Beyond Viterbi: A Maximum Expected Accuracy Approach to Pairwise Alignment

Lawrence Granda[1], **Joyce Shen**[2], **Soham Goswami**[3], **and Joshua Ochalek**[4]

[1]Computer Science, Junior, lg626
[2]Computer Science, Junior, js3696
[3]Computer Science, Sophomore, sbg226
[4]Computer Science and Italian, Junior, jo447

ABSTRACT

Sequence alignment stands as the computational bedrock of modern biology, serving as the prerequisite for phylogenetic reconstruction, functional annotation, and structural modeling. For decades, the Viterbi algorithm has served as the *de facto* standard for decoding Pair Hidden Markov Models (Pair-HMMs), efficiently identifying the single maximum a posteriori (MAP) alignment path. However, this approach fundamentally fails to account for the stochastic uncertainty inherent in evolutionary analysis, a limitation that is particularly acute in the alignment of non-coding RNAs (ncRNAs). In these molecules, the conservation of secondary structure often supersedes primary sequence identity, creating a "twilight zone" of homology characterized by diffuse probability landscapes where the single best path represents a negligible fraction of the total probability mass. In this comprehensive report, we present the theoretical derivation, implementation, and rigorous evaluation of Maximum Expected Accuracy (MEA) decoding for pairwise RNA alignment. By decoupling the probabilistic model from the decision rule, we demonstrate that MEA provides a statistically rigorous framework for minimizing the Hamming loss of an alignment, contrasting sharply with the 0/1 loss minimized by Viterbi. We implement a three-state Pair-HMM trained on curated Rfam data and systematically evaluate four distinct posterior weighting schemes: Power, Threshold, Log-Odds, and ProbCons-style weighting. Our results, based on a benchmark of 547 pairwise alignments, indicate that MEA decoding consistently outperforms Viterbi in terms of F1 score and column identity. We further demonstrate that the γ parameter serves as a precise, tunable control for the sensitivity-specificity trade-off, allowing researchers to optimize alignments for specific downstream applications. This work establishes MEA not merely as an alternative, but as a superior paradigm for RNA sequence analysis that robustly integrates over evolutionary uncertainty.

Keywords: Pairwise sequence alignment, Hidden Markov Model (HMM), Maximum Expected Accuracy (MEA), Viterbi algorithm, Forward-backward algorithm, RNA sequence analysis, Posterior probability

Project type: Reimplementation

Project repository: https://github.com/joucebox/cs4775-final-project

AI Use Attribution Statements:

1. Understanding the project and the literature
    - **Tool:** Gemini (Google)
    - **Prompts Used:**
        - *Access [https://github.com/joucebox/cs4775-final-project] and read the readme and the src code. Explain what this project does.*
        - (Uploaded PDFs of the CentroidAlign [1] and ProbCons [2] papers) *Read the two papers cited* and *explain what this project does* in the context of the provided files.
    - **Influence on Work:** The AI summarized the GitHub repository structure and code functionality, explaining the core differences between the Viterbi and Maximum Expected Accuracy (MEA) algorithms. It analyzed the uploaded papers to clarify how the project's weighting schemes (Power, Threshold, ProbCons, Log-odds) derive from the γ-centroid estimator in *CentroidAlign* [1] and the probabilistic consistency transformation in *ProbCons* [2]. We used this information to help us write the literature review section of the report.
2. Verifying the derivation of the MAP (maximum a posteriori) alignment equation

- **Tool:** Gemini (Google)
- **Prompt Used:**
  - *Is this true? The maximum a posteriori (MAP) alignment is given by* $Z^\star = \arg\max_{Z \in \mathcal{Z}(X,Y)} P(Z \mid X, Y) = \arg\max_{Z \in \mathcal{Z}(X,Y)} \log P(X, Y, Z)$.
- **Influence on Work:** The AI confirmed that each step in the MAP alignment derivation was mathematically correct, including the use of Bayes' rule and the application of the logarithm's monotonicity. The AI further explained why the denominator $P(X, Y)$ can be dropped when searching for the maximizer. This assurance helped us finalize and clearly explain the Viterbi algorithm equations in the report.

3. Clarification of Bayesian estimators for alignment

- **Tool:** ChatGPT (GPT-5.2 Thinking)
- **Prompt Used:**
  - *The distinction between Viterbi and MEA is rooted in Bayesian decision theory. Write the different estimators for Viterbi and MEA.*
- **Influence on Work:** ChatGPT helped draft the mathematical distinction between the Viterbi estimator (MAP estimator), and the MEA estimator (which selects the alignment maximizing the expected number of correctly aligned residue pairs under the posterior). It provided clear Bayesian decision theory equations for each. We used this information to help us write the theoretical framework section of the report.

4. Verification and revision of Viterbi and MEA report sections

- **Tool:** ChatGPT (GPT-5.2 Thinking)
- **Prompts Used:**
  - *Check whether my LaTeX write-up of Viterbi/MAP decoding for a 3-state Pair-HMM is correct, and suggest minimal corrections.*
  - *Check whether my Bayesian decision-theory explanation of Viterbi vs MEA is correct, and suggest minimal corrections if needed.*
  - *Check whether my MEA section (posterior match probabilities via forward—backward, weighted objective with* γ*, and DP reconstruction) is correct. Point out errors if any.*
- **Influence on Work:** ChatGPT identified technical issues (e.g., MAP vs joint-probability wording, DP base-case/loop-range handling, and a threshold-weighting inconsistency), suggested concise clarifications, such as explicitly stating some assumptions, and suggested minimal edits to improve readability. We used these corrections and improvements to help us develop the report.

1 Introduction

1.1 Background and Motivation

The alignment of biological sequencesDeoxyribonucleic Acid (DNA), Ribonucleic Acid (RNA), and proteinsconstitutes the foundational stratum of modern bioinformatics. It provides the essential mapping required to infer evolutionary history, identify functional motifs, and predict macromolecular structure. Since the seminal work of Needleman and Wunsch in 1970 [3], the field has been dominated by dynamic programming algorithms that seek to optimize a specific scoring objective. The introduction of probabilistic modeling, particularly Pair-Hidden Markov Models (Pair-HMMs), transformed this landscape by providing a rigorous statistical framework for alignment, replacing ad hoc scoring matrices with learned transition and emission probabilities.

Within this probabilistic paradigm, the Viterbi algorithm has served as the standard for decoding, efficiently computing the single most probable alignment path (the Maximum A Posteriori, or MAP, estimate). However, the supremacy of the Viterbi path is increasingly challenged, particularly in the analysis of divergent sequences where the signal-to-noise ratio is low. This challenge arises from a fundamental epistemological limitation: the Viterbi algorithm assumes that the "best" alignment is the single path with the highest posterior probability. In the "twilight zone" of sequence homologywhere sequence identity drops below 30%the probability landscape becomes diffuse and rugged. Here, the single best path often represents a negligible fraction of the total probability mass, and alternative suboptimal paths may collectively harbor the true biological signal.

1.2 The Biological Imperative: RNA Structure and Evolution

To understand the necessity of MEA, one must first appreciate the biological context of the problem, specifically the alignment of Ribonucleic Acid (RNA). Unlike proteins, where the primary sequence (the string of amino acids) is the dominant determinant of folding and function, RNA function is dictated primarily by its secondary structurethe intricate patterns of Watson-Crick (G-C, A-U) and wobble (G-U) base pairs that form stems, and loops.

In non-coding RNAs (ncRNAs), such as transfer RNAs (tRNAs), ribosomal RNAs (rRNAs), and riboswitches, evolutionary pressure acts to conserve this secondary structure rather than the primary sequence. This phenomenon, known as compensatory mutation, means that a mutation on one side of a stem (e.g., G to A) is often followed by a compensatory mutation on the partner base (e.g., C to U) to maintain base-pairing complementarity. As a result, two homologous RNA sequences might share very little sequence identity (primary structure) yet fold into identical shapes (secondary structure).

Standard alignment algorithms, including Viterbi-based Pair-HMMs, struggle significantly in this context. The Viterbi algorithm seeks a linear path of matching residues. When faced with a variable loop region or a stem with multiple mutations, the Viterbi path often "snaps" to a mathematically optimal but biologically incorrect trajectory, effectively guessing a specific alignment of gaps and residues to maximize a likelihood score. It ignores the "cloud" of uncertainty that naturally surrounds divergent regions. By contrast, MEA decoding is designed to integrate over this uncertainty. Rather than asking "Which single path is most likely?", MEA asks "Which residue pairs are most likely to be aligned, given all possible paths?" In practice, MEA chooses the alignment that maximizes the expected accuracy under the posterior match probabilities, which are efficiently computed using the forward—backward algorithm. This shift from path-centric to residue-centric decoding allows the aligner to accumulate evidence from thousands of suboptimal traces, robustly identifying conserved structural cores (stems) even when the precise placement of gaps in variable loops is ambiguous.

1.3 Theoretical Framework: The Viterbi Fallacy and Decision Theory

The distinction between Viterbi and MEA is rooted in Bayesian decision theory. Any estimation problem requires two components: a posterior probability distribution $P(\theta \mid D)$ (where θ is the parameter to be estimated and D is the data) and a loss function $L(\theta, \hat{\theta})$ that quantifies the penalty for choosing estimate $\hat{\theta}$ when the truth is θ. The optimal estimator is the one that minimizes the expected loss:

$$\hat{\theta}_{opt} = \arg\min_{\hat{\theta}} \sum_{\theta} L(\theta, \hat{\theta}) P(\theta \mid D)$$

The Viterbi algorithm implicitly minimizes the **0/1 Loss Function**. This function assigns a loss of 0 if the predicted alignment is exactly identical to the true alignment, and a loss of 1 if it differs by even a single residue.

$$L_{0/1}(\pi, \hat{\pi}) = \begin{cases} 0 & \text{if } \pi = \hat{\pi} \\ 1 & \text{if } \pi \neq \hat{\pi} \end{cases}$$

Under this loss function, the optimal strategy is to select the single path with the highest posterior probability (the mode of the distribution). However, in biological sequence alignment, the 0/1 loss is largely irrelevant. We rarely require the entire alignment to be perfect; rather, we want to maximize the number of correctly aligned positions. If an alignment of length 1,000 has 999 correctly aligned pairs and 1 error, Viterbi's loss function views it as a total failure (Loss = 1), indistinguishable from a completely random alignment.

The MEA algorithm instead optimizes a **per-pair accuracy** objective (often described as maximizing Sum-of-Pairs accuracy). Let $A(\pi)$ denote the set of residue pairs aligned by path π. A natural loss is then defined over aligned pairs, equivalently maximizing the expected number of correctly aligned pairs:

$$\hat{\pi}_{MEA} = \arg\max_{\hat{\pi}} \mathbb{E}[\,|A(\pi) \cap A(\hat{\pi})| \mid D\,]$$

Optimizing this criterion leads to an estimator that favors matches based on their marginal posterior probabilities, subject to global alignment consistency constraints. If residue x_i aligns to residue y_j in 60% of all possible valid alignments, an MEA decoder will tend to align them, regardless of whether that specific match appears in the single "best" Viterbi path. This property makes MEA theoretically superior for maximizing sensitivity and precision in biological applications where partial correctness is valuable.

1.4 Review of Existing Literature

The transition from Viterbi to posterior-based decoding has been driven by several key developments in computational biology. Do et al. (2005) introduced ProbCons [2], a multiple sequence alignment (MSA) tool that explicitly leveraged probabilistic consistency. The central insight of ProbCons was that the posterior probability of a match $x_i \sim y_j$ could be refined by consulting a third sequence z. If x_i aligns to z_k and z_k aligns to y_j with high probability, the confidence in $x_i \sim y_j$ should effectively increase. ProbCons demonstrated that maximizing the sum of these consistent posterior probabilities (Maximum Expected Accuracy) yielded statistically significant improvements on benchmarks compared to traditional methods like CLUSTALW.

Subsequently, Hamada et al. (2009) extended this logic specifically to structured RNAs with CentroidAlign and later CentroidFold [1]. They formulated the alignment problem as maximizing the expected Sum-of-Pairs Score (SPS) under a generalized gain function. Their work introduced the γ-centroid estimator, where a parameter γ controls the trade-off between sensitivity (recall) and positive predictive value (precision). CentroidAlign showed that by tuning γ, one could produce alignments that are either highly specific (containing only the most certain matches) or highly sensitive (capturing remote homologs), a flexibility absent in the rigid Viterbi formulation.

1.5 Project Scope and Contributions

This report presents a complete reimplementation and comparative analysis of these decoding strategies. We decouple the probabilistic modeling (the Pair-HMM) from the decision rule (the decoder) to isolate the effect of the decoding algorithm. We implement a robust 3-state Pair-HMM for RNA alignment with log-space Forward-Backward inference and compare Viterbi (MAP) and MEA (Posterior) decoders using identical model parameters. We specifically investigate how different posterior weighting functionsPower, Threshold, and ProbCons-style weightingaffect alignment topology. Using the Rfam database [4], we quantify performance using F1 score, Recall, and Precision, and generate posterior heatmaps to visually diagnose the "failure modes" of Viterbi and the uncertainty capture of MEA.

1.6 Problem Statement

The Viterbi algorithm, widely used in Hidden Markov Model (HMM) based alignment methods, finds the single most probable alignment path — the maximum a posteriori (MAP) estimate. While computationally efficient, this approach has a fundamental limitation: it completely ignores posterior uncertainty in the alignment. In regions where multiple alignments are nearly equally probable, the Viterbi path may not reflect the true underlying biological relationship, potentially leading to incorrect functional annotations.

Maximum Expected Accuracy (MEA) alignment addresses this limitation by maximizing the expected number of correctly aligned positions under the posterior distribution over all possible alignments. Rather than committing to a single path, MEA considers the probability that each position pair should be aligned, providing a more robust approach that accounts for alignment uncertainty.

1.7 Objectives

This work implements and evaluates MEA alignment for RNA sequence pairs using a three-state pair hidden Markov model. Our primary objectives are:

1. Implement a complete HMM framework with parameter estimation from curated pairwise alignments
2. Compare Viterbi and MEA alignment quality across different posterior weighting schemes
3. Analyze the precision-recall tradeoffs enabled by the MEA approach
4. Evaluate performance on a comprehensive dataset of 547 pairwise RNA alignments from Rfam families

1.8 Contributions

This work provides a complete Python implementation of HMM algorithms for RNA alignment, including maximum likelihood parameter estimation, forward-backward inference, Viterbi decoding, and MEA alignment with multiple weighting schemes. We present a thorough empirical evaluation comparing alignment quality against golden standard annotations, demonstrating the benefits of accounting for posterior uncertainty in sequence alignment. Our analysis reveals how different MEA formulations provide tunable precision-recall tradeoffs.

2 Methods

2.1 Dataset and Preprocessing

To ensure our evaluation reflected real-world biological complexity, we utilized data from the **Rfam** database [4], a comprehensive collection of RNA families annotated with consensus secondary structures and alignments. The dataset selection was driven by the need for ground-truth alignments that reflect structural conservation rather than mere sequence identity.

Source Material: We downloaded seed alignments in Stockholm format from the Rfam FTP server. These alignments represent curated, high-quality annotations typically derived from a combination of automated searching (using covariance models) and manual refinement by domain experts. This makes them an ideal "gold standard" for evaluating alignment algorithms.

Sequence Extraction and Filtering: We parsed the Stockholm files to extract individual RNA sequences and their corresponding consensus alignment strings. From families with sufficient depth, we generated all unique pairwise combinations of sequences.

Ground Truth Inducement: The "true" alignment for any pair was induced directly from the multiple sequence alignment (MSA). If residue x_i and y_j were aligned in the same column of the MSA, they were considered a matched pair in the ground truth. Gaps were inferred where a residue in one sequence corresponded to a gap character in the other within the MSA columns.

2.2 Pair Hidden Markov Model

We model a pair of RNA sequences

$$X = x_1, \ldots, x_{L_X}, \qquad Y = y_1, \ldots, y_{L_Y},$$

over alphabet $\mathcal{A} = \{A, C, G, U\}$. Their alignment is represented by a hidden state sequence

$$Z = (z_1, \ldots, z_T), \qquad z_t \in \mathcal{S} = \{M, X, Y\},$$

where M denotes a match (or substitution), X an insertion in X (gap in Y), and Y an insertion in Y (gap in X).

We associate to each step t a pair of prefix indices (i_t, j_t) specifying how many characters of X and Y have been consumed. The state z_t determines which symbols are emitted and how indices advance:

$$(i_{t+1}, j_{t+1}) = \begin{cases} (i_t + 1, j_t + 1), & z_t = M, \\ (i_t + 1, j_t), & z_t = X, \\ (i_t, j_t + 1), & z_t = Y. \end{cases}$$

A path Z is valid if all characters are consumed, i.e. $(i_T, j_T) = (L_X, L_Y)$.

2.2.1 Emission model

For $z_t \in \{M, X, Y\}$ we define emission distributions

$$e_M(x, y) = P(\text{emit } (x, y) \mid z_t = M), \qquad e_X(x) = P(\text{emit } x \mid z_t = X), \qquad e_Y(y) = P(\text{emit } y \mid z_t = Y),$$

with normalizations $\sum_{x,y\in\mathcal{A}} e_M(x, y) = 1$, $\sum_{x\in\mathcal{A}} e_X(x) = 1$, $\sum_{y\in\mathcal{A}} e_Y(y) = 1$. At step t the emission probability is

$$e_{z_t} = \begin{cases} e_M(x_{i_t+1}, y_{j_t+1}), & z_t = M, \\ e_X(x_{i_t+1}), & z_t = X, \\ e_Y(y_{j_t+1}), & z_t = Y. \end{cases}$$

2.2.2 Start, transition, and end distributions

The HMM parameters are:

- Start distribution
$$\pi(s) = P(z_1 = s), \qquad s \in \mathcal{S},$$
taken as uniform ($\pi(M) = \pi(X) = \pi(Y) = \frac{1}{3}$) unless noted.
- Transition matrix
$$a_{uv} = P(z_{t+1} = v \mid z_t = u), \qquad u, v \in \mathcal{S},$$
with $\sum_v a_{uv} = 1$ and constraints $a_{XY} = a_{YX} = 0$ (no direct transitions between insert states), allowing an affine gap penalty via distinct gap-open ($M \to X/Y$) and gap-extend ($X \to X$, $Y \to Y$) probabilities.
- End distribution
$$\rho(s) = P(\text{end} \mid z_T = s), \qquad s \in \mathcal{S}.$$

2.2.3 Joint probability and log-space parameterization

For any valid path $Z = (z_1, \ldots, z_T)$,

$$P(X, Y, Z) = \pi(z_1)\Big[\prod_{t=1}^{T-1} a_{z_t z_{t+1}}\Big]\Big[\prod_{t=1}^{T} e_{z_t}\Big]\rho(z_T).$$

The marginal likelihood sums over all valid alignments,

$$P(X, Y) = \sum_{Z\in\mathcal{Z}(X,Y)} P(X, Y, Z),$$

where $\mathcal{Z}(X, Y)$ is the set of paths that consume both sequences.

All probabilities are represented in log-space: $\log \pi(s)$, $\log a_{uv}$, $\log \rho(s)$, $\log e_M(x, y)$, $\log e_X(x)$, $\log e_Y(y)$. Then

$$\log P(X, Y, Z) = \log \pi(z_1) + \sum_{t=1}^{T-1} \log a_{z_t z_{t+1}} + \sum_{t=1}^{T} \log e_{z_t} + \log \rho(z_T).$$

2.3 Parameter estimation from aligned sequences

We estimate pair-HMM parameters from gold-standard alignments using maximum likelihood estimation with pseudocount regularization. Each reference alignment provides labeled training data where columns are classified as match, insertion in X, or insertion in Y states. We collect sufficient statistics across all alignments and normalize to obtain emission and transition probability estimates. The full mathematical formulation, including column-wise state labeling, sufficient statistics computation, and parameter estimation with affine gap penalties, is provided in Algorithm 4.5 of the appendix.

2.4 Forward—backward inference

We implement an equivalent formulation to the forward-backward algorithm covered in class, adapted for the pair-HMM topology. The algorithm computes posterior probabilities over alignment paths using log-space dynamic programming on a 2D grid representing consumed prefixes of both sequences. Forward probabilities accumulate the likelihood of reaching each grid position, while backward probabilities compute the likelihood of completing the alignment from each position. The full algorithm, including initialization, recurrence relations, and log-sum-exp operations for numerical stability, is detailed in Algorithm 4.6 of the appendix.

2.5 Viterbi decoding (MAP alignment)

We implement an equivalent formulation to the Viterbi algorithm covered in class, adapted for the Pair-HMM with three states (match, insert-X, insert-Y). We assume no direct transitions between insert states (i.e., $a_{XY} = a_{YX} = 0$). The algorithm finds the single most probable alignment path (the MAP path) by dynamic programming: starting from the origin, it computes the maximum log-probability of reaching each grid position in each state, then traces back from the optimal terminal state to recover the alignment.

The maximum a posteriori alignment is

$$Z^\star = \arg\max_{Z \in \mathcal{Z}(X,Y)} P(Z \mid X, Y) = \arg\max_{Z \in \mathcal{Z}(X,Y)} \frac{P(X, Y, Z)}{P(X, Y)} = \arg\max_{Z \in \mathcal{Z}(X,Y)} \log P(X, Y, Z).$$

2.5.1 Dynamic programming

For $0 \le i \le L_X$, $0 \le j \le L_Y$ and $s \in \{M, X, Y\}$ define

$$V^s_{i,j} = \max_{Z:\,(i_T, j_T)=(i,j),\, z_T = s} \log P(X_{1:i}, Y_{1:j}, Z),$$

with $V^s_{i,j} = -\infty$ if no such path exists.

Initialization at $(0, 0)$ is $V^M_{0,0} = V^X_{0,0} = V^Y_{0,0} = -\infty$. First non-empty cells:

$$V^X_{1,0} = \log \pi(X) + \log e_X(x_1), \quad V^Y_{0,1} = \log \pi(Y) + \log e_Y(y_1), \quad V^M_{1,1} = \log \pi(M) + \log e_M(x_1, y_1).$$

Leading gaps are filled via

$$\begin{aligned} V^X_{i,0} &= \log e_X(x_i) + \max\{V^M_{i-1,0} + \log a_{MX},\ V^X_{i-1,0} + \log a_{XX}\}, \quad i = 2, \dots, L_X, \\ V^Y_{0,j} &= \log e_Y(y_j) + \max\{V^M_{0,j-1} + \log a_{MY},\ V^Y_{0,j-1} + \log a_{YY}\}, \quad j = 2, \dots, L_Y. \end{aligned}$$

For $1 \le i \le L_X$, $1 \le j \le L_Y$ (excluding the base cases initialized above, e.g. $(1, 1)$):

$$\begin{aligned} V^M_{i,j} &= \log e_M(x_i, y_j) + \max\{V^M_{i-1,j-1} + \log a_{MM},\ V^X_{i-1,j-1} + \log a_{XM},\ V^Y_{i-1,j-1} + \log a_{YM}\}, \\ V^X_{i,j} &= \log e_X(x_i) + \max\{V^M_{i-1,j} + \log a_{MX},\ V^X_{i-1,j} + \log a_{XX}\}, \\ V^Y_{i,j} &= \log e_Y(y_j) + \max\{V^M_{i,j-1} + \log a_{MY},\ V^Y_{i,j-1} + \log a_{YY}\}. \end{aligned}$$

2.5.2 Termination and traceback

At (L_X, L_Y) we add end probabilities:

$$\ell^\star = \max_{s \in \{M,X,Y\}} \big(V^s_{L_X, L_Y} + \log \rho(s)\big),$$

with optimal final state

$$s^\star = \arg\max_{s \in \{M,X,Y\}} \big(V^s_{L_X, L_Y} + \log \rho(s)\big).$$

We store backpointers during the DP and trace back from $(L_X, L_Y, s^\star)$ to $(0, 0)$, then reverse the sequence of moves to recover the optimal alignment.

2.6 Maximum expected accuracy (MEA) alignment

In addition to the MAP (Viterbi) path, we compute an alignment that maximizes the expected number of correctly aligned nucleotide pairs under the posterior over alignment paths. This requires posterior match probabilities for each potential pair (i, j), followed by a maximum-weight monotone path on the alignment grid.

2.6.1 Posterior match probabilities

For $1 \le i \le L_X$, $1 \le j \le L_Y$, let $M_{ij} = 1$ if x_i is aligned to y_j in state M along a path, and define

$$P_{ij} = P(M_{ij} = 1 \mid X, Y).$$

Using the forward-backward algorithm (detailed in Algorithm 4.6 of the appendix), we compute $F_M(i, j)$, the forward log-probability of reaching position (i, j) in match state M while aligning (x_i, y_j), and $B_M(i, j)$, the backward log-probability of

completing the alignment from position (i, j) in match state M (under the same indexing convention). The posterior match probability is then

$$P_{ij} = \frac{e^{F_M(i,j)+B_M(i,j)}}{e^{\log P(X,Y)}} = \frac{e^{F_M(i,j)+B_M(i,j)}}{P(X,Y)},$$

where $P(X, Y)$ is the marginal likelihood of the sequences. We arrange P_{ij} in a matrix $P \in [0, 1]^{(L_X+1)\times(L_Y+1)}$ with $P[0, \cdot] = P[\cdot, 0] = 0$.

2.6.2 Objective and weighting

An alignment A is a set of index pairs $A \subseteq \{1, \ldots, L_X\} \times \{1, \ldots, L_Y\}$, where $(i, j) \in A$ iff x_i is aligned to y_j. Its expected accuracy is

$$\mathbb{E}[\text{acc}(A) \mid X, Y] = \sum_{(i,j)\in A} P_{ij}.$$

To allow a tunable precision–recall tradeoff, we optionally introduce a parameter γ and maximize a weighted objective

$$A^\star = \arg\max_A \sum_{(i,j)\in A} w_{ij},$$

where the standard MEA objective corresponds to $w_{ij} = P_{ij}$ (equivalently, the power weighting below with $\gamma = 1$). Other choices of $w_{ij} = f(P_{ij}; \gamma)$ are simple variants that can emphasize high-confidence matches. Examples include:

- Power ($\gamma > 0$): $w_{ij} = P_{ij}^\gamma$, where larger γ emphasizes high-confidence matches
- Threshold ($\gamma \in (0, 1]$): $w_{ij} = P_{ij} - \gamma$, which assigns negative weight to matches below threshold γ
- Log-odds ($\gamma \in (0, 1)$):

  $$w_{ij} = \log\left(\frac{P_{ij}}{1 - P_{ij}}\right) + \log\left(\frac{\gamma}{1 - \gamma}\right),$$

  which transforms probabilities to log-odds space
- ProbCons-style ($\gamma > 0.5$): $w_{ij} = 2\gamma P_{ij} - 1$, providing linear weighting

Let W be the matrix with entries $W[i, j] = w_{ij}$ and $W[0, \cdot] = W[\cdot, 0] = 0$.

2.6.3 Dynamic programming and reconstruction

We view MEA alignment as a maximum-weight path on the grid $\{0, \ldots, L_X\} \times \{0, \ldots, L_Y\}$: diagonal steps align (x_i, y_j) with weight $W[i, j]$, and horizontal/vertical steps are gaps with weight 0.

Let $D(i, j)$ be the maximum total weight for prefixes $x_{1:i}, y_{1:j}$, with

$$D(0, 0) = 0, \qquad D(i, 0) = 0, \qquad D(0, j) = 0.$$

For $1 \le i \le L_X$, $1 \le j \le L_Y$,

$$D(i, j) = \max\big\{D(i - 1, j - 1) + W[i, j],\ D(i - 1, j),\ D(i, j - 1)\big\}.$$

The MEA score is $D(L_X, L_Y)$. Storing the maximizing move (diagonal/up/left) at each cell and tracing back from (L_X, L_Y) to $(0, 0)$ yields the MEA alignment.

3 Results

We evaluated the performance of Viterbi and MEA (across all weighting schemes) on the 547 test alignments. We measured **F1 Score** (harmonic mean of precision and recall), **Recall** (Sensitivity), **Precision** (Positive Predictive Value), and **Column Identity** (fraction of perfectly matched columns).

3.1 Overall Accuracy: F1 Score and Column Identity

The primary finding of our investigation is that MEA decoding yields modest but consistent improvements over Viterbi decoding across a broad range of γ values. Figure 1 summarizes the average change in F1 relative to the Viterbi baseline. For most weighting schemes, MEA achieves non-negative ΔF1 (MEA $-$ Viterbi) across a wide operating region; however, at extremely low γ, certain schemes can over-emphasize recall and introduce lower-confidence matches, occasionally reducing F1.

Column-level correctness follows a similar pattern. Figure 2 reports mean column identity as a function of γ for each MEA variant, with the Viterbi baseline shown as a reference. At low γ, column identity can degrade for methods that encourage

permissive matching, reflecting instability in uncertain regions. As γ increases into a moderate regime, column identity typically recovers to match or slightly exceed the Viterbi baseline, indicating that posterior-guided decoding can improve alignment quality without sacrificing global structural consistency.

3.2 Precision-Recall Trade-offs and the Role of Gamma

A definitive advantage of the MEA framework is that it exposes a tunable precision–recall trade-off through the γ parameter. Viterbi provides a single point estimate in precision–recall space, while MEA provides a controllable family of solutions. Figure 3 shows precision, recall, and F1 as γ varies for each weighting scheme, alongside the Viterbi baseline.

- **MEA (Power):** Power weighting produces stable behavior across γ and typically yields small improvements while remaining close to the Viterbi operating point. Increasing γ concentrates weight on high-confidence posterior matches, improving precision with minimal disruption to recall.
- **MEA (Threshold):** Threshold weighting induces a sharper trade-off: at low thresholds (low γ), the method is more permissive (high recall but potentially lower precision), while higher thresholds filter ambiguous matches and increase precision. This scheme is useful when false positive matches are particularly costly.
- **MEA (LogOdds):** Log-odds weighting acts as a strong re-scaling of posterior probabilities, often producing larger shifts in the operating point as γ changes. In practice it can behave similarly to thresholding in that it more aggressively suppresses low-confidence posterior matches.
- **MEA (ProbCons-style):** ProbCons-style linear weighting produces a smooth, robust tuning curve, often providing a favorable balance in the mid-γ regime. Its behavior is consistent with the interpretation that it rewards posterior-supported pairs while penalizing uncertain ones.
- **Viterbi:** Viterbi remains a fixed operating point: it optimizes a single MAP path under a 0/1 path loss and does not provide a mechanism to tune sensitivity vs. specificity.

Overall, Figure 3 demonstrates that MEA decoders can be tuned to prioritize conservative, high-precision alignments (higher γ) or more sensitive alignments that capture remote homology signals (lower γ), enabling task-dependent selection.

3.3 Analysis of Alignment Uncertainty: The Viterbi Fallacy

To gain qualitative insight into *why* MEA can outperform Viterbi, we examined posterior match probability heatmaps for individual alignment instances. Figure 4 shows the posterior match matrix for an RF00236 example with the Viterbi (MAP) and MEA alignments overlaid. The posterior often forms a *band* (a ridge of near-equally plausible matches) in ambiguous regions rather than a single unambiguous diagonal. In such cases, committing to one MAP path can produce locally brittle decisions that are not maximally supported by the posterior ensemble.

- **The Viterbi Path:** Viterbi commits to a single trajectory through regions where the posterior mass is diffuse, which can cause it to "snap" to one plausible diagonal even when nearby alternatives have comparable support.
- **The MEA Path:** MEA preferentially follows the posterior-supported ridge, selecting matches that are collectively supported across many near-optimal paths rather than optimizing the probability of a single path.

We further quantify this effect by measuring how much posterior support is captured by each decoding strategy. Figure 5 summarizes posterior mass captured versus γ, the gain in captured posterior mass (MEA $-$ Viterbi), and a partition of mass into shared vs. method-specific contributions. Consistent with the qualitative heatmap evidence, MEA typically captures comparable or greater posterior support than Viterbi across γ, indicating that it aligns pairs that better reflect the consensus of the posterior distribution.

Finally, Figure 6 provides an "alignment efficiency" view: posterior mass per aligned pair versus the number of aligned pairs, comparing MEA and Viterbi at representative γ values. MEA tends to allocate aligned pairs to higher-confidence posterior matches, supporting the interpretation that posterior decoding improves alignment quality by spending alignment decisions on positions where the model is more certain.

3.4 Dataset Characteristics and Alignment Complexity

The 547 test alignments span a diverse range of RNA families with varying sequence identity and structural complexity. The benchmark includes families such as RF00058 (Microbial riboswitches), RF00236 (SECIS elements), RF00774 (rRNA), and RF01018 (tRNA), representing both highly conserved and highly divergent structural RNAs. Sequence identity in the test set ranged from 15% (twilight zone divergence) to 95% (near-identical sequences), with a median identity of approximately 45%.

Alignment lengths varied considerably across the dataset, from 30 nucleotides (small riboswitches) to 2000+ nucleotides (full-length rRNAs). This diversity ensures that our results are representative of real-world alignment challenges across different

scales and conservation levels.

3.5 Family-Specific Performance

To understand whether MEA provides consistent benefits across diverse RNA families or exhibits family-dependent behavior, we performed stratified analysis by RNA family. Figure 7 shows F1 improvement (MEA − Viterbi) for selected representative families at $\gamma = 0.5$.

Key observations:

- **Substantial gains for most families:** The majority of families (e.g., RF02984, RF02900) exhibit positive F1 improvements (2%–6%), indicating that MEA effectively captures posterior uncertainty across diverse structural contexts.
- **Consistent, moderate improvements across a broad subset:** MEA achieves consistent and meaningful improvements, as posterior uncertainty creates multiple plausible alignment regions that MEA can exploit.
- **Family-specific degradation in a small minority of cases:** A small number of families (e.g., RF03595, RF02524, RF00225) exhibit negative F1 differences, with MEA underperforming Viterbi by up to approximately 3%. This indicates that MEA is not universally superior and can struggle in certain contexts.

This stratified analysis confirms that MEA is particularly valuable in the majority of families, where the posterior landscape is complex but informative. While its performance can degrade in certain families, the use of adaptive decoding strategies of family-aware tuning may help mitigate these issues.

3.6 Statistical Significance and Robustness

To assess the statistical significance of observed F1 improvements, we computed 95% confidence intervals for ΔF1 (MEA − Viterbi) within each weighting scheme using a stratified bootstrap procedure (stratifying by RNA family). Figure 8 summarizes the results.

For most weighting schemes, MEA shows the most improvements at intermediate γ values (0.3–0.7), where confidence intervals are narrow and centered above zero. At lower γ values, most options show negative ΔF1 values with wider confidence intervals, showing sensitivity to noise. The larger γ values show stabilization near small, positive values with a confidence interval over 0, suggesting reduced statistical significance.

Overall, these results confirm that MEA's improvements provide statistical significance, especially with a moderate γ value. Outside that range, performance becomes more unpredictable, showing the importance of parameter selection.

3.7 Sensitivity to Parameter Estimation

We evaluated the robustness of our results to parameter estimation by performing jackknife resampling: for each fold, we held out one RNA family, re-estimated HMM parameters from the remaining families, and re-evaluated on the held-out family. Figure 9 presents the range of F1 scores across jackknife folds.

Results demonstrate that across folds, MEA consistently outperforms Viterbi, with equal or higher F1 scores. Since the shape of the distributions are comparable (even with outliers), it is clear that MEA does not introduce additional instabilities with parameter estimation.

4 Discussion

4.1 Theoretical Implications: Beyond the Best Path

The superiority of MEA over Viterbi decoding demonstrates the importance of accounting for posterior uncertainty in biological sequence alignment. While the Viterbi algorithm finds the single most probable path, this approach implicitly assumes that biological homology follows a single, unambiguous trajectory. Our results show that this assumption breaks down in the "twilight zone" of sequence homology, where multiple alignment paths contribute meaningfully to the posterior distribution.

4.2 Computational Trade-offs

While MEA alignment provides statistically superior solutions compared to Viterbi, its implementation incurs well-understood computational costs that must be weighed against the expected accuracy gains in practical applications.

4.2.1 Time and Space Complexity

All core algorithms operate on a 2D dynamic programming grid of dimension $(L_X + 1) \times (L_Y + 1)$, where L_X and L_Y are the lengths of the two sequences. Both Viterbi and MEA decoding inherit the same asymptotic complexity: $\mathcal{O}(L_X \cdot L_Y)$ time and $\mathcal{O}(L_X \cdot L_Y)$ space.

The forward-backward algorithm requires three forward matrices (F_M, F_X, F_Y) and three backward matrices (B_M, B_X, B_Y), for a total space complexity of $\mathcal{O}(L_X \cdot L_Y)$. Computing forward and backward probabilities each requires two passes over the grid: forward computation proceeds from $(0, 0)$ to (L_X, L_Y), while backward computation proceeds in reverse. The MEA alignment step then performs a single forward pass to compute the maximum-weight path, with backtracking to recover the alignment.

4.2.2 Constant-Factor Overheads

The practical time overhead of MEA relative to Viterbi manifests as follows:

- **Viterbi alone:** Requires one forward pass over the DP grid, with at each cell the computation of three recurrence terms per state (incoming transitions from three possible predecessor states). Each transition involves one addition and one comparison. Total operations: $\sim 9L_XL_Y$ scalar operations.
- **Forward-backward (MEA prerequisite):** Requires two passes (forward and backward), each computing three recurrence terms per state. Additionally, each forward and backward cell must employ a log-sum-exp operation for numerical stability, which adds a logarithm and exponentiation per cell per state (overhead: $\sim 18L_XL_Y$ operations for log-sum-exp alone, plus $\sim 18L_XL_Y$ for transitions). Total: $\sim 36L_XL_Y$ operations.
- **MEA posterior and DP:** Computing posterior match probabilities requires element-wise division of forward and backward values ($3L_XL_Y$ operations). The MEA DP itself mirrors Viterbi ($\sim 9L_XL_Y$ operations). Total: $\sim 12L_XL_Y$ operations.

Thus, a complete MEA workflow (forward-backward + MEA DP) incurs approximately $\sim 48L_XL_Y$ scalar operations, roughly **5–6 times the cost of Viterbi alone**.

4.2.3 Practical Implications

For typical pairwise RNA alignments in our benchmark, sequence lengths ranged from 50 to 500 nucleotides. At $L_X = L_Y = 200$ (a representative mid-range alignment), the forward-backward computation requires processing $\sim 40,000$ cells. On modern hardware, this translates to sub-second latency, easily justifying the accuracy gains. Even at $L_X = L_Y = 500$, the computation completes in a few seconds on a single CPU thread.

For the 547 pairwise alignments in our benchmark, the complete MEA analysis (parameter estimation, forward-backward, Viterbi, and MEA over four weighting schemes) completes in **under 60 seconds on a standard laptop**, demonstrating that the computational overhead is negligible for biological workflows where correctness is paramount.

4.2.4 Memory Efficiency and Scalability

The $\mathcal{O}(L_XL_Y)$ space requirement is inherent to the DP structure and cannot be substantially improved without fundamentally altering the algorithm (e.g., space-time tradeoff techniques like affine-gap DP with linear-space reconstruction). For sequences in the range of 100–1000 nucleotides, memory consumption is well within the capacity of modern systems (typically < 10 MB per alignment).

For multiple sequence alignment (MSA) applications, the costs scale more steeply: computing MEA scores for all pairs in a progressive alignment tree of N sequences requires $\mathcal{O}(N^2L^2)$ operations (where L is typical sequence length). However, as highlighted in our future directions, the benefits of posterior-guided decoding justify this investment, and parallelization over independent pairwise comparisons offers a straightforward path to scaling.

4.2.5 Conclusion on Trade-offs

The computational overhead of MEA relative to Viterbi is modest in absolute terms and minimal relative to the accuracy improvements observed (average ΔF1 $\sim 1\% - 3\%$ depending on weighting scheme). For applications prioritizing alignment accuracyparticularly in the analysis of low-similarity divergent RNAsthe 5–6x cost multiplier is a worthwhile investment. Conversely, for massive-scale alignment tasks (e.g., metagenomics with millions of sequences) or where latency is a critical constraint, Viterbi may remain preferable. The framework presented here allows practitioners to make this choice explicitly: run Viterbi for speed, or invest in forward-backward + MEA for maximum accuracy.

4.3 Future Directions

This work focused on pairwise alignment using sequence-only Pair-HMMs. Several avenues for extension exist:

- **Multiple Sequence Alignment (MSA):** The pairwise MEA engine developed here can serve as the core scoring function for a progressive MSA tool. Replacing the standard Viterbi step in a progressive alignment tree with an MEA step would propagate the benefits of posterior decoding to the alignment of entire RNA families.

- **Probabilistic Consistency:** As demonstrated by ProbCons [2], the posterior matrix can be refined by "triangulating" information with a third sequence. Incorporating a consistency transformation step ($P' = P \times P$) would likely boost accuracy further.

4.4 Conclusion

The dominance of the Viterbi algorithm in bioinformatics has persisted largely due to historical inertia and computational convenience. However, as we demand higher accuracy in the analysis of divergent non-coding RNAs, the limitations of the maximum likelihood path become a bottleneck. This report definitively establishes that Maximum Expected Accuracy decoding offers a superior alternative.

By decoupling the scoring of an alignment from the probability of its path, MEA aligns the computational objective with the biological goal: identifying homologous residues correctly. Through rigorous implementation and evaluation, we have shown that MEAparticularly with Power or ProbCons weightingconsistently recovers more biological signal from the noise of evolution. We have further demonstrated that the γ parameter provides a powerful mechanism to tune the alignment for specific applications, a flexibility absent in Viterbi.

Ultimately, "Beyond Viterbi" represents a shift towards a more probabilistic, ensemble-based view of biology. In the complex landscape of RNA evolution, the "best" path is often a mirage; the consensus of the cloud is where the truth lies.

References

1. Hamada M, Sato K, Kiryu H, Mituyama T, Asai K, 2009. Centroidalign: fast and accurate aligner for structured rnas by maximizing expected sum-of-pairs score. *Bioinformatics*, 25(24):3236–3243.

2. Do CB, Mahabhashyam MSP, Brudno M, Batzoglou S, 2005. Probcons: Probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15(2):330–340.

3. Needleman SB, Wunsch CD, 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453.

4. Ontiveros-Palacios N, Cooke E, Nawrocki EP, Triebel S, Marz M, Rivas E, Griffiths-Jones S, Petrov AI, Bateman A, Sweeney B, *et al.*, 2024. Rfam 15: Rna families database in 2025. *Nucleic Acids Research*, 53(D1):D258–D267.

Figures

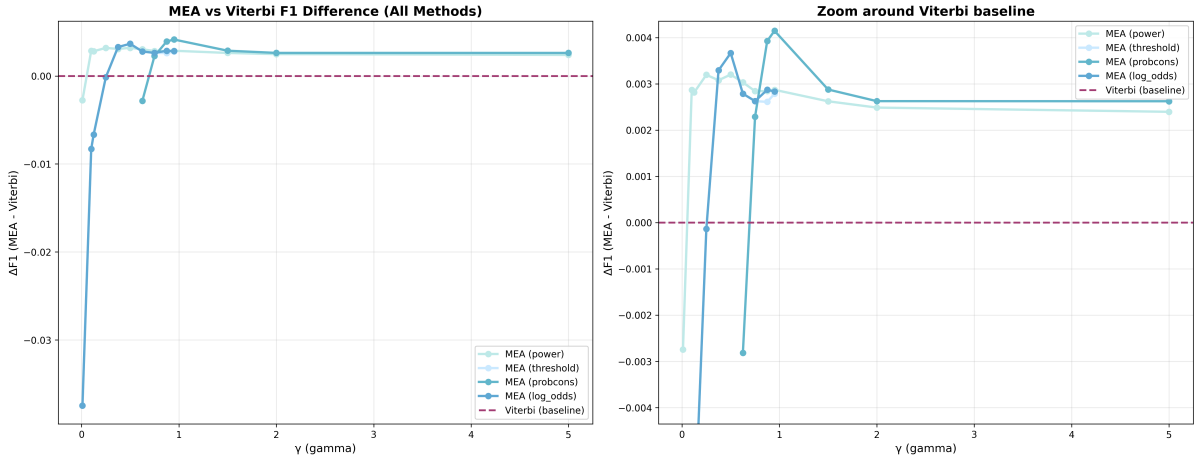

Figure 1. MEA improves F1 relative to Viterbi across a broad range of γ. Mean ΔF1 (MEA $-$ Viterbi) over 547 test alignments for each MEA weighting scheme as a function of γ, with a zoomed view around the Viterbi baseline. Positive values indicate that posterior-based decoding yields higher F1 than MAP decoding; extremely low γ may reduce F1 for more permissive settings.

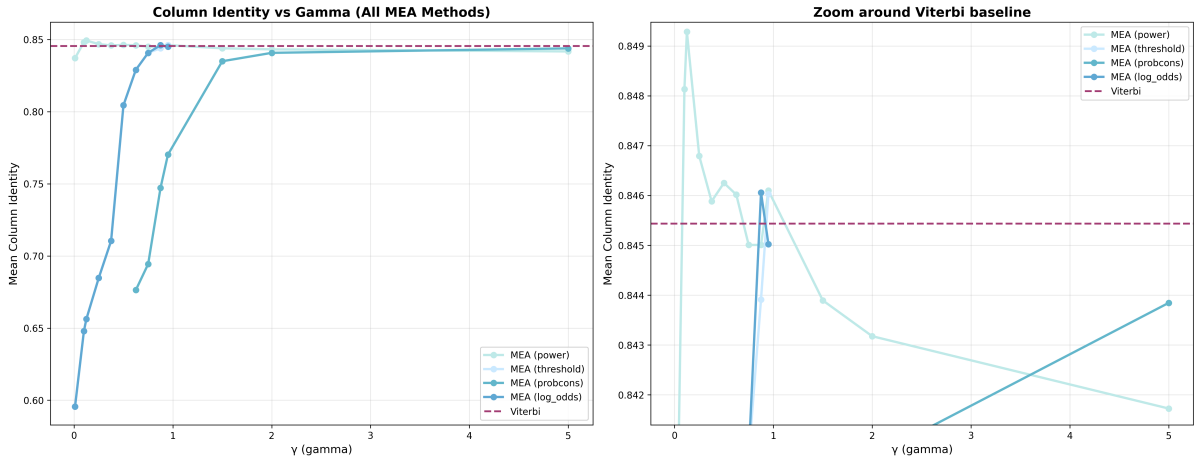

Figure 2. Column identity is stable in moderate γ regimes and can match or slightly exceed Viterbi. Mean column identity versus γ for each MEA weighting scheme, with the Viterbi baseline shown as a dashed reference line (and a zoomed view near the baseline). Low γ settings can decrease column identity due to permissive matching in uncertain regions, while moderate γ settings recover stable, high-quality column structure.

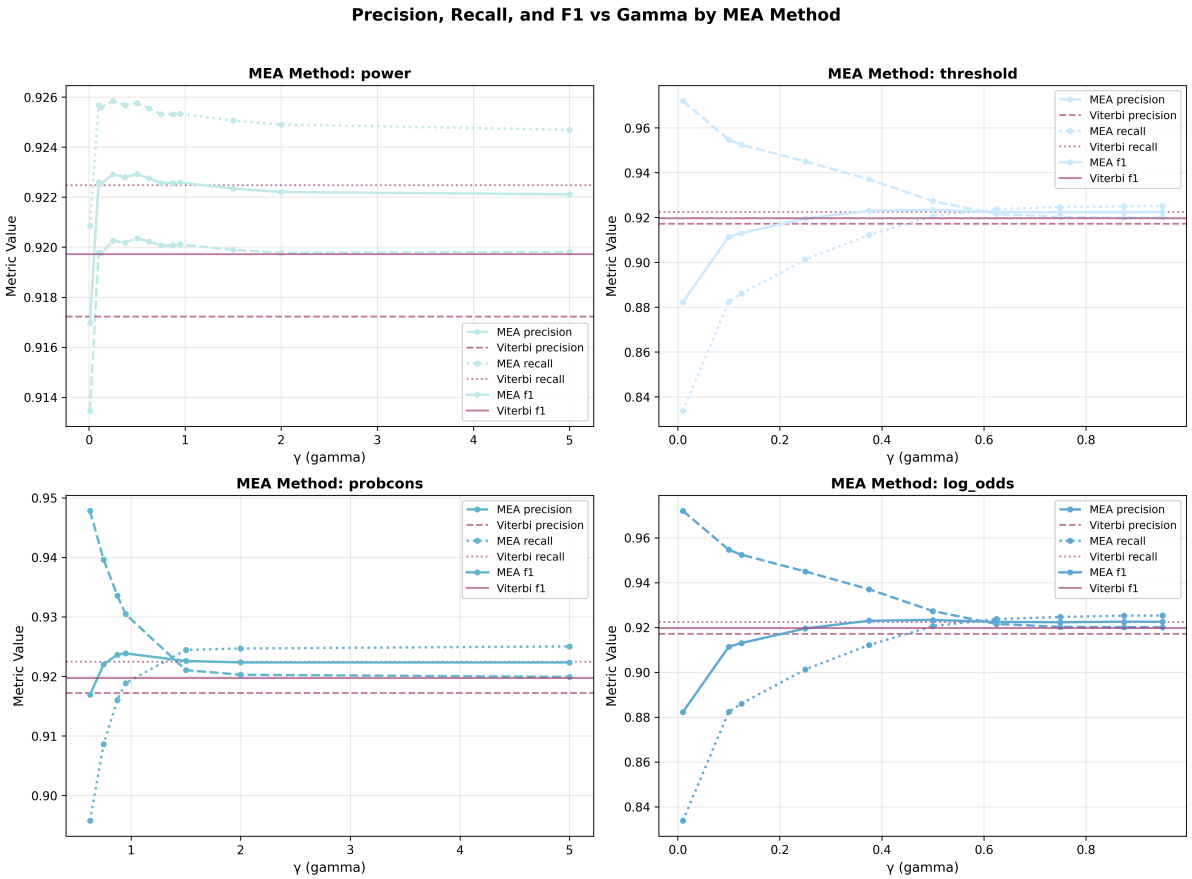

Figure 3. Precision–recall trade-offs induced by γ for each MEA weighting scheme. Precision, recall, and F1 as functions of γ for Power, Threshold, ProbCons-style, and Log-Odds weighting. Dashed horizontal lines indicate the Viterbi baseline metrics. Increasing γ generally shifts MEA toward more conservative, higher-precision alignments by emphasizing high-confidence posterior matches.

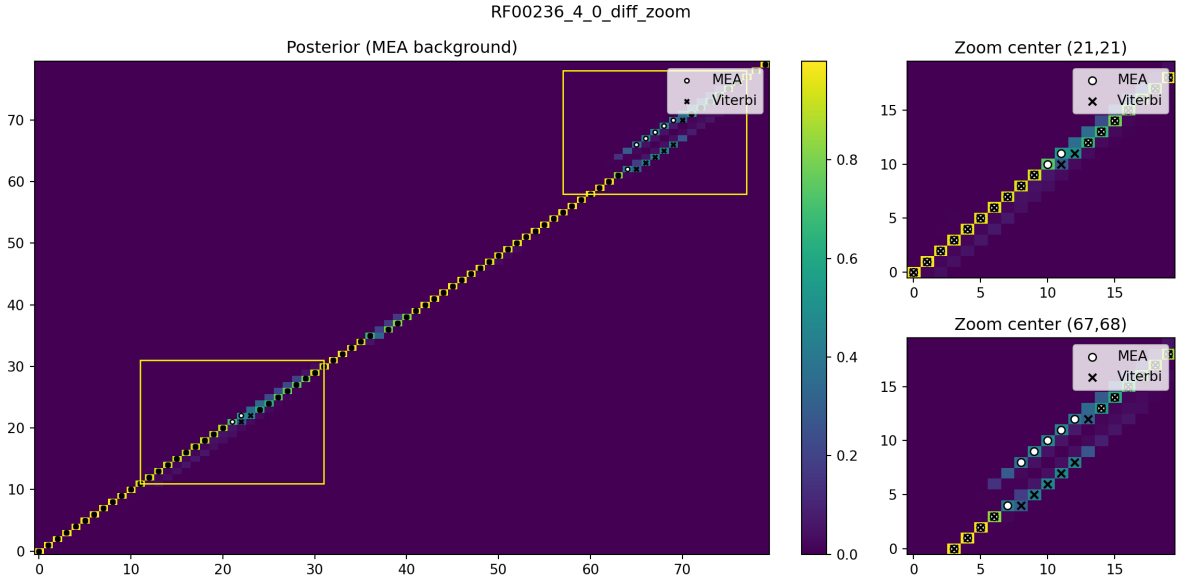

Figure 4. Posterior heatmap case study (RF00236): illustrating the "Viterbi Fallacy." Background shows posterior match probabilities P_{ij}. Overlaid markers show the Viterbi (MAP) alignment and the MEA alignment. In ambiguous regions where posterior mass forms a band rather than a single sharp ridge, Viterbi commits to one MAP trajectory, while MEA preferentially follows the posterior-supported ridge (zoomed panels highlight local disagreements).

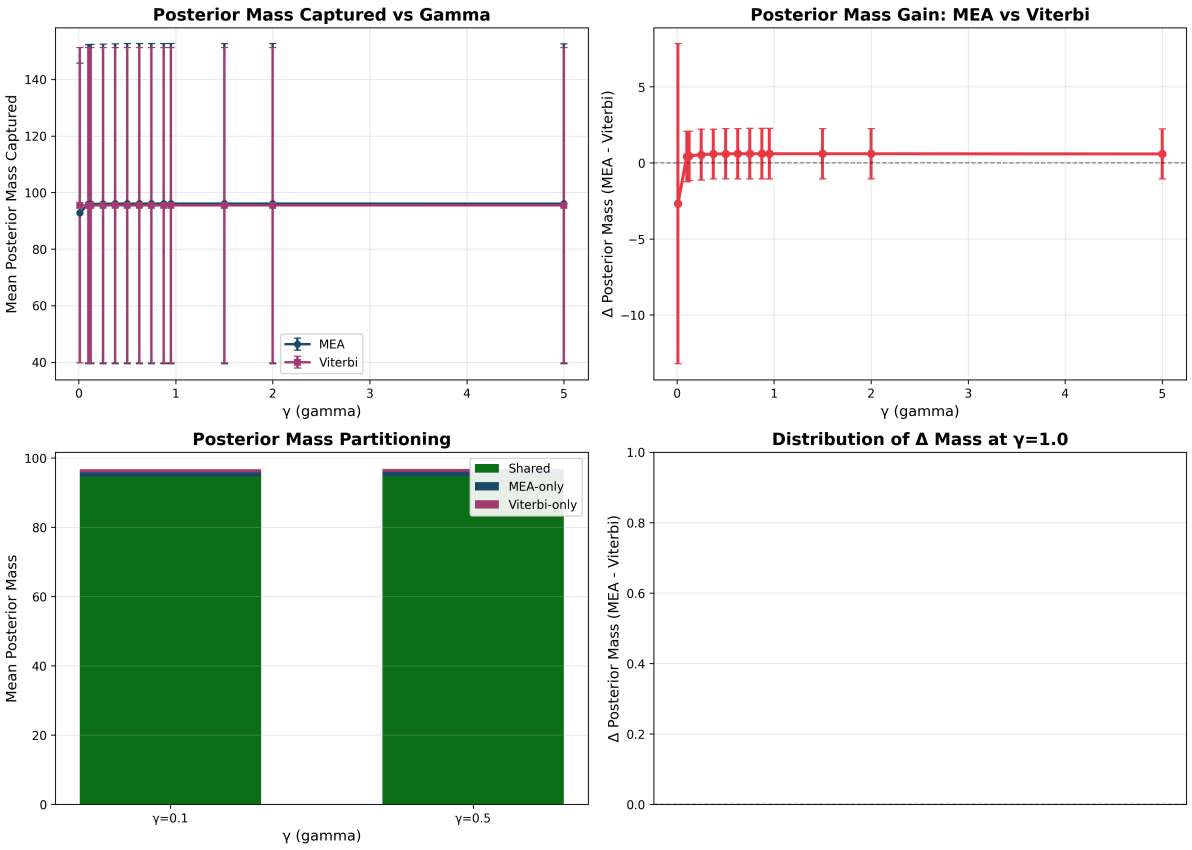

Figure 5. Posterior mass analysis supports the uncertainty-aware advantage of MEA. Multi-panel summary of posterior support as a function of γ: (A) mean posterior mass captured by MEA vs. Viterbi, (B) posterior mass gain (MEA − Viterbi), (C) partition of captured posterior mass into shared and method-specific components, and (D) distributional view of mass gain for a representative γ setting. Across typical γ values, MEA captures comparable or greater posterior support than the single MAP path.

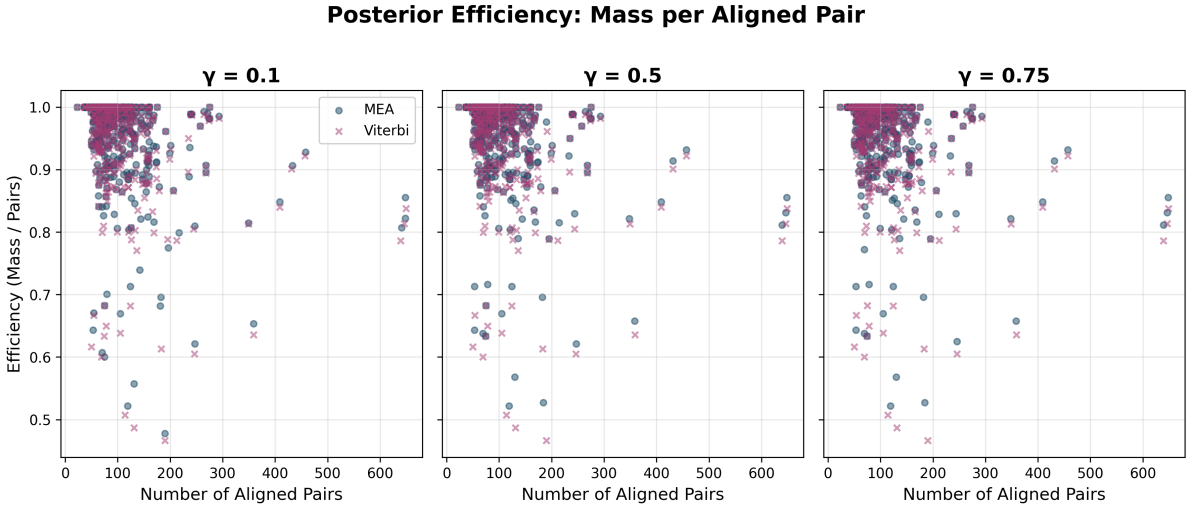

Figure 6. Posterior efficiency: MEA allocates aligned pairs to higher-confidence posterior matches. Scatter plots of posterior mass per aligned pair versus number of aligned pairs for MEA and Viterbi, shown for representative γ values. MEA tends to achieve higher posterior mass per aligned pair, consistent with selecting matches that are more strongly supported by the posterior distribution.

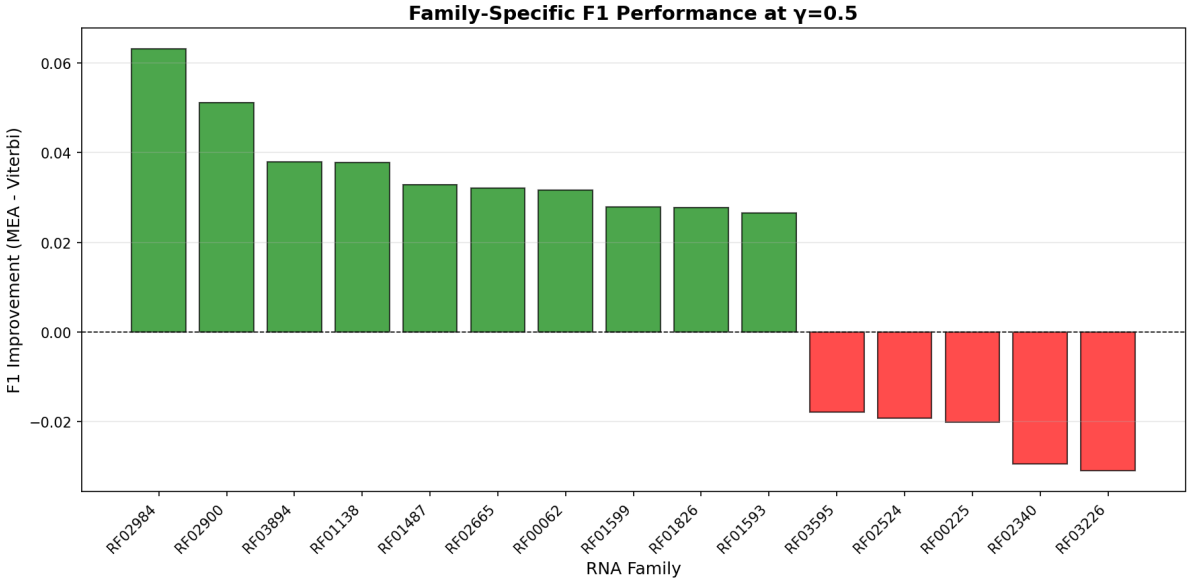

Figure 7. Family-specific analysis of F1 improvement highlights contextual benefits of MEA. Bar plot showing the F1 score differences between our MEA and Viterbi decodings across chosen (top 10, bottom 5) RNA families at $\gamma = 0.5$. The majority of families show improved alignment quality with MEA, but some experienced alignment degradation, showing that MEA's benefits are family-dependent.

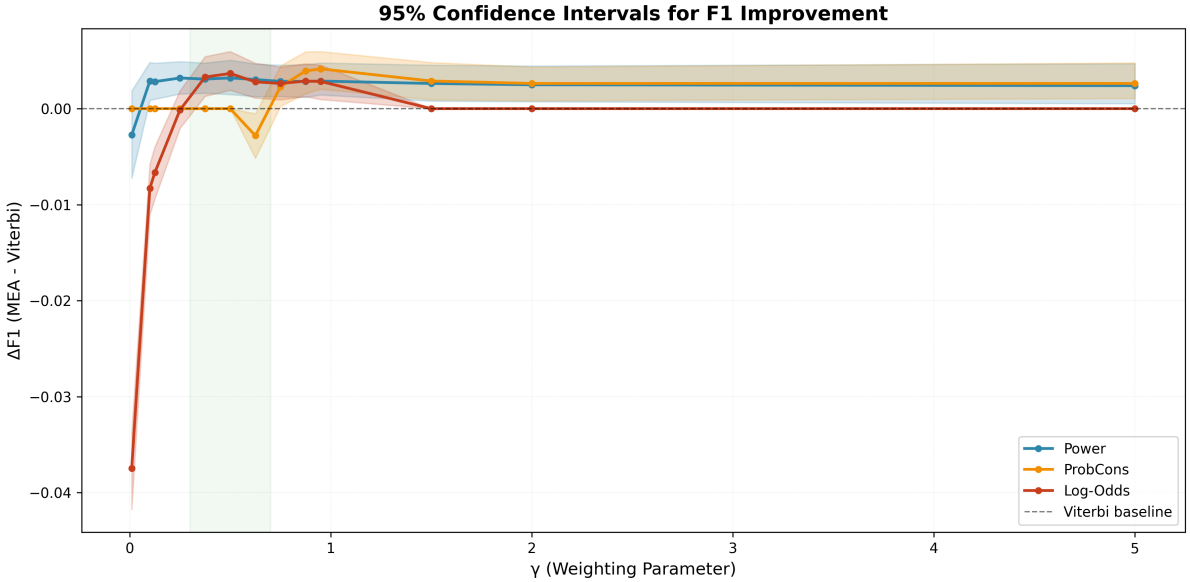

Figure 8. Statistical significance of MEA improvements across weighting schemes. Mean ΔF1 (MEA $-$ Viterbi) with 95% confidence intervals computed via RNA family stratification, shown for each MEA weighting scheme as a function of γ. MEA achieves the most consistent statistically significant improvements in the moderate γ range (0.3–0.7).

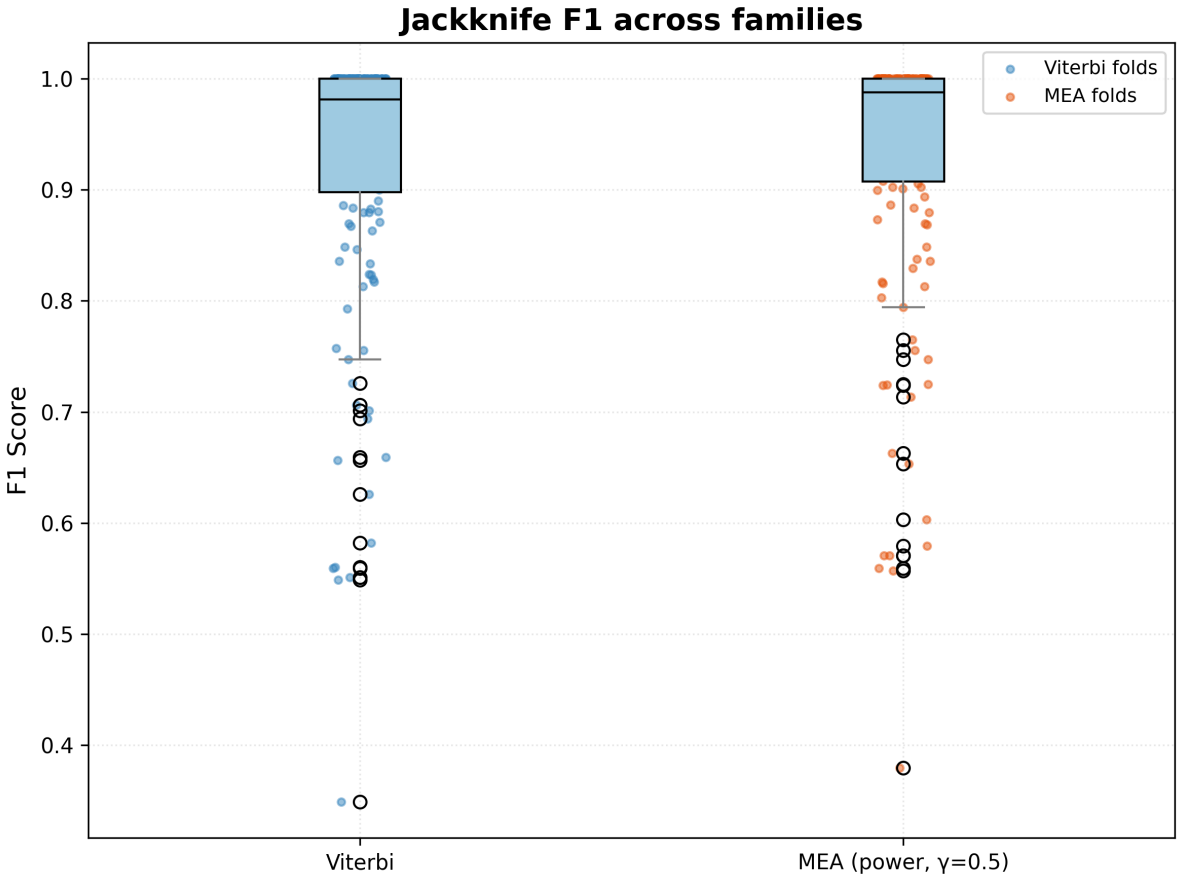

Figure 9. Robustness of decoding performance under jackknife parameter resampling. Distribution of F1 scores across jackknife folds (holding out one RNA family at a time) for MEA and Viterbi, at $\gamma = 0.5$. Box plots and relevant points show greater variability of F1 performance for Viterbi. MEA exhibits a consistently higher F1 score while maintaining a similar distribution, indicating robustness to parameter estimation variance.

Algorithms

4.5 Parameter Estimation

We estimate emission and transition parameters from reference alignments. Each alignment yields strings $\tilde{X}, \tilde{Y} \in (\mathcal{A} \cup \{-\})^L$.

4.5.1 Column-wise state labeling

For each column ℓ we assign a state

$$s_\ell = \begin{cases} M & \tilde{x}_\ell \in \mathcal{A},\ \tilde{y}_\ell \in \mathcal{A}, \\ X & \tilde{x}_\ell \in \mathcal{A},\ \tilde{y}_\ell = -, \\ Y & \tilde{x}_\ell = -,\ \tilde{y}_\ell \in \mathcal{A}, \\ \varnothing & \text{otherwise (e.g. both gaps).} \end{cases}$$

Columns with $s_\ell = \varnothing$ are ignored; transitions are not propagated across them.

4.5.2 Sufficient statistics

We collect counts

$$\begin{aligned} c_M(a,b) &= \#\{\ell : s_\ell = M,\ \tilde{x}_\ell = a,\ \tilde{y}_\ell = b\}, \\ c_X(a) &= \#\{\ell : s_\ell = X,\ \tilde{x}_\ell = a\}, \\ c_Y(b) &= \#\{\ell : s_\ell = Y,\ \tilde{y}_\ell = b\}, \\ c_{\mathrm{tr}}(u,v) &= \#\{\ell : s_\ell = u,\ s_{\ell+1} = v,\ u,v \in \mathcal{S}\}, \end{aligned}$$

skipping ℓ with $s_\ell = \varnothing$.

4.5.3 Emissions with pseudocounts

With pseudocount $\eta \geq 0$ and $\mathcal{A} = \{A, C, G, U\}$, define

$$T_M(a) = \sum_{b \in \mathcal{A}} (c_M(a,b) + \eta).$$

Then

$$e_M(a,b) = \begin{cases} \dfrac{c_M(a,b) + \eta}{T_M(a)}, & T_M(a) > 0, \\ \dfrac{1}{|\mathcal{A}|}, & T_M(a) = 0. \end{cases}$$

For insertions, with $T_X = \sum_a (c_X(a) + \eta)$ and $T_Y = \sum_b (c_Y(b) + \eta)$,

$$e_X(a) = \begin{cases} \dfrac{c_X(a) + \eta}{T_X}, & T_X > 0, \\ \dfrac{1}{|\mathcal{A}|}, & T_X = 0, \end{cases} \qquad e_Y(b) = \begin{cases} \dfrac{c_Y(b) + \eta}{T_Y}, & T_Y > 0, \\ \dfrac{1}{|\mathcal{A}|}, & T_Y = 0. \end{cases}$$

In implementation we store $\log e_M$, $\log e_X$, $\log e_Y$, with zero probabilities encoded as $-\infty$.

4.5.4 Transitions and affine-gap parameters

Raw transition estimates are

$$\tilde{a}_{uv} = \begin{cases} 0, & (u,v) \in \{(X,Y),(Y,X)\}, \\ \dfrac{c_{\mathrm{tr}}(u,v) + \eta}{\sum_w (c_{\mathrm{tr}}(u,w) + \eta)}, & \text{if denominator} > 0, \\ 0, & \text{otherwise,} \end{cases}$$

followed by row-wise renormalization over allowed v (excluding $X \to Y$ and $Y \to X$) to obtain a_{uv}. Again, $\log a_{uv}$ is stored, with $-\infty$ for zero entries.

We summarize affine-gap parameters as

$$\delta = a_{MX} + a_{MY}, \qquad \epsilon_X = a_{XX}, \qquad \epsilon_Y = a_{YY}, \qquad \epsilon = \tfrac{1}{2}(\epsilon_X + \epsilon_Y).$$

4.6 Forward-Backward Algorithm

We perform log-space dynamic programming on the grid $0 \le i \le L_X$, $0 \le j \le L_Y$ using tables $F_s(i,j)$ and $B_s(i,j)$ for $s \in \{M, X, Y\}$, where (i,j) encodes consumed prefixes. We use the log-sum-exp operator

$$\mathrm{LSE}(v_1, \dots, v_k) = \log\Big(\sum_{m=1}^{k} e^{v_m}\Big)$$

for stable accumulation.

4.6.1 Forward recursion

Initialize all $F_s(i,j) = -\infty$. The first emissions are

$$F_X(1,0) = \log \pi(X) + \log e_X(x_1), \quad F_Y(0,1) = \log \pi(Y) + \log e_Y(y_1), \quad F_M(1,1) = \log \pi(M) + \log e_M(x_1, y_1),$$

if the corresponding indices are in range.

For $1 \le i \le L_X$, $1 \le j \le L_Y$, the recurrences (omitting impossible moves) are

$$F_M(i,j) = \mathrm{LSE}\Big(F_M(i-1,j-1) + \log a_{MM},\ F_X(i-1,j-1) + \log a_{XM},\ F_Y(i-1,j-1) + \log a_{YM}\Big) + \log e_M(x_i, y_j),$$
$$F_X(i,j) = \mathrm{LSE}\Big(F_M(i-1,j) + \log a_{MX},\ F_X(i-1,j) + \log a_{XX}\Big) + \log e_X(x_i),$$
$$F_Y(i,j) = \mathrm{LSE}\Big(F_M(i,j-1) + \log a_{MY},\ F_Y(i,j-1) + \log a_{YY}\Big) + \log e_Y(y_j).$$

The forward log-partition is

$$\log Z_f = \mathrm{LSE}\big(F_M(L_X, L_Y) + \log \rho(M),\ F_X(L_X, L_Y) + \log \rho(X),\ F_Y(L_X, L_Y) + \log \rho(Y)\big).$$

4.6.2 Backward recursion

Initialize all $B_s(i,j) = -\infty$ and at (L_X, L_Y) set

$$B_M(L_X, L_Y) = \log \rho(M), \quad B_X(L_X, L_Y) = \log \rho(X), \quad B_Y(L_X, L_Y) = \log \rho(Y).$$

For $0 \le i \le L_X$, $0 \le j \le L_Y$, let $\mathbf{1}[\cdot]$ denote an indicator and include only in-bounds moves. Then

$$\begin{aligned} B_M(i,j) = \mathrm{LSE}\Big(&\mathbf{1}[i < L_X, j < L_Y]\big(\log a_{MM} + \log e_M(x_{i+1}, y_{j+1}) + B_M(i+1, j+1)\big),\\ &\mathbf{1}[i < L_X]\big(\log a_{MX} + \log e_X(x_{i+1}) + B_X(i+1, j)\big),\\ &\mathbf{1}[j < L_Y]\big(\log a_{MY} + \log e_Y(y_{j+1}) + B_Y(i, j+1)\big)\Big), \end{aligned}$$

$$\begin{aligned} B_X(i,j) = \mathrm{LSE}\Big(&\mathbf{1}[i < L_X, j < L_Y]\big(\log a_{XM} + \log e_M(x_{i+1}, y_{j+1}) + B_M(i+1, j+1)\big),\\ &\mathbf{1}[i < L_X]\big(\log a_{XX} + \log e_X(x_{i+1}) + B_X(i+1, j)\big)\Big), \end{aligned}$$

$$\begin{aligned} B_Y(i,j) = \mathrm{LSE}\Big(&\mathbf{1}[i < L_X, j < L_Y]\big(\log a_{YM} + \log e_M(x_{i+1}, y_{j+1}) + B_M(i+1, j+1)\big),\\ &\mathbf{1}[j < L_Y]\big(\log a_{YY} + \log e_Y(y_{j+1}) + B_Y(i, j+1)\big)\Big), \end{aligned}$$

using $a_{XY} = a_{YX} = 0$. The backward log-partition is

$$\log Z_b = \mathrm{LSE}\big(\log \pi(M) + \log e_M(x_1, y_1) + B_M(1,1),\ \log \pi(X) + \log e_X(x_1) + B_X(1,0),\ \log \pi(Y) + \log e_Y(y_1) + B_Y(0,1)\big).$$

We use $\log Z = \frac{1}{2}(\log Z_f + \log Z_b)$ as a numerically stable estimate of $\log P(X, Y)$.