FastTree 2-Approximately Maximum-Likelihood Trees for Large Alignments

By Morgan N. Price, Paramvir S. Dehal, Adam P. Arkin

Presented by Binghui Cheng

Main focus of the paper

- Introduce FastTree-2 in details
  - It's a Maximum Likelihood optimization method
  - a tool for inferring ML trees for large alignments
  - It takes sequence alignment as input and output phylogeny trees
- Compare the performance of FastTree-2 with other methods
  - Performance is measured by
    - RF-distance
    - log-likelihood
    - running time
  - Methods mainly includes:
    - PhyML
    - RAxML

Introduction to FastTree-2

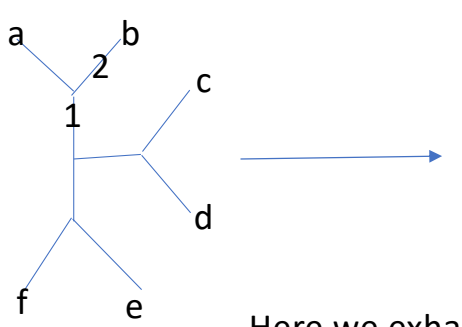
- First recall FastTree:
  - FastTree first of all is not a ML method, it's a minimum evolution method
  - It also takes sequence alignment as input and output phylogeny trees
  - The process in FastTree:
    - From the sequence alignment, we store the profiles for the internal nodes
    - Apply variant of neighbor joining to the profiles to compute a starting tree quickly
    - Then, it tries to improve the tree topology via nearest-neighbor interchanges (NNI)
  - Recall NNI, Near Neighbout Interchange, it can change tree topology (((A,B),C),D) to (((A,C),B),D) or (((A,D),C),B)
  - FastTree performs NNI to try to reduce the length of the tree topology
  - It is fast but not so accurate when comparing to ML methods (RAxML, PhyML)

Introduction to FastTree-2

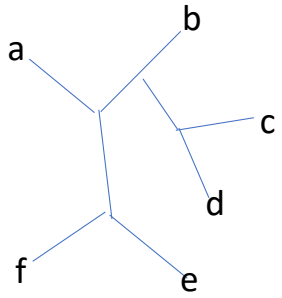
- Difference between FastTree-2 and FastTree:
  - Recall SPR, which is “stronger” than NNI since it allows pruning and regrafting for example from (((A,B),(C,D)),E) we cut C and reform (((A,(B,C)),D),E)
  - After computing the initial topology and refining it with NNI, FastTree-2 will first perform SPR and then ML NNI
- Linear SPR
  - linear SPR will only perform 2 rounds of SPR moves and therefore, for each node FastTree-2 will do an exhaustive search for moves up to length 10
- ML NNI
  - Computes the posterior distribution for each internal nodes

Linear SPR

- This step could be easily understand with visualization



Here we exhaustively search edges up to length 2 (w.r.t to internal node adjacent to c and d)



Then recursively search even further up to length 10 (assume b is attached to some other branches)

In a tree we have O(n) internal nodes and for each of them we only search up to length 10 at so, Linear SPR is at most O(n)

ML NNI

- From the previous steps, we have already generated the topology of the tree and optimized the likelihood of the topology approximately, using Linear SPR and original NNI
- This steps focus on the further optimize the topology carefully which will be time consuming
- we use NNI moves and Felsenstein's pruning algorithm to score the tree topology, if the corresponding NNI moves generate a tree with likelihood better than our current maximum likelihood tree, we replace current best topology with the topology after NNI moves

Testing FastTree-2

- FastTree-2 is tested on
  - Simulated protein alignments (derived from COG)
  - Biological dataset (16S ribosomal RNAs and on protein families from COG)
- Compared the speed and accuracy of FastTree-2 with PhyML 3.0 and RAxML 7 (they are the most popular ML methods)
  - The topology accuracy are measured by RF-distance and we compared it for all the methods if it is running the simulated datasets
  - For PhyML 3.0, we examine the local support values to measure the significance of the splits found by FastTree-2
  - For RAxML 7, we further tested them on biological datasets and compared the log-likelihood and the support values

Results

Table 1. Topological accuracy of trees inferred from simulated alignments.

	250	1,250	5,000	78,132
Method	**a.a.**	**a.a.**	**a.a.**	**nt.**
RAxML 7 (JTT+CAT, SPRs)	90.5%	88.4%	88.4%	–
PhyML 3.0 (JTT+Γ_4, SPRs)	89.9%	–	–	–
FastTree 2.0.0 (JTT+CAT or JC+CAT)	86.9%	83.7%	84.3%	92.1%
PhyML 3.0 (JTT+Γ_4, no SPRs)	86.0%	–	–	–
FastME 2.06 (log-corrected distances, SPRs)	80.5%	78.8%	77.0%	–
FastTree 2.0.0, no ML NNIs	80.4%	78.3%	76.6%	91.4%
BIONJ (ML distances)	77.7%	73.7%	73.1%	–
Parsimony (RAxML)	76.8%	76.5%	69.4%	–
Neighbor joining (log-corrected distances)	76.0%	72.6%	71.6%	66.1%
Clearcut (log-corrected distances)	75.5%	72.3%	71.5%	58.1%

For alignments with 5,000 sequences, we used RAxML 7.2.1 with fast convergence; for smaller alignments we used RAxML 7.0.4. An earlier version of PhyML 3 took up to 4 days for individual simulations with 1,250 sequences, even without Γ_4, so we did not try to run PhyML 3 with Γ_4 on the larger simulations.

doi:10.1371/journal.pone.0009490.t001

Fig1 is the topological accuracy comparison
On Simulate datasets

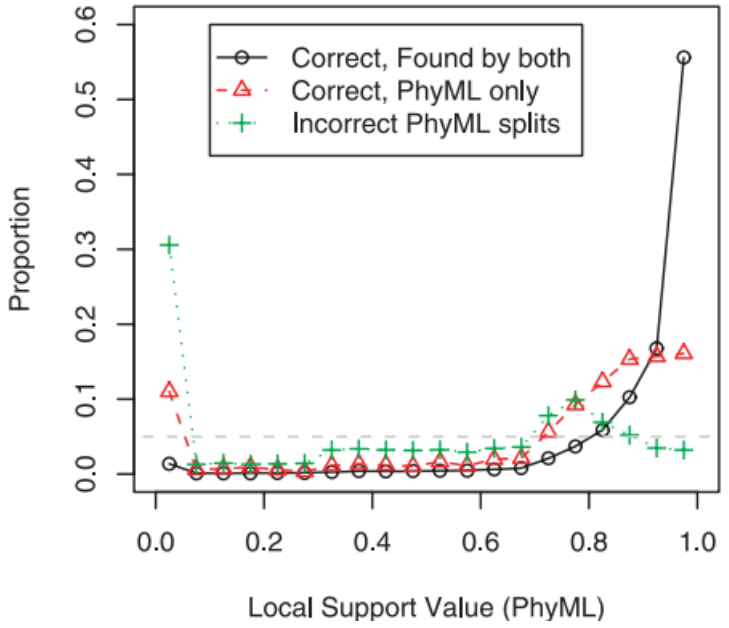
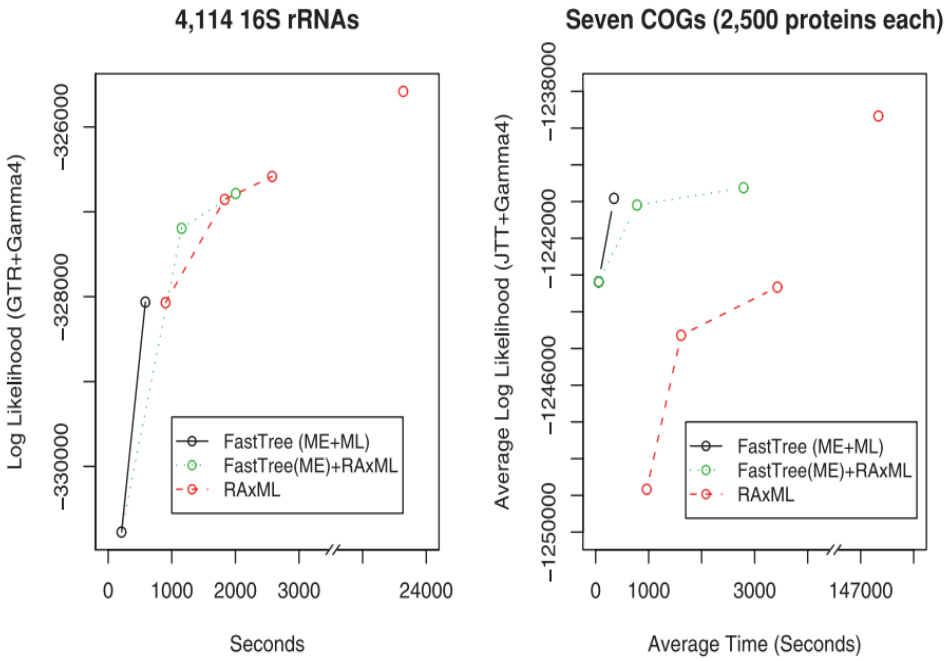


Fig2 is the local support value difference
between FastTree-2 and PhyML

Results



Comparison between the improvement vs time in log-likelihood between RAxML and FastTree-2

Table 3. Comparison of RAxML and FastTree's log likelihoods, and the agreement of FastTree with RAxML's well-supported splits, for large genuine alignments.

	16S rRNA	16S rRNA	7 COGs
Number of sequences	4,114	6,718	2,500
RAxML 7's Log Likelihood	−325,581	−481,259	−1,238,666
FastTree 2's Log Likelihood	−328,062	−493,841	−1,240,916
Difference	2,481	12,582	2,251
Well-supported RAxML splits (bootstrap ≥ 0.9)			
Total in RAxML tree	851	1,124	–
Found by FastTree	837	1,075	–
Weakly-supported RAxML splits (bootstrap 0.8–0.9)			
Total in RAxML tree	265	419	–
Found by FastTree	250	365	–
Locally-supported RAxML splits (SH ≥ 0.95)			
Total in RAxML tree	1,336	1,927	1,018
Found by FastTree	1,033	1,319	889

We ran RAxML with the fast convergence option. All values for COGs are averages over seven families. Log likelihoods for all topologies were computed with RAxML using Γ_4 and GTR or JTT. Global bootstrap values are from using the standard bootstrap with RAxML 7.0.4 (from [11]). SH-like local support values for RAxML's topology were computed with FastTree 2, the CAT approximation, and GTR or JTT.

doi:10.1371/journal.pone.0009490.t003

Results

Table 4. Running time and memory usage on genuine alignments.

	Distinct		FastTree 2.0.0			RAxML 7	PhyML 3
Alignment	**Sequences**	**Positions**	**Model**	**Hours**	**GB**	**Hours**	**Hours**
16S rRNA, subsets	500	1,287 nt.	GTR	0.02	–	2.2	2.9
COGs, subsets	500	65–1,009 a.a.	JTT	0.02	–	5.2	7.2
COGs, subsets	2,500	197–384 a.a.	JTT	0.11	–	61	–
Efflux permeases	8,362	394 a.a.	JTT	0.25	0.35	197	>1,200
16S rRNAs, families	15,011	1,287 nt.	GTR	0.66	0.56	64	>2,000
ABC transporters	39,092	214 a.a.	JTT	1.02	0.96	–	–
16S rRNAs, all	237,882	1,287 nt.	JC	21.8	5.8	–	–

All runs used a single thread of execution. All runs accounted for variable rates across sites, using CAT for RAxML 7 and FastTree 2 or Γ_4 for PhyML 3. All FastTree runs include local SH-like supports and all RAxML runs include branch lengths under Γ_4. RAxML and PhyML were run without support values (no bootstrap). For random subsets of 500 16S rRNAs or for COGs, we show average running times. For alignments with over 1,000 sequences, we used RAxML 7.2.1's fast convergence option.
doi:10.1371/journal.pone.0009490.t004

This shows that the running time of FastTree-2 compare to other ML methods is extremely fast (about 100 times faster)

Conclusion

- FastTree-2 improved the original FastTree algorithm via adding 2 new phrases (Linear SPR & ML NNI)
- FastTree-2 maintained its fast speed even after adding 2 new phrases
- Compared to other ML algorithms (RAxML, PhyML), FastTree-2 is very close in topological accuracy, support values, log-likelihood and many other things but is slightly less good than those ML algorithms

Connection to my course project

- My course project aims to compare the performance of three ML methods: iqtree, RAxML and fastTree-2 on simulated sequence alignments with different indel rates and p-distance
- I will simulate sequences with same topology but will change its branches length and indel rates in INDELible to get sequences of varying features
- Then run all 3 methods on the sequences to compare the results
- And so the results of comparison between RAxML and fastTree-2 in this paper and its experiment set up provide important references for my project