researchers frequently need to address the question: Which species are present in my sample? A common solution consists in identifying and subsequently sequencing a well-conserved region of the genome which is present in all organisms under study [1–3]. Such regions, known as barcodes [4], are then used to identify species. The ribosomal 16S gene, the cytochrome oxidase 1 (COX1), and the internal transcribed spacer (ITS) regions are examples of frequently used barcodes in distinct regions of the tree of life (see, e.g., [5–7]). As using barcodes for DNA-based species identification constitutes a routine analysis task, there exist several widely used taxonomic classification tools, such as SINTAX [8], IDTAXA [9], the RDP Naive Bayesian classifier (RDP) [10], and BayesANT [11]. These highly cited tools, deploy distinct algorithmic approaches to determine the species that are present in a sample.

There is one major design and one major quality criterion for any taxonomic classification tool: assign sequences quickly and correctly. Species identification accuracy is critical, as it typically constitutes the first step in biodiversity analyses.

Therefore, errors are likely to be propagated to downstream analyses and results. However, we are in the midst of the next generation sequencing data avalanche which is being further intensified by an increasing number of biodiversity field studies [12, 13]. The amount of data being generated has outpaced Moore's law for the last decade [14]. Hence, we need to perform barcoding sequence data analysis more efficiently. Otherwise, biodiversity research will be increasingly constrained by the computational resources available.

To alleviate this scalability challenge we introduce a novel tool, which we call **raxtax**, and demonstrate that it is at least as accurate as the widely-used existing tools SINTAX, IDTAXA, RDP, and BayesANT. Furthermore, we demonstrate the superior computational performance of **raxtax** in comparison to the competing tools. The respective speedups over the fastest competing tool range between 2.67x and 35.24x on the largest dataset we used.

raxtax achieves high accuracy in conjunction with computational efficiency via a *k*-mer based matching approach. That is, we formulate sequence similarity as follows: Compute

the expected number of matching k -mers between the reference sequence and a random sampling of the k -mers of a query sequence. The key insight is that if a query sequence is more similar to a reference sequence, the number of expected matching k -mers will be higher. With this reformulation of the problem we can derive closed analytical solutions that allow for computing the exact probability that a given reference sequence is (among) the best matches for a random sample of query sequence k -mers. **raxtax** then uses these probabilities to determine which reference database sequence constitutes the best match for a given query. Subsequently, we aggregate these probabilities at each taxonomic rank (clade) to obtain a taxonomic lineage assignment with per-rank confidence scores. Finally, we also use these per-rank confidence scores to compute uncertainty scores for each assignment of a query to a lineage.

The input to **raxtax** is a set of reference (DNA) sequences that have a taxonomic annotation and a set of anonymous query sequences. Given the reference sequences, **raxtax** computes the best-match probabilities for each query sequence, and reports the best matching lineages with their per-rank confidence scores and the uncertainty scores. Each of these quantities and their interpretations are discussed in Section 2. **raxtax** is available as open source code and pre-compiled binaries at <https://github.com/noahares/raxtax> under a CC-NC-BY-SA license.

2. Method

Given a sequence \mathcal{S} (consisting of characters from the set $\{\text{A, C, G, T}\}$), a k -mer is a sub-sequence $\mathcal{S}[i..i+k]$, $i \in [|\mathcal{S}| - k]$ of length k . The set of k -mers, Q , associated with \mathcal{S} includes all unique k -mers of \mathcal{S} . Here, we fix $k := 8$ to allow for some computational optimizations (see Section 3.1), but in principle the method can be adapted to any k .

Strictly matching *all* k -mers of each query sequence against *all* reference sequences is not only time and memory intensive, but also highly sensitive to sequencing errors. On the other hand, only matching a small random sample of k -mers does not constitute an appropriate solution either. In particular, if the reference sequences are highly similar and/or share a large fraction of k -mers, numerous repetitions with small random samples will be required to distinguish between plausible assignments and therefore increase runtimes. Instead, we use a combinatorial approach for selecting a random subset of k -mers from the query to match against the reference. This allows to obtain accurate results while being computationally efficient at the same time.

Assume that we are given the set of all k -mers Q which have been extracted from a query sequence and that we intend to match them against a set of reference k -mer sets $D = \{D_1, \dots, D_n\}$. To find the best matching D_i for a given Q , we need to identify the D_i which maximizes the expected number of matches from a random sampling of t k -mers from Q . We label this sample as $S_t(Q)$. Define P_i as the probability that the reference sequence D_i has the most k -mers in common with a random sampling of Q , or more formally $D_i \cap S_t(Q) \geq D_j \cap S_t(Q) \forall D_j \in D$. Our method for computing this probability is described in Sections 3.3 and 3.4.

Define the probability that a reference sequence D_i has m matching k -mers with $S_t(Q)$ as

$$p_i(m) := P(|D_i \cap S_t(Q)| = m), \quad (1)$$

which is a probability mass function (PMF). Using this definition, we can marginalize over the possible match sizes

indexed by l , and over the other reference sequences indexed by j ,

$$C_i(m) := \prod_{j \neq i} \left(\sum_{l \leq m} p_j(l) \right). \quad (2)$$

The probability that D_i is among the best matches, given a sample size t is

$$P_i := \sum_{m \leq t} p_i(m) C_i(m). \quad (3)$$

Additionally, we normalize the values in P via the $L1$ norm in order to compute *confidence (scores)*. This operation simplifies the subsequent confidence computation for a subset of D . As a result, the reported values are not, strictly speaking, probabilities. Instead, they report the confidence regarding the relative ranking of reference sequences for matching a query.

Given a clade B of the reference taxonomy, we define the confidence of B being among the best matches relative to other clades of the same rank as

$$L(B) := \sum_{D_i \in B} \frac{P_i}{\|P\|_1}. \quad (4)$$

To simplify the notation, we define $\mathcal{L}(D_i)$ as the lineage confidence vector for reference sequence D_i . $\mathcal{L}(D_i)$ is a sequence of $L(\cdot)$ values for the taxonomic lineage where A_i is a series of nested partitions (clades) of the reference sequences ($D_i = A_0 \subseteq \dots \subseteq A_i \subseteq \dots \subseteq D$).

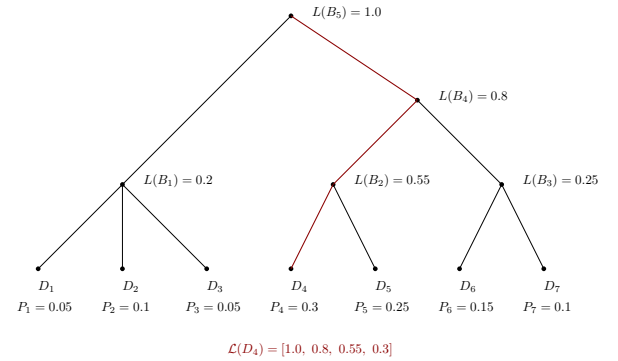


Fig. 1. A simple lineage showing how $L(\cdot)$ and $\mathcal{L}(\cdot)$ values are being accumulated. The contributors to $\mathcal{L}(D_4)$ are highlighted in dark red.

2.1. Uncertainty scores

The per-rank confidence values $L(\cdot)$ that we compute with **raxtax** will be biased by the taxonomic distribution of reference sequences in the database. Because the values at high-level ranks are the sum over all per-sequence values within those ranks, interpreting a confidence value of 0.5 requires knowledge about the relative frequency of that clade in the reference database. For instance, consider the case that one family represents 50% of the database. In this case, by chance alone, a substantial proportion of the total confidence score will be assigned to reference sequences in this over-represented family. Therefore, to better interpret the confidence values relative to the reference database properties, we report two additional uncertainty scores.

Let $\bar{P} := (\frac{1}{n}, \dots, \frac{1}{n})$ be the *expected* confidence vector for a sequence that is highly dissimilar (i.e., k -mer set intersections will be of approximately the same size) to all reference sequences. In analogy to using L for P values (Equation 4), we define \bar{L} as the *expected* confidence of obtaining a higher-level rank assignment based on \bar{P} . This means that the expected values of higher-level ranks represent the potential database bias. We will use these values to derive an uncertainty score for the global (*per-sequence*) and local (*per-rank*) assignment signals.

The *local assignment signal*

$$s_l(D_i) := \left\| \frac{\mathcal{L}(D_i)}{\|\mathcal{L}(D_i)\|_1} - \frac{\bar{\mathcal{L}}(D_i)}{\|\bar{\mathcal{L}}(D_i)\|_1} \right\|_2, \quad (5)$$

quantifies the uncertainty in $\mathcal{L}(D_i)$ as the Euclidean distance between the computed and expected per-rank confidence values (with normalization). Analogously, we define the *global assignment signal*

$$s_g := \|P - \bar{P}\|_2 \quad (6)$$

to quantify the reference sequence level confidence scores as the Euclidean distance between the computed and expected per-sequence confidence values.

3. Implementation

raxtax is written in Rust (compiled with version 1.76) and is parallelized over the query sequences using the **rayon** library [15]. In this section we describe the algorithmic techniques and data structures we use to optimize **raxtax**.

3.1. Calculating Intersection Sizes

To compute the match scores for all query-reference pairs, we need to compute the intersection of the two k -mer sets. A naïve implementation requires computing $\mathcal{O}(nm)$ intersections, where n is the number of query sequences, and m is the number of reference sequences. The best case run time for a sorted set intersection of sets A and B is $\mathcal{O}(\min(|A|, |B|))$ via a linear scan when $A \subseteq B$. Because computing intersection sizes accounts for at least half of the processing time of a query, it is therefore important to optimize them.

While there exist numerous fast set intersection algorithms [16–18], most pairs of k -mer sets satisfy $|A \cap B| \ll \min(|A|, |B|)$. Hence, it will be more efficient to ask which reference sequences contain a specific k -mer and store these results in a lookup table. This lookup table is computed once for all k -mers and reference sequences and is query-independent. It can therefore be saved for any analyses that use the same reference database. Given this lookup table, we simply perform a lookup of the k -mers in the query sequence to compute the intersection of a query-reference pair. Thereby, we reduce the work for one query-reference pair from $\mathcal{O}(\min(|A|, |B|))$ to $\mathcal{O}(|A \cap B|)$ where A and B are the respective k -mer sets.

Because we discard k -mers that include gaps and ambiguous characters, they can be represented in a memory-efficient manner by only using two bits per DNA character. By setting $k := 8$, we can thus uniquely store an 8-mer in a 16-bit unsigned integer (u16) by using its corresponding bit representation. While parsing the reference sequences, we create a lookup table that for each 8-mer (represented as a u16) holds a sorted list of reference sequences that contain it. When extracting the k -mers from a query sequence later-on, we can use this lookup table to rapidly identify those reference sequences that contain each

query sequence k -mer. This allows to efficiently create an array of intersection sizes with all reference sequences on demand.

3.2. Post-order Lineage Tree

The core of **raxtax** is a multi-furcating tree data structure that reflects the entire lineage tree of the reference sequence set D . For each query, we create a new array A of size $|D|$ to hold the confidence scores. The indices of A correspond to the leaves of the tree in post-order. Each inner node B of the tree also stores an integer pair (a, b) that contains the index interval of A that belongs to the rank associated with this node. After computing the confidence scores as described in Section 2 and storing them in A , we compute their prefix sum A_p . To subsequently determine the clade confidence score $L(B)$ (see Equation 4) for any clade B of the tree, we calculate it via $A_p[b] - A_p[a]$ (see Figure 2).

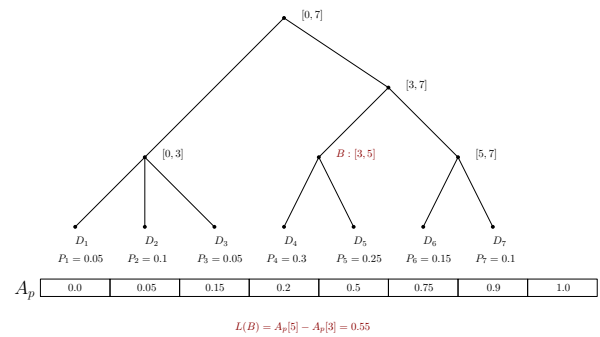


Fig. 2. A simple lineage showing the prefix sum A_p and inner nodes indices. An example for node B is highlighted in dark red.

We stop computing further $L(\cdot)$ values when the confidence of a node drops below a threshold of 0.005 to avoid an unnecessary evaluation of the entire tree. Thereby we only report relevant lineages.

3.3. The Probability of Exactly m Matching k -mers

We defined the probability mass function (PMF) $p_i(m)$ of a reference sequence D_i having exactly m out of t matches in Equation 1. If we expand this, we obtain

$$p_i(m) := \frac{\binom{|Q \cap D_i| + m - 1}{m} \binom{|D_i| - |Q \cap D_i| + (t - m) - 1}{t - m}}{\binom{|Q| + t - 1}{t}}. \quad (7)$$

Note that for a given query, the divisor is fixed and only depends on the size of the k -mer set of the query sequence and t , that is, the number of k -mers to be sampled. Also note that we need to calculate the numerator for each $m \leq t$ with m being the only variable. By utilizing the equivalence

$$\binom{n+1}{k+1} = \binom{n}{k} \frac{n+1}{k+1}, \quad (8)$$

we can iteratively compute both binomial coefficients in the numerator by only using a single multiplication and division per each value of m .

3.4. Caching PMF and CMF Values

We define

$$C(m) := \prod_{j \in [n]} \sum_{l \in [m]} p_j(l), \quad (9)$$

where the inner sum is the cumulative mass function (CMF) over p_j for a reference sequence D_j . Therefore, $C(m)$ is the product over all n reference sequence CMFs for some match count m . Given this definition, we can compute

$$P_i = \sum_{m \in [t]} p_i(m) \frac{C(m)}{c_i(m)} \quad (10)$$

via $2t$ additional operations. This decreases the time complexity for computing P_i from $\mathcal{O}(|D|t)$ to $\mathcal{O}(t + |D|)$. That is, the computation of best-match probabilities no longer scales with a factor of $|D|$. For all but the smallest reference databases, $t \ll |D|$, so this caching substantially accelerates the computation.

3.5. Improving Runtime for Repeated Execution with the same Reference Sequences

The Lineage Tree (cf. Section 3.2) and k -mer-to-sequence mapping (cf. Section 3.1) is independent of any queries and can therefore be shared between runs using the same reference sequences. To this end, we save the reference database in a binary file using `bincode` [19] which conducts encoding and decoding via a tiny binary serialization strategy. This file can initially be generated and then used for further queries at a later time. Often, this saves a substantial amount of time on reference databases that comprise a large amount of sequences and/or long sequences. In our experiments with the BOLD database [20], using the binary file created by `bincode` is two times faster than parsing the original input.

4. Experimental Evaluation

4.1. Cross validation benchmarks

To evaluate `raxtax` we performed a 10-fold cross validation across three datasets from respective databases (UNITE ITS [21], Greengenes 16S [22], BOLD COX1 [20]). In each dataset we only retained entries with complete taxonomic information and also removed duplicate sequences (Table 1).

Table 1. Databases

Database	UNITE	Greengenes	BOLD
Highest Taxonomic Rank	Fungi	Bacteria	Arthropoda
No. of sequences	47154	187329	1254059
No. of unique species	31479	629	136622

We executed four additional taxonomic assignment tools (SINTAX, RDP, IDTAXA, BayesANT) and calculated the F_1 score to assess the accuracy (TP = True Positives, MC = Missclassified, FN = False Negatives, FP = False Positives) at different taxonomic ranks.

$$Recall = \frac{TP}{TP + MC + FN} \quad (11)$$

$$Precision = \frac{TP}{TP + FP} \quad (12)$$

$$F_1 = 2 * \frac{Recall * Precision}{Recall + Precision} \quad (13)$$

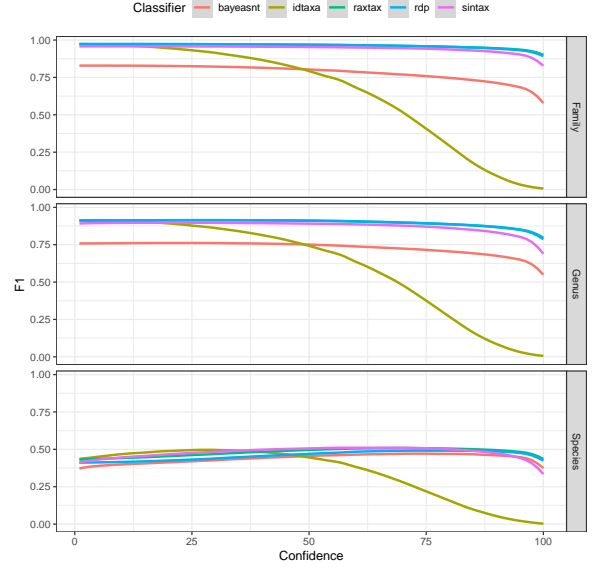


Fig. 3. F_1 scores (y-axis) for classification of UNITE sequences at the family, genus, and species level (top to bottom) where the reported confidence exceeds the confidence cut-off (x-axis).

Each tool provides a confidence score for the result of each query assignment and for each taxonomic rank that ranges between 0 and 100. We evaluated our algorithm against the competing tools by setting a continuous confidence cut-off thresholds that labels all results below the respective cut-off as "not classified". We then calculate the F_1 score for each confidence cut-off value.

Figure 3 shows that on the UNITE database, `raxtax`, RDP, and SINTAX perform equally well at all taxonomic levels. Further, `raxtax` and RDP are indistinguishable at the family and genus level. IDTAXA was developed to circumvent over-classification. Hence, once the confidence threshold approaches values of 0.25-0.5 the computed F_1 scores rapidly decline as a consequence of this conservative approach. BayesANT is only competitive at the species level.

For sequences from the BOLD database (Figure 4) only `raxtax` and SINTAX finished within the time and memory limits we set (see Subsection 4.2, so we compare only their F_1 scores. The `raxtax` F_1 score is consistently better at the family and genus level. At species level, the difference is statistically significant under the Wilcoxon signed-rank test [23] ($p = 1.272 \times 10^{-76}$). In general, both tools perform exceptionally well at classifying these sequences. However, as we show in the following sections, `raxtax` is 2.669 times faster than SINTAX for the comparatively large BOLD database and exhibits growing speedups as we simultaneously increase the number of query and reference sequences.

4.2. Performance Benchmarks

We measured the runtime and memory requirements of each tool for a single test (i.e., one out of the 10 cross validations) on each dataset (Table 2). We set a time limit of 48 hours –a common job time limit on clusters– to accommodate for trade-offs between accuracy and time requirements. As `raxtax` finished all analyses in under 90 minutes, this represents a generous time limit for the competing tools. The memory

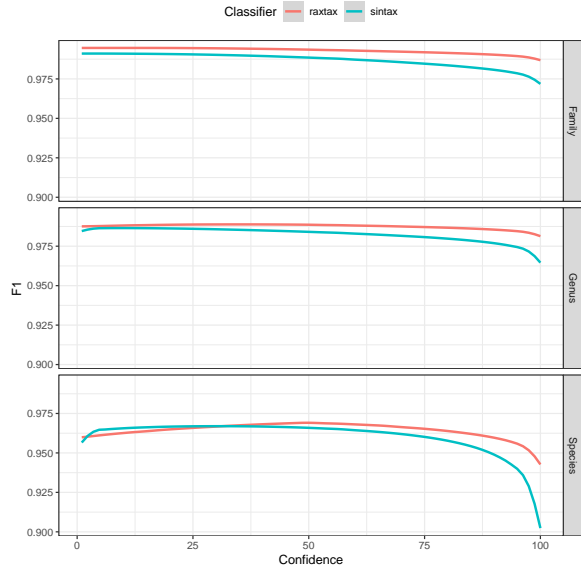


Fig. 4. F_1 scores (y-axis) for classification of BOLD sequences at the family, genus, and species level (top to bottom) where the reported confidence exceeds the confidence cut-off (x-axis).

Table 2. Time and memory requirements

Database	UNITE		Greengenes		BOLD	
Resource	T(s)	M(Gb)	T(s)	M(Gb)	T(s)	M(Gb)
raxtax	9	0.56	495	3.53	4904	12.41
SINTAX	9	0.11	346	1.16	13089	3.51
RDP	420	10.37	130	0.92	*	
IDTAXA	2004	3.46	5396	4.17	†	
BayesANT	1387	2.83	370	9.67	‡	

*exceeded memory limit (48Gb)

†exceeded time limit (48h)

‡R error (attempt to make table with $\geq 2^{31}$ elements)

limit of 48Gb is induced by the machine we used for these experiments. Every tool could use a maximum of 8 threads on an Intel(R) Core(TM) i7-7800X CPU @ 3.50GHz with 6 physical cores (12 logical CPUs). On the BOLD dataset, only **raxtax** and SINTAX completed within the memory and time limits. We observed that RDP and BayesANT require more resources as a function of the unique species number in the reference, while SINTAX and IDTAXA performance depends on the number of query and reference sequences. Datasets will continue to grow over time, both, in terms of the species diversity they cover, and the number of query as well as reference sequences they contain. Hence, we expect that the computational resource requirements of some of the tools we tested might prohibit their future deployment.

4.3. Snapshot benchmark

In order to validate our algorithm via a more realistic setting, we used two different BOLD database snapshots that were generated eleven months apart from each other. We taxonomically classified the sequences that were added during these eleven months by treating them as query sequences and subsequently compared the inferred annotation results with the respective “true” taxonomic annotation. Given the data volume

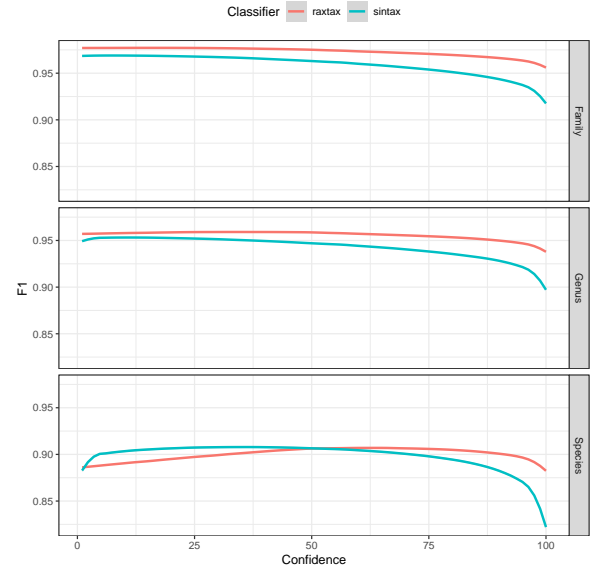


Fig. 5. F_1 scores (y-axis) for the classification of BOLD snapshots at the family, genus, and species level (top to bottom) where the reported confidence exceeds the confidence cut-off (x-axis).

of this analysis, only **raxtax** and SINTAX were able to terminate within the limits 48h and 48Gb limitation using 8 threads on the machine specified in Section 4.2. The F_1 scores are shown in Figure 5. As for the 10-fold cross validation on the BOLD database (Figure 4), **raxtax** and SINTAX are equally accurate. **raxtax** again outperforms SINTAX at the family and genus level. However, the difference at species level is not statistically significant in this test (Wilcoxon, $p = 0.7622$).

4.4. Time and Memory Scaling

We conducted our experiments on a different machine than for the preceding benchmarks (4.2 and 4.3). Here we used a 2-socket machine with 2x Intel Xeon Platinum 8260 CPUs @ 2.40GHz with 24 physical cores (48 threads) each. Figure 6 shows quadratic run time scaling for both tools when we simultaneously increase the number of reference *and* query sequences. The number of threads for both tools is again fixed to 8. **raxtax** clearly scales better when we increase the number of query and reference sequences. Going from 50,000 to 1,000,000 total sequences the speedup over SINTAX increases from 1.4 to 1.9.

In terms of memory requirements (Figure 7), both tools exhibit a strictly linear memory scaling as the total dataset size (no. of query and reference sequences) increases. The main memory requirements of **raxtax** (and presumably SINTAX as well) are dominated by the data structures that hold the reference database. Hence, this linear scaling is expected. SINTAX exhibits lower memory requirements and better scaling when we increase the total number of sequences. However, even for 1,000,000 sequences **raxtax**'s memory requirements remain below 10 GiB, so we argue that this is a favorable resource trade-off for using **raxtax** because of faster run times.

We also measure strong and weak parallel **raxtax** speedup for a varying number of threads on samples from the BOLD database. Figure 9 shows that as **raxtax** conducts increasing

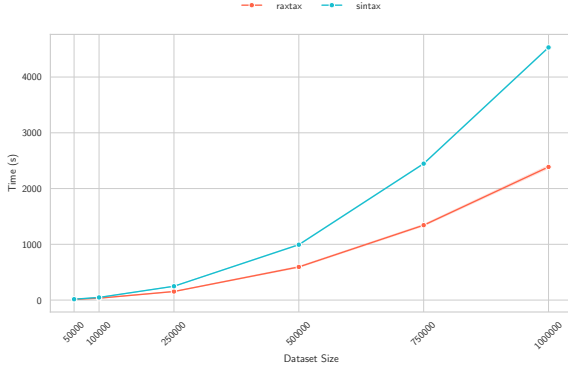


Fig. 6. Time (y-axis) for classification of BOLD samples of different sizes (x-axis). 90% of the sample is the reference, the remaining 10% are the queries (5 random samples per sample size).

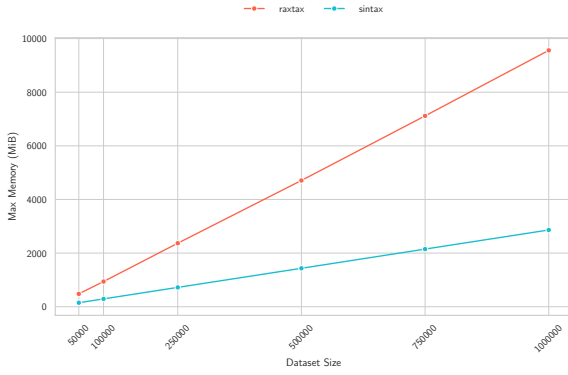


Fig. 7. Maximum memory usage (y-axis) for classification of BOLD samples of different sizes (x-axis). 90% of the sample is the reference, the remaining 10% are the queries (5 random samples per sample size).

work (i.e., additional queries) relative to the number of threads, we do not observe a decrease in parallel efficiency for up to 16 threads under a 16x increased workload compared to the baseline with 1 thread. Thereafter, we observe a rapid decline in parallel efficiency, with a parallel efficiency of 0.85 on 24 threads (24x workload) and 0.54 using 48 threads (48x workload), respectively. In the above experiments, we use explicit thread-pinning to avoid executing threads on the same physical core.

We obtain analogous results for the strong parallel scaling experiment (Figure 8) where **raxtax** again exhibits an almost optimal speedup of 15.1 up to 16 threads. The parallel efficiency loss for 24 threads is already substantial (speedup: 18.37). Thereafter, parallel efficiency continues to rapidly deteriorate with increasing number of threads.

It is currently unclear why the speedup already decreases for 24 threads, as distorting factors such as hyper-threading or data traffic using distributed shared memory interconnects have been excluded. The most likely explanation is cache contention.

5. Conclusion

We have presented a novel analytical approach for classifying unlabeled sequences based on k -mer matching and derive the

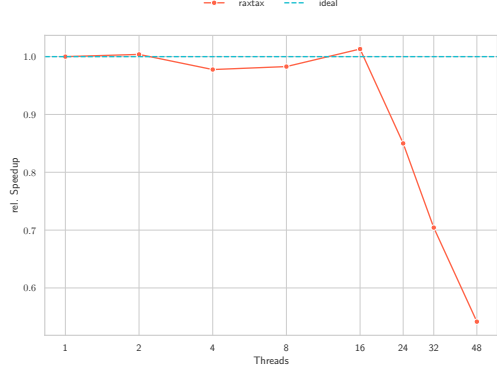


Fig. 8. Weak self-relative speedup (y-axis) over increasing thread numbers (x-axis). The reference database size is fixed at 250,000 sequences, that are randomly sampled three times. For each additional thread a further 2,000 query sequences are included.

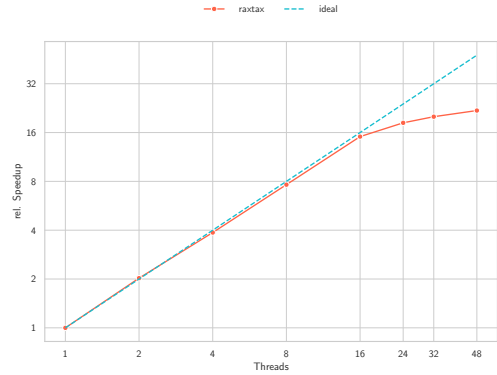


Fig. 9. Strong self-relative speedup (y-axis) over increasing thread numbers (x-axis). The reference database size is fixed at 225,000 sequences with 25,000 queries (90-10 split), and we randomly sample five times. Both axes are scaled by a base-2 logarithm for readability.

equations of our match scoring function for determining the best matching taxonomic lineage. Our method also introduces two additional uncertainty scores that are sensitive to an unbalanced distribution of ranks in the reference database and thereby provide users more context for drawing informed conclusions. We implemented this approach in **raxtax** as open source software. Further, we conducted a thorough code optimization to ensure that the tool is fast and efficient. An extensive evaluation of **raxtax** in conjunction with a comparison to existing tools demonstrates that we attain better or equivalent classification accuracy based on F_1 scores. Further, **raxtax** can handle the ever-increasing dataset sizes in taxonomic classification and can efficiently use all available computational resources on modern hardware. We argue that the increased memory requirements compared to SINTAX are an acceptable trade-off for the reduced run-times. In the future, we aim to deploy **raxtax** as part of a comprehensive meta-barcoding pipeline for real-world queries and adapt our approach in order to apply it beyond short barcoding sequences. Finally, we intend to investigate the design of a distributed memory parallelization.

6. Competing interests

No competing interests are declared.

7. Author contributions statement

B.B., N.W. and G.K. devised the method, N.W. implemented the code, G.K. and N.W. conceived and conducted the experiments, and analyzed the results. N.W., B.B., G.K., and A.S. wrote and reviewed the manuscript.

8. Acknowledgments

We thank Robert C. Edgar for sharing his insights about developing and validating SINTAX. This work is supported by the European Union (EU) under Grant Agreement No 101087081 (Comp-Biodiv-GR).



9. Data and Code Availability

This tool is available as source code and precompiled binaries at <https://github.com/noahares/raxtax> under a CC-NC-BY-SA license. The data from the analysis is available at **TODO**.

References

- Conrad L Schoch, Keith A Seifert, Sabine Huhndorf, Vincent Robert, John L Spouge, C André Levesque, Wen Chen, Fungal Barcoding Consortium, Fungal Barcoding Consortium Author List, Elena Bolchacova, et al. Nuclear ribosomal internal transcribed spacer (its) region as a universal dna barcode marker for fungi. *Proceedings of the national academy of Sciences*, 109(16):6241–6246, 2012.
- Robert D Ward, Tyler S Zemplak, Bronwyn H Innes, Peter R Last, and Paul DN Hebert. Dna barcoding australia’s fish species. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1462):1847–1857, 2005.
- Paul DN Hebert, Sujeevan Ratnasingham, and Jeremy R De Waard. Barcoding animal life: cytochrome c oxidase subunit 1 divergences among closely related species. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(suppl.1):S96–S99, 2003.
- Paul DN Hebert, Alina Cywinska, Shelley L Ball, and Jeremy R DeWaard. Biological identifications through dna barcodes. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(1512):313–321, 2003.
- Peter H Janssen. Identifying the dominant soil bacterial taxa in libraries of 16s rrna and 16s rrna genes. *Applied and environmental microbiology*, 72(3):1719–1728, 2006.
- Vasco Elbrecht, Pierre Taberlet, Tony Dejean, Alice Valentini, Philippe Usseglio-Polatera, Jean-Nicolas Beisel, Eric Coissac, Frederic Boyer, and Florian Leese. Testing the potential of a ribosomal 16s marker for dna metabarcoding of insects. *PeerJ*, 4:e1966, 2016.
- Rui-Heng Yang, Jin-He Su, Jun-Jun Shang, Ying-Ying Wu, Yan Li, Da-Peng Bao, and Yi-Jian Yao. Evaluation of the ribosomal dna internal transcribed spacer (its), specifically its1 and its2, for the analysis of fungal diversity by deep sequencing. *PloS one*, 13(10):e0206428, 2018.
- Robert C Edgar. Syntax: a simple non-bayesian taxonomy classifier for 16s and its sequences. *bioRxiv*, page 074161, 2016.
- Adithya Murali, Aniruddha Bhargava, and Erik S Wright. Idtaxa: a novel approach for accurate taxonomic classification of microbiome sequences. *Microbiome*, 6:1–14, 2018.
- Qiong Wang, George M Garrity, James M Tiedje, and James R Cole. Naive bayesian classifier for rapid assignment of rrna sequences into the new bacterial taxonomy. *Applied and environmental microbiology*, 73(16):5261–5267, 2007.
- Alessandro Zito, Tommaso Rigon, and David B Dunson. Inferring taxonomic placement from dna barcoding aiding in discovery of new taxa. *Methods in Ecology and Evolution*, 14(2):529–542, 2023.
- John La Salle, Kristen J Williams, and Craig Moritz. Biodiversity analysis in the digital era. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 371(1702):20150337, 2016.
- Xingjian Liu, Liang Zhang, and Song Hong. Global biodiversity research during 1900–2009: a bibliometric analysis. *Biodiversity and conservation*, 20:807–826, 2011.
- Kris A Wetterstrand. Dna sequencing costs: data. *National Human Genome Research Institute*, 2019.
- Rayon. <https://github.com/rayon-rs/rayon>.
- Jiyuan Zhang, Yi Lu, Daniele G. Spampinato, and Franz Franchetti. Fesia: A fast and simd-efficient set intersection approach on modern cpus. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, page 1465–1476, Dallas, TX, USA, April 2020. IEEE.
- Hiroshi Inoue, Moriyoshi Ohara, and Kenjiro Taura. Faster set intersection with simd instructions by reducing branch mispredictions. *Proceedings of the VLDB Endowment*, 8(3):293–304, November 2014.
- Benjamin Schlegel, Thomas Willhalm, and Wolfgang Lehner. Fast sorted-set intersection using simd instructions.
- Bincode. <https://github.com/bincode-org/bincode>.
- Sujeevan Ratnasingham and Paul D N Hebert. bold: The barcode of life data system (<http://www.barcodinglife.org>). *Mol Ecol Notes*, 7(3):355–364, May 2007.
- Kessy Abarenkov, R Henrik Nilsson, Karl-Henrik Larsson, Andy F S Taylor, Tom W May, Tobias Guldberg Frøslev, Julia Pawlowska, Björn Lindahl, Kadri Pöldmaa, Camille Truong, Duong Vu, Tsuyoshi Hosoya, Tuula Niskanen, Timo Piirmann, Filipp Ivanov, Allan Zirk, Marko Peterson, Tanya E Cheeke, Yui Ishigami, Arnold Tobias Jansson, Thomas Stjernegaard Jeppesen, Erik Kristiansson, Vladimir Mikryukov, Joseph T Miller, Ryoko Oono, Francisco J Ossandon, Joana Paupério, Irja Saar, Dmitry Schigel, Ave Suija, Leho Tedersoo, and Urmas Kõljalg. The UNITE database for molecular identification and taxonomic communication of fungi and other eukaryotes: sequences, taxa and classifications reconsidered. *Nucleic Acids Research*, 52(D1):D791–D797, 11 2023.
- Daniel McDonald, Yueyu Jiang, Metin Balaban, Kalen Cantrell, Qiyun Zhu, Antonio Gonzalez, James T. Morton, Giorgia Nicolaou, Donovan H. Parks, Søren M. Karst, Mads Albertsen, Philip Hugenholtz, Todd DeSantis, Se Jin Song, Andrew Bartko, Aki S. Havulinna, Pekka Jousilahti, Susan Cheng, Michael Inouye, Teemu Niiranen, Mohit Jain, Veikko Salomaa, Leo Lahti, Siavash Mirarab, and Rob Knight. Greengenes2 unifies microbial data in a single

- reference tree. *Nature Biotechnology*, 42(5):715–718, May 2024.
23. Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics: Methodology and distribution*, pages 196–202. Springer, 1992.