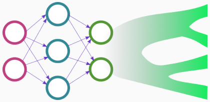
Deep end-to-end likelihood-free inference of phylogenetic trees

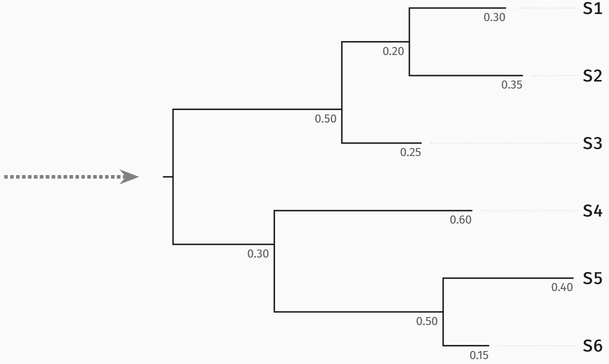


Luc Blassel, Nicolas Lartillot, Bastien Boussau, Laurent Jacob

MLCB - September 11th, 2025

Context - Phylogenetic inference

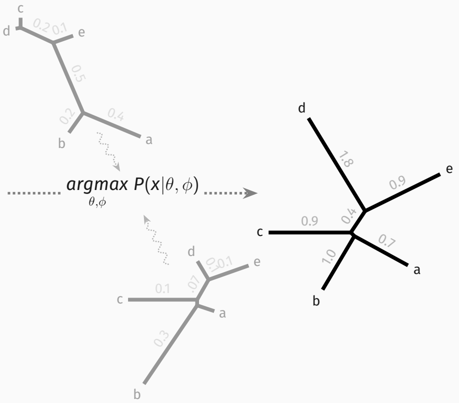
```
S1 TQPRLPTC
S2 TQPSVPKC
S3 TGVPVPAC
S4 TLGRSPSC
S5 TQPRAKTC
S6 TQPRIPAC
```



Goal: describe **evolutionary-history** of MSA

Context - Likelihood-based tree reconstruction

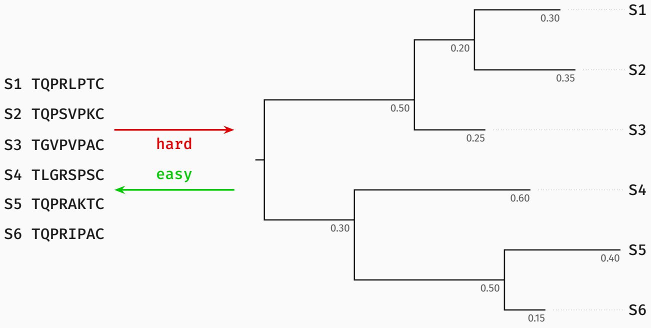
a	TQPRLPTC
b	TQPSVPKC
c	TGVPVPAC
d	TLGRSPSC
e	TQPRAKTC



- **accurate** but **slow**
- $P(x|\theta, \phi)$ must be **computable**

x : MSA, $\theta = (\tau, \ell)$: Phylogenetic tree, ϕ : Evolution model Felsenstein 1993; Kleinman et al. 2010

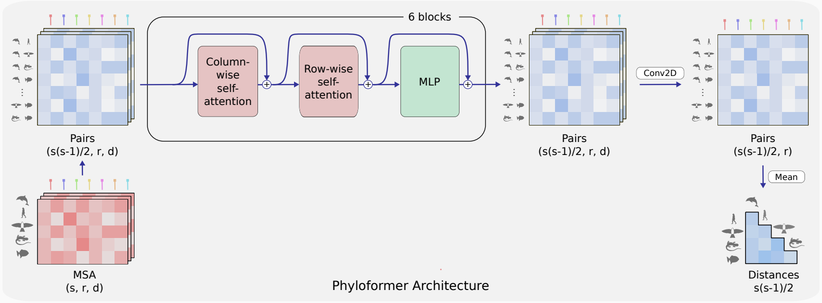
Motivation - Likelihood-free inference



- We can simulate many[1] (tree, MSA) pairs
- Can we **learn** the mapping **from MSA to tree**?

[1] pretty much practically ∞

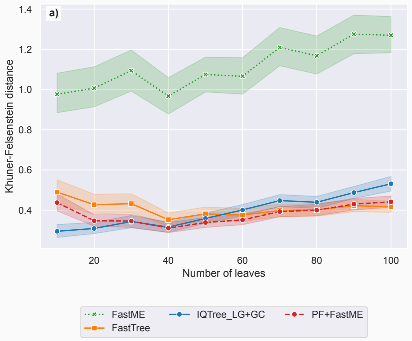
Related Work - Phyloformer, our first approach

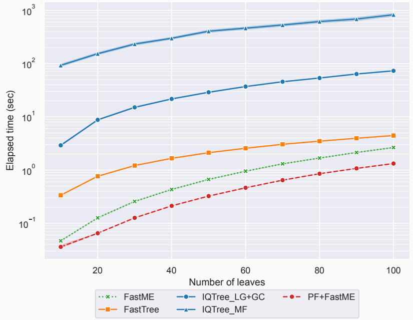


- Input an **MSA**, get a **Distance matrix**
- Feed Distance matrix to **FastME** to get **tree**

Nesterenko et al. 2025; Lefort et al. 2015

Related Work - Phyloformer is good!





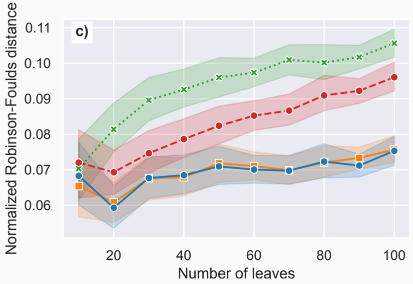
Tree inference accuracy (KF)

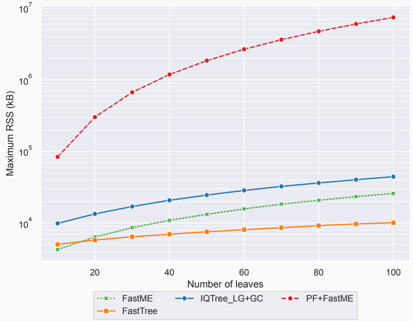
Runtime

- Fairly **competitive** even on simple LG+GC model
- **Fast** because we use GPUs [1]

Nesterenko et al. 2025, [1]🙏 Jean-Zay

Related Work - But also sometimes less good...





Topological accuracy (RF)

Memory usage

- **Gap** between PF and **ML methods**
- PF is **by far** the most **memory intensive**

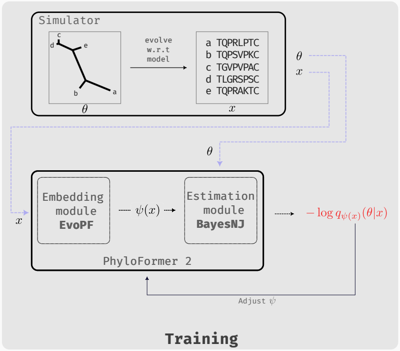
How to do phylogenetic inference end-to-end ?

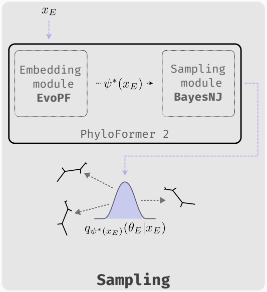
Methods - Neural Posterior Estimation (NPE)

- Given a **probabilistic model** $p(x|\theta)$ with some prior $p(\theta)$
- We want to **estimate the posterior**: $p(\theta|x)$
- We build $q_{\psi}(\theta|x)$ a **family** of distributions **parametrized** by ψ (our NN)
- We find $q_{\psi^*} = \underset{\psi}{\operatorname{argmin}} \mathbb{E}_{p(x)}[KL(q_{\psi}(\theta|x)||p(\theta|x)]$
- In practice we **maximize** $\mathbb{E}_{p(x,\theta)}[\log q_{\psi(x)}(\theta|x)]$ by **sampling** from $p(x,\theta)$

x : MSA, $\theta = (\tau, \ell)$: Phylogenetic tree, $\psi(x)$: NN applied to x

Methods - How do we do NPE?





- During **training** find $\psi^* = \underset{\psi}{\operatorname{argmin}} \; -\sum_i \log q_{\psi(x_i)}(\theta_i|x_i)$
- At **inference** time **sample** from: $q_{\psi^*(x_E)}(\theta_E|x_E)$ [1]

[1]WIP: so for now only point-estimation

Methods - The EvoPF module, intro

the EvoPF module is an **adaptation** of the **EvoFormer** module from **AlphaFold2**. The tasks are **transpositions** of each other:

given input MSA ($n \times r$)

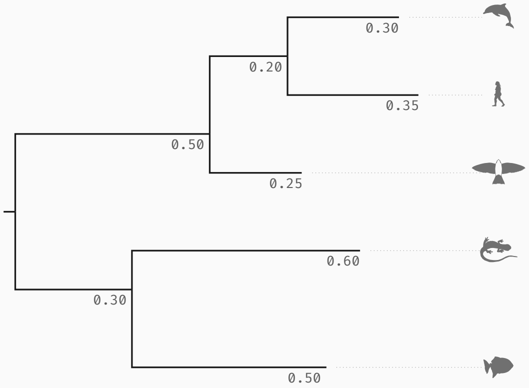
EvoFormer	represent $r \times r$ relationships between sites
EvoPF	represent $n \times n$ relationships between sequences

More expressive than MSA transformer
More lightweight than PF

Jumper et al. 2021; Rao et al. 2021

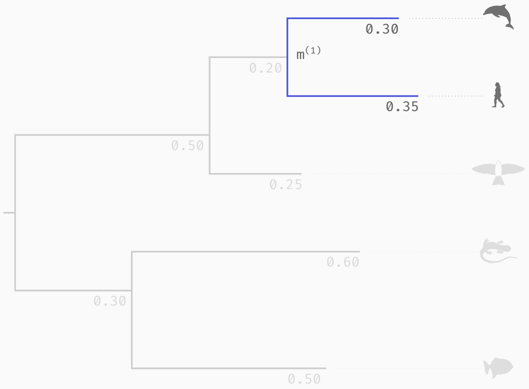
Methods - A tree is a series of merges

We want to describe the following tree:



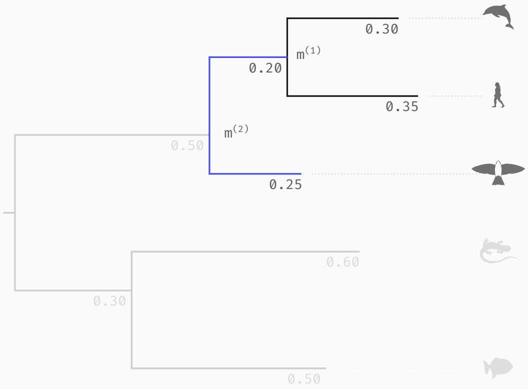
Methods - A tree is a series of merges

Iteratively merge shortest cherry:



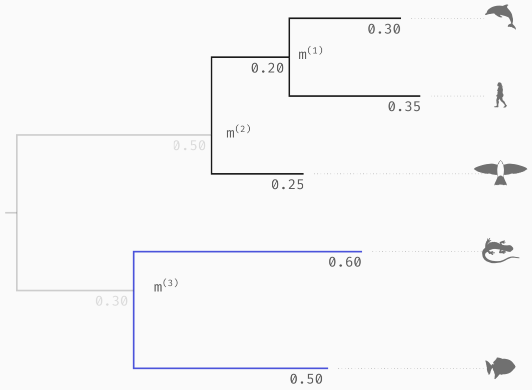
Methods - A tree is a series of merges

Iteratively merge shortest cherry:



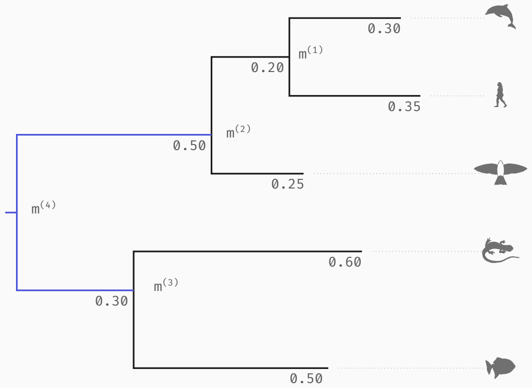
Methods - A tree is a series of merges

Iteratively merge shortest cherry:



Methods - A tree is a series of merges

Iteratively merge shortest cherry:



$$\tau = \{m^{(1)}, m^{(2)}, m^{(3)}, m^{(4)}\}$$

Methods - the BayesNJ module

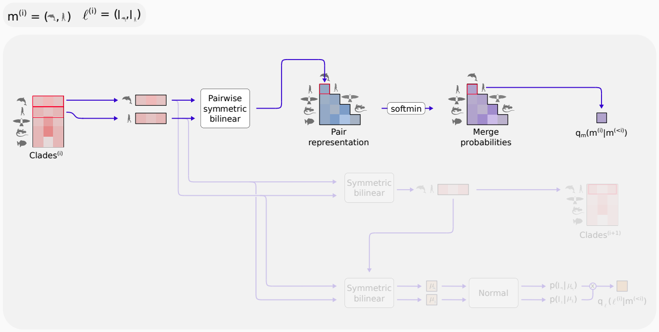
- **Tree** is an **ordered set** of merges: $\theta : \{m^{(1)}, \dots, m^{(N-1)}\}$
- We **factorize** $q_{\psi(x)}(\theta|x)$ as the product of successive merge probabilities:

$$q_{\psi(x)}(\theta|x) = \prod_{k=1}^{N-1} q_m(m^{(k)}|m^{(<k)}) q_\ell(\ell^{(k)}|m^{(\leq k)})$$

- **Merge** probabilities have **2 components**:

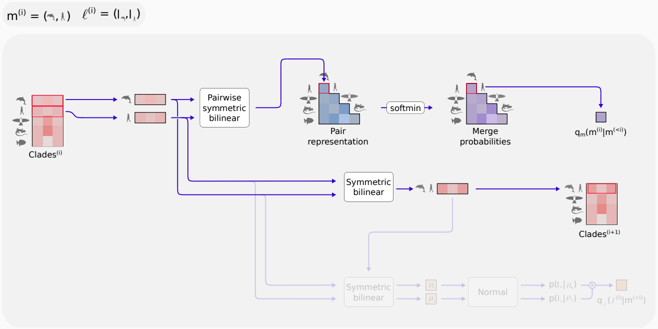
topological:	$q_m(m^{(k)}|m^{(<k)})$
branch-length:	$q_\ell(\ell^{(k)}|m^{(\leq k)})$

Methods - BayesNJ, evaluating topological probabilities



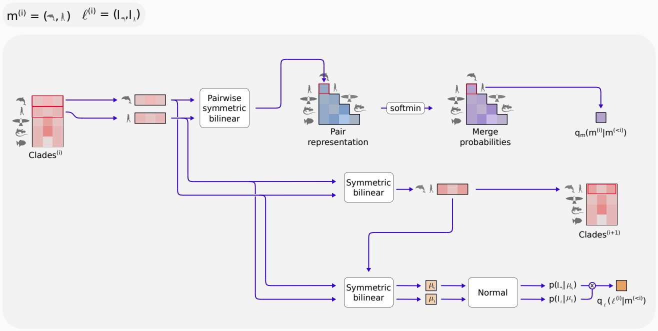
Compute **merge probability**

Methods - BayesNJ, evaluating topological probabilities



Update clade **representation** for next merge

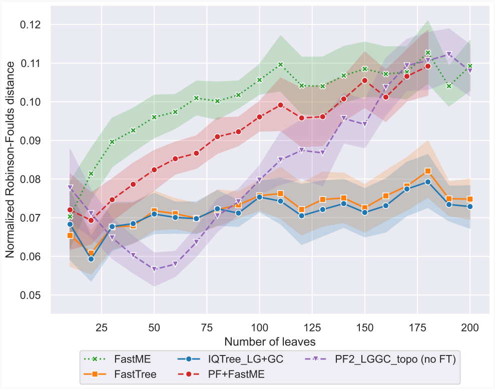
Methods - BayesNJ, evaluating topological probabilities



Compute **branch-length** probabilities

Does it work ?

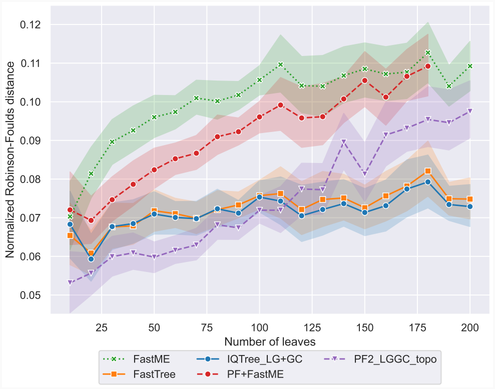
Results - Training topology only



- **overfitting** on tree-size is an **issue**

Same train set as PF1 paper: $\approx 170k$ 50 seq LG+GC MSAs on rescaled BD trees

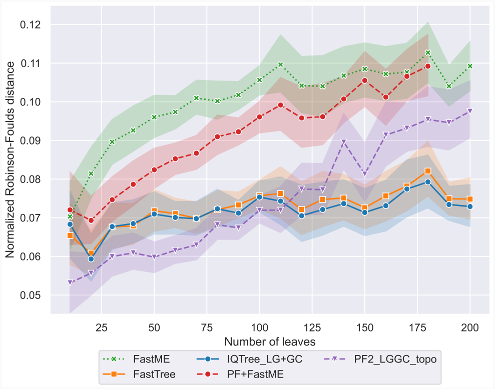
Results - Training topology only



- **overfitting** on tree-size is an **issue**
- **Fine tuning** helps

Same train set as PF1 paper: ≈ 170k 50 seq LG+GC MSAs on rescaled BD trees

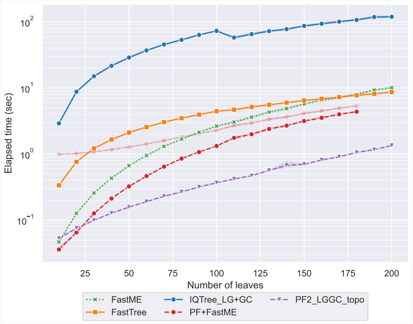
Results - Training topology only



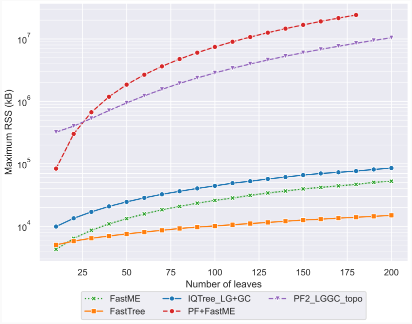
- **overfitting** on tree-size is an **issue**
- **Fine tuning** helps
- We **beat ML** methods in certain cases
- Marked **improvement** w.r.t **Phyloformer**

Same train set as PF1 paper: $\approx 170k$ 50 seq LG+GC MSAs on rescaled BD trees

Results - Scalability



Execution time



Memory usage[1]

[1] With 2× bigger sequence, and 4× bigger pair embeddings...

Perspectives - Short term challenges

This is very much still a **work in progress...**

- Training on **more complex** data *(e.g. indels)* **increases** length-**overfitting**
- Learning **topology** and **branch-lengths** is also **challenging**
- How can we move **away** from **point-estimation** ?
- We might need to **adjust** our **priors** to compare with MCMC tools

Conclusion

Takeaways

- **Topologically** we manage to **beat** ML-methods[1] on LG
- While being **more scalable** than PF1
- Still needs some **work** for a fully **end-to-end** phylogenetic **inference** tool

What next ?

- Can we do **better** where computing $p(\theta|x)$ is **difficult** or **intractable**? *(e.g. Potts, epistatis, selection, ...)*
- **Confident** this can **work** given our experience with **PF**

Prillo et al. 2023; Duchemin et al. 2023; Latrille et al. 2021

[1] Yay!

Thanks to:

- Luca Nesterenko
- Laurent Jacob
- Bastien Boussau
- Nicolas Lartillot
- Philippe Veber
- Vincent Garot
- Amélie Leroy
- Anybody that listened to me!

Special thanks to Jean-Zay for all the GPUs!

References

Dong, J. et al. (2024). ***Flex Attention: A Programming Model for Generating Optimized Attention Kernels.***

Duchemin, L. et al. (2023). ***Evaluation of methods to detect shifts in directional selection at the genome scale.*** In: *Molecular Biology and Evolution* 40.2, msac247.

Felsenstein, J. (1993). ***PHYLIP (phylogeny inference package), version 3.5 c.*** Joseph Felsenstein.

Jumper, J. et al. (2021). ***Highly accurate protein structure prediction with AlphaFold.*** In: *Nature* 596.7873, pp. 583–589.

Kleinman, C. L. et al. (2010). ***Statistical Potentials for Improved Structurally Constrained Evolutionary Models.*** In: *Molecular Biology and Evolution* 27.7, pp. 1546–1560.

Latrille, T. et al. (2021). ***Inferring Long-Term Effective Population Size with Mutation–Selection Models.*** In: *Molecular Biology and Evolution* 38.10, pp. 4573–4587.

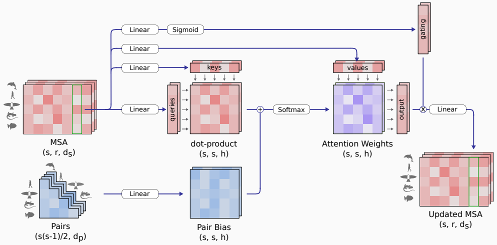
Lefort, V. et al. (2015). ***FastME 2.0: a comprehensive, accurate, and fast distance-based phylogeny inference program.*** In: *Molecular biology and evolution* 32.10, pp. 2798–2800.

Nesterenko, L. et al. (2025). ***Phyloformer: Fast, Accurate, and Versatile Phylogenetic Reconstruction with Deep Neural Networks*.** In: *Molecular Biology and Evolution* 42.4, msaf051.

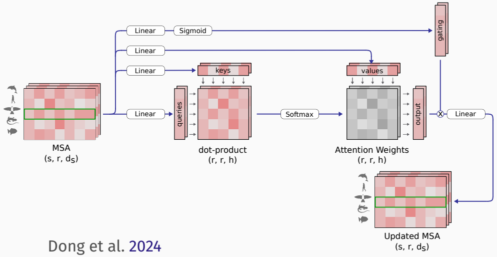
Prillo, S. et al. (2023). ***CherryML: scalable maximum likelihood estimation of phylogenetic models*.** In: *Nature methods* 20.8, pp. 1232–1236.

Rao, R. M. et al. (2021). ***MSA Transformer*.** In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by M. Meila and T. Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 8844–8856.

Supp. Methods - EvoPF, the MSA stack



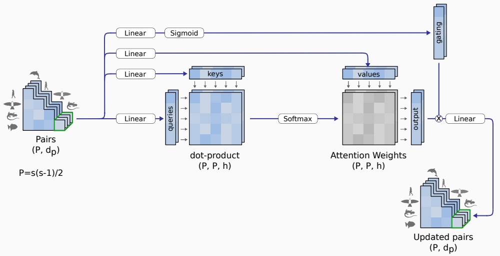
Column-wise attention with pair-bias



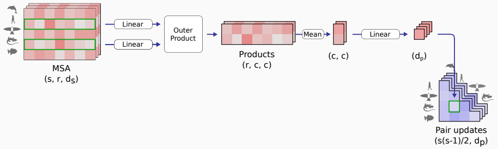
Row-wise attention

Dong et al. 2024

Sup. Methods - EvoPF, the pair stack



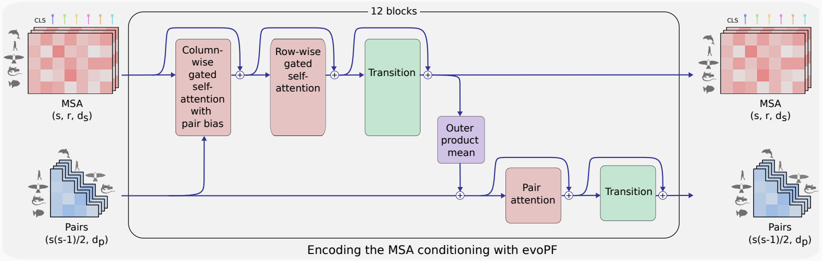
Pair attention



Outer product mean

Dong et al. 2024

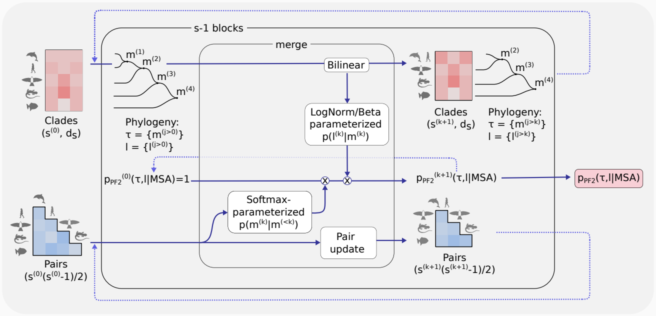
Sup. Methods - The EvoPF module, details



- Input an **MSA** and get:
  sequence embedding $\{s_i\}$
  sequence-pair embeddings $\{z_{ij}\}$
- **Both** embedding-types used to **update each-other**

Figure inspired by Jumper et al. 2021

Sup. Methods - BayesNJ evaluation mode



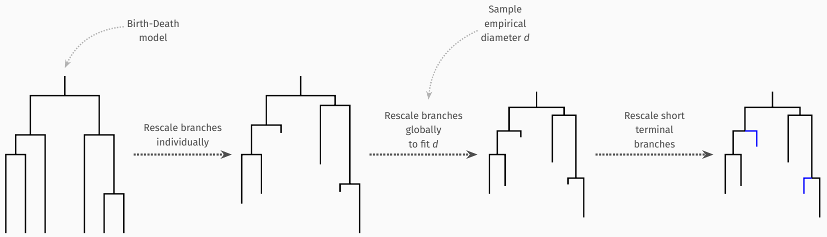
Sup. Methods - Ensuring the merge order is unique

Ensuring a **unique order** on merges ensures that we **define a distribution**. It also keeps **training** and **sampling** comparable [1]

- On a given tree τ always **merge** the **shortest** available **cherry**
- When **sampling**, add **constraints**:
  1. Start with a $N \times N$ constraints matrix $M_{ij} = 0$
  2. At iteration k sample merge $m^{(k)} = (i, j)$ and cherry length $s^{(k)} = M_{ij} + X$
  3. **Update constraints** for cherries **available** when sampling $m^{(k)}$: $M'_{ij} = max(M_{ij}, s^{(k)}) \quad M'_{ui} = 0$
- During evaluation compute $p_{PF2}(s^{(k)} - M_{ij} | m^{(\leq k)})$

[1] Which is not the same if we use the NJ merge order

Sup. Methods - Tree simulation


Birth-Death model
Rescale branches individually
Sample empirical diameter *d*
Rescale branches globally to fit *d*
Rescale short terminal branches