Peek Search: Near-Optimal Online Markov Decoding

Vikas K. Garg 1  Tamar Pichkhadze 1

Abstract

We resolve the fundamental problem of online decoding with ergodic Markov models. Specifically, we provide deterministic and randomized algorithms that are provably near-optimal under latency constraints with respect to the unconstrained offline optimal algorithm. Our algorithms admit efficient implementation via dynamic programs, and extend to (possibly adversarial) non-stationary or time-varying Markov settings as well. Moreover, we establish lower bounds in both deterministic and randomized settings subject to latency requirements, and prove that no online algorithm can perform significantly better than our algorithms.

1. Introduction

Markov models, in their various incarnations, have for long formed the backbone of diverse domains such as telecommunication (convolution codes) (Viterbi, 1967), biology (sequence analysis, gene prediction, protein structure prediction) (Chu et al., 2004; Gotoh, 2018), natural language processing (language modeling, tagging, chunking) (Heafield et al., 2013), speech (automatic recognition) (Bengio, 2003), finance (time series analysis) (Bulla & Bulla, 2006), computer vision (gesture recognition) (Wang et al., 2006) and web (traffic analysis) (Felzenszwalb et al., 2003). An $n^{th}$ order Markov model assumes that the conditional distribution of next state at any time $i$ depends only on the current state and the previous $n-1$ states, i.e.,

$$P(z_i|z_1, \ldots, z_{i-1}) = P(z_i|z_{i-n}, \ldots, z_{i-1}) \forall i.$$  

In several practical scenarios, the states are not directly accessible but need to be inferred or decoded from the observations that are dependent on states. For instance, in tagging applications (Altun et al., 2003), each state pertains to a part-of-speech tag (e.g. noun, adjective) and each word $w_i$ in an input sentence $w = (w_1, \ldots, w_T)$ needs to be labeled with a probable tag $z_i$ that might have generated the word. Therefore, it is natural to endow each state with a distribution over the possible tokens. For example, $n^{th}$ order hidden Markov models (HMM) (Rabiner, 1990) and $(n+1)$-gram language models (Heafield et al., 2013) assume the joint distribution $P(z_1, \ldots, z_T, w_1, \ldots, w_T)$ of states $z$ and observations $w$ factorizes as the product

$$P(z, w) = \prod_{i=1}^{T} P(z_i|z_{i-n}, \ldots, z_{i-1}) P(w_i|z_i),$$

where $z_{-n+1}, \ldots, z_0$ are dummy states, and the transition distributions $P(z_i|z_{i-n}, \ldots, z_{i-1})$ and emission distributions $P(w_i|z_i)$ are estimated from the data. When the transition distributions do not change with $i$, the model is called time-homogeneous, otherwise it is non-stationary, time-varying or non-homogeneous (Ocana-Riola, 2005; Perez-Ocon et al., 2001; Chenand & Zhou, 2011). Given observations $\tilde{w}$ of length $\tilde{T}$, the decoding problem in these models is to infer a most probable sequence or path of states

$$\tilde{z} \in \arg \max_{z_1, \ldots, z_{\tilde{T}}} P(z, \tilde{w}) = \arg \max_{z_1, \ldots, z_{\tilde{T}}} \log P(z, \tilde{w}).$$

Note that decoding is akin to inference in structured prediction settings (Taskar et al., 2003; Tschantaridis et al., 2004). Decoding is of paramount interest in other Markov models too, e.g., that employ parameters $\theta$ and define feature vectors $\phi$ using the entire observation sequence, see e.g., maximum entropy Markov models (MEMM) (McCallum et al., 2000) and conditional random fields (CRF) (Lafferty et al., 2001; Peng et al., 2009). The decoding task in all these models can be expressed in the form

$$\tilde{z} \in \arg \max_{z_1, \ldots, z_{\tilde{T}}} \sum_{i=1}^{\tilde{T}} R_i(z_i|z_{i-n}, \ldots, z_{i-1}),$$  \hspace{1cm} (1)

where we have omitted the dependence on observations $\tilde{w}$, and specified the reward functions $R_i$ in Table 1.

The Viterbi algorithm (Viterbi, 1967) is a standard method for solving problems of the form (1). However, the algorithm cannot output any states in the most probable path until it has stored and computed best paths up to sequence length $\tilde{T}$ for each state. Therefore, the algorithm cannot be applied to real-time applications, or systems that have low latency requirements, and prove that no online algorithm can perform significantly better than our algorithms.

\footnote{Massachusetts Institute of Technology (MIT). Correspondence to: <tamarp@alum.mit.edu>.}
Table 1. Expressing standard Markov models in the reward form.

| Model       | Reward $\tilde{R}_i(\cdot | i)$ at time $i$ |
|-------------|---------------------------------------------|
| 1-HMM       | $\log P(z_i | z_{i-1}) + \log P(w_i | z_i)$ |
| $(n + 1)$-GRAM | $\log P(z_{i-n}, \ldots, z_{i-1}) + \log P(w_i | z_i)$ |
| $n$-CRF     | $\theta^\top \phi(z_{i-n,i-1}, z_i, w, i)$ |
| $n$-MEMM    | $\log \frac{\exp(\theta^\top \phi(z_{i-n,i-1}, z_i, w, i))}{\sum_{z'_i} \exp(\theta^\top \phi(z_{i-n,i-1}, z'_i, w, i))}$ |

latency and storage requirements, e.g., resource constrained platforms such as IoT devices (Gupta et al., 2017; Kumar et al., 2017; Garg et al., 2018). Besides, the algorithm is not suitable for critical scenarios such as patient monitoring, or intrusion detection in network streams and credit card fraud monitoring where any diameter following the onset of a suspicious activity might be detrimental (Narasimhan et al., 2006). Finally, detecting interleaved coding regions or exons, as and when they are encountered, in massive gene sequences would facilitate drug discovery.

A lot of effort has thus been invested into speeding up the Viterbi algorithm or reducing its memory footprint. Some prominent recent approaches include fast matrix multiplication (Cairo et al., 2016), compression and storage reduction for HMM (Šrámek et al., 2007; Churbanov & Winters-Hilt, 2008; Lifshits et al., 2009), and heuristics such as beam search or simulated annealing (Kaji et al., 2010; Daumé & Marcu, 2005). However, these methods do not guarantee reduction in latency since they still need to process the rewards for all time steps. Moreover, improving Viterbi is provably hard (Backurs & Tzamos, 2017). (Narasimhan et al., 2006) introduce some algorithms to deal with latency in first order models, however, they make assumptions on the input that may not necessarily hold. Besides, none of these algorithms work for higher order and non-homogeneous Markov settings. Our work draws inspiration from (Jayram et al., 2001), who introduced discounting based algorithms for near-optimal online server allocation in what may be viewed as a first order HMM setting that requires transitions between any pair of states. Again, their work does not deal with higher order, and ergodic\(^1\) models whose states are not fully connected.

Our contributions

In this work, we introduce efficient and almost optimal deterministic and randomized algorithms for problems of the form (1) under latency constraints. Our bounds hold when the rewards $R_i$ depend on time $i$ (non-homogeneous settings), or even when they are presented to our algorithms in an adversarial or adaptive fashion. Our algorithms may be employed in all settings where the Viterbi algorithm might be used, and as explained in Section 1, several others where it cannot. Thus, our work would potentially widen the scope of and expedite scientific discovery in numerous fields that rely critically on efficient Markov decoding.

We provide a complete resolution to the decoding problem by establishing rigorous lower bounds in both deterministic and randomized settings for ergodic Markov models, and proving that no online algorithm can perform significantly better. Moreover, our bounds are finite, and thus hold for any positive latency (i.e. not only asymptotically), and the quality of our algorithms improves with increase in the permissible latency. In particular, our algorithms essentially match the lower bounds for sufficiently large latency.

Beyond our theoretical analysis, we sketch geometric or visual intuition into several of our proofs to clearly delineate the role of latency $L$, order $n$, and diameter $\Delta$ (i.e. the minimum number of steps required to move between any pair of states). In particular, our design of constructions toward proving lower bounds in a setting predicated on interplay of several heterogeneous variables is especially novel, both conceptually and technically. We believe similar constructions will find use elsewhere, e.g., in motivating new online algorithms and establishing combinatorial bounds.

2. Overview of our results

We first introduce some notation. We define $[a, b] \triangleq (a, a + 1, \ldots, b)$ and $[N] \triangleq (1, 2, \ldots, N)$. Likewise, $y_{[N]} \triangleq (y_1, \ldots, y_N)$ and $y_{[a,b]} \triangleq (y_a, \ldots, y_b)$. We will denote the last $n$ states visited by the online algorithm at time $i$ by $y_{[i-n,i-1]}$, and those by the optimal offline algorithm by $y^*_{[i-n,i-1]}$. Defining non-negative reward functions $R_i = \tilde{R}_i + p$ by adding a sufficiently large positive number $p$ to each reward, we note from (1) that the optimal sequence of states in our setting over a time horizon $T$ is

$$\arg \max_{y_1, \ldots, y_T} \sum_{i=1}^T R_i(y_i | y_{[i-n,i-1]}).$$

We use $OPT$ to denote the total reward accumulated by the optimal offline algorithm, and $ON$ to denote that by the online algorithm. We evaluate the performance of any online algorithm in terms of its competitive ratio $\rho$, which is defined as the ratio $OPT/ON$. That is,

$$\rho = \frac{\sum_{i=1}^T R_i(y^*_i | y^*_{[i-n,i-1]})}{\sum_{i=1}^T R_i(\hat{y}_i | y_{[i-n,i-1]})}.$$

Clearly, $\rho \geq 1$. Our goal is to design online algorithms

\(^1\) A Markov model is ergodic if any state is reachable from any other state in a finite number of steps $N$. The minimum value of this number, denoted by $\Delta$, is called the diameter of the model.
that have competitive ratio close to 1. For randomized algorithms, we analyze the ratio obtained by taking expectation of the total online reward over its internal randomness.

The performance of any online algorithm depends on the order \(n\), latency \(L\), and diameter \(\Delta\). Table 2 provides a summary of our results. Note that our algorithms are asymptotically optimal in \(L\). For the finite \(L\) case, we first consider the fully connected first order models. Our randomized algorithm matches the lower bound even\(^2\) with \(L = 1\) since we may set \(\epsilon\) arbitrarily close to 0. Note that just with \(L = 1\), our deterministic algorithm achieves a competitive ratio 4, and this ratio reduces further as \(L\) increases. Moreover our ratio rapidly approaches the lower bound with increase in \(L\). Finally, in the general setting, our algorithms are almost optimal when \(L\) is sufficiently large compared to \(\Delta = \Delta + n - 1\), which we call the effective diameter.

The rest of the paper is organized as follows. We first introduce and analyze our deterministic Peek Search algorithm for homogeneous settings in Section 3. We show that the algorithm extends to non-homogeneous Markov settings in Section 4. We then introduce the Randomized Peek Search algorithm in Section 5. We propose the deterministic Peek Reset algorithm, in Section 6, that performs better than deterministic Peek Search for large \(L\). We present the lower bounds in Section 7. Finally, we outline an efficient dynamic program to implement Peek Search in Section 8. We sketch the main intuition underlying our results, and defer the details of our proofs to the Supplementary material.

### 3. Peek Search

Let \(\gamma \in (0, 1)\) be a discount factor. The Peek Search algorithm peeks into the future observations and repeats the following procedure at each time \(i\). First, it finds a path of length \(L + 1\) emanating from the current state that fetches maximum discounted reward. The discounted reward on any path is computed by scaling down the \(\ell\)th edge, \(\ell \in \{0, \ldots, L\}\), on this path by \(\gamma^\ell\). Then, the algorithm moves to the first state of this path and repeats the procedure at time \(i + 1\). Note that at time \(i + 1\), the algorithm need not continue with the second edge on the optimal discounted path computed at previous time step, and is free to choose an alternative path. Formally, at time \(i\), the algorithm computes \(\hat{y}_i \triangleq (\hat{y}_i^0, \hat{y}_i^1, \ldots, \hat{y}_i^L)\) that maximizes the following over valid paths \(y = (y_1, \ldots, y_{i+L})\):

\[
R(y_i|\hat{y}_{i-n,i-1}) + \gamma R(y_{i+1}|\hat{y}_{i+1}, y_{i+1})
\]

receives \(R(y_{i+1}|\hat{y}_{i+1})\) and sets the next state \(\hat{y}_{i+1} = \gamma^0\).

Note that we have dropped the subscript \(i\) from \(R\), since in the homogeneous settings, the reward functions do not change with time \(i\). We optimize \(\gamma\) for each \(R\), and \(\gamma\) approaches 1 asymptotically. Intuitively, \(\gamma\) indicates the confidence of the online algorithm in the best discounted path: \(\gamma\) grows as \(L\) increases, and thus a high value of \(\gamma\) indicates that the path computed at a time \(i\) may be worth tracing at times subsequent to \(i + 1\) as well. We make this intuition precise in the following result for the setting \(n = 1, \Delta = 1\).

**Theorem 1.** The competitive ratio of Peek Search on first order Markov models with unit diameter, using peek \(L \geq 1\),

\[
\rho \leq \left(1 + \frac{1}{L}\right) \sqrt[\gamma]{L + 1}.
\]

**Proof.** (Sketch.) The analysis hinges on two important facts. Since the online algorithm chooses a path that maximizes the total discounted reward over next \((L + 1)\) steps, it is guaranteed to fetch all of the discounted reward pertaining
We complete the analysis for Peek Search by extending the Markov models of order $n$ with the maximizing path computed at the previous time step.

Theorem 2. The competitive ratio of Peek Search on Markov models of order $n \geq 1$ with unit diameter, using peek $L \geq n$ is

$$
\rho \leq \frac{L + 1}{L - n + 1} \frac{(L-n+1)}{L+1} \sqrt{\frac{L + 1}{n}} = 1 + \Theta \left( \frac{\log L}{L - n + 1} \right).
$$

Proof. (Sketch.) The online algorithm may jump to any state on the optimal offline path in one step. However, the reward at each state depends on the previous $n$ states, and so the online algorithm may have to wait additional $n - 1$ steps before it could trace the remaining optimal path.

We complete the analysis for Peek Search by extending the previous analysis to ergodic models with arbitrary $\Delta$.

Theorem 3. The competitive ratio of Peek Search on Markov models of order $n \geq 1$ with diameter $\Delta \geq 1$, using peek $L \geq \Delta + n - 1$ is bounded as

$$
\rho \leq \frac{L + 1}{L - \Delta - n + 2} \sqrt{\frac{L + 1}{\Delta + n - 1}} = 1 + \Theta \left( \frac{\log L}{L - \Delta - n + 2} \right).
$$

Proof. (Sketch.) As explained in Fig. 1, when $\Delta > 1$, the online algorithm may have to forfeit rewards on at most $\Delta$ steps, in order to join the optimal path. Again, it might have to wait additional $n - 1$ steps as discussed above.

4. Non-homogeneous Markov Models

We next prove that the guarantees on Peek Search extend to the non-homogeneous settings even when the rewards may be adversarially chosen. For this setting, we require a mild assumption that the rewards revealed as part of any peek do not change during the duration of the peek.

Theorem 4. The competitive ratio of Peek Search on non-homogeneous (time-varying) Markov models of order $n \geq 1$ with diameter $\Delta \geq 1$, using peek $L$ is bounded as

$$
\rho \leq \frac{L + 1}{L - \Delta - n + 2} \sqrt{\frac{L + 1}{\Delta + n - 1}} = 1 + \Theta \left( \frac{\log L}{L - \Delta - n + 2} \right),
$$

provided the reward associated with any transition does not change for (at least) $L + 1$ steps from the time it is revealed as peek information to the online algorithm.

Proof. (Sketch.) The proof combines the analysis for ho-
mogeneous settings with the condition that rewards do not change during each peek window. This is sufficient to ensure near optimal performance of the online algorithm since it recomputes the likely path at each step.

\section{Randomized Peek Search}

We now introduce the Randomized Peek Search algorithm. The algorithm first selects a reset point $\ell$ uniformly at random from $\{1, 2, \ldots, L + 1\}$. This number is a private information for the online algorithm and is not revealed. The randomized algorithm recomputes the optimal non-discounted path (which corresponds to $\gamma = 1$) of length $(L + 1)$, once every $L + 1$ steps, at each time $i * (L + 1) + \ell$, and follows this path for next $L + 1$ steps without any updates. We have the following result.

\begin{theorem}
Randomized Peek Search achieves, in expectation over its internal randomness, on Markov models of order $n$ with diameter $\Delta$, using peek $L$, a competitive ratio
\begin{equation}
\rho \leq 1 + \frac{\Delta + n - 1}{L + 1 - (\Delta + n - 1)} = 1 + \Theta\left(\frac{1}{L - \Delta + 1}\right).
\end{equation}
\end{theorem}

\begin{proof}
(Sketch.) Since it maximizes the non-discounted reward, for each random $\ell$, the online algorithm receives at least as much reward as the optimal offline algorithm minus the reward on at most $\tilde{\Delta}$ steps every $L + 1$ steps.
\end{proof}

\section{Peek Reset}

We now present the deterministic Peek Reset algorithm, which performs better than Peek Search when the latency $L$ is large. Like Randomized Peek Search, Peek Reset recomputes a best non-discounted path and takes multiple steps on this path. However, the number of steps taken is not fixed to $L + 1$ but may vary in each phase. Specifically, let $(i)$ denote the time at which phase $i$ begins. The algorithm follows, in phase $i$, a sequence of states $\tilde{y}(i) \triangleq (\tilde{y}(i), \tilde{y}(i)+1, \ldots, \tilde{y}(T_i-1))$ that maximizes the follow-
We now state our lower bounds on the performance of any algorithm:

\[ f(y) \leq R(y(i)\hat{y}(i)_{n,(i)-1}) + \sum_{j=1}^{T_i-1} R(y(i+j)\hat{y}(i)_{n,j,(i)-1}; y(i),(i)+j) + \sum_{j=n}^{T_i-1} R(y(i+j)\hat{y}(i)_{n,j,(i)+j}) \]

where \( T_i \) is chosen arbitrarily from the set

\[ \arg \min_{t \in [(i)+L/2+1,(i)+L]} \max_{y(i-n)\ldots y_n} R(y(t-n,t-1)) . \]

Then, the next phase \((i+1)\) begins at time \( T_i \). We have the following result on the performance of Peek Reset.

**Theorem 6.** The competitive ratio of Peek Reset on Markov models of order \( n \) with diameter \( \Delta \), using peek \( L \),

\[ \rho \leq 1 + \frac{2(\Delta + n)(\Delta + n - 1)}{L - 8(\Delta + n - 1) + 1} = 1 + \Theta \left( \frac{1}{L - 8\Delta + 1} \right) . \]

**Proof.** Peek Reset may be loosely viewed as a derandomization of Randomized Peek Search. The algorithm gives up on at most \( L \) steps every \( L + 1 \) steps, however these steps are cleverly selected. Note that \( T_i \) is chosen from the interval \([(i)+L/2+1,(i)+L]\), which contains steps from both phases \((i)\) and \((i+1)\). Thus, the algorithm essentially peaks into phase \((i+1)\) before deciding on the number of steps to be taken in phase \((i)\).

### 7. Lower Bounds

We now state our lower bounds on the performance of any deterministic and any randomized algorithm in the general non-homogeneous Markov setting.

**Theorem 7.** The competitive ratio of any deterministic online algorithm, using peek \( L \), on \( n \)-th order time-varying Markov models that have diameter \( \Delta \) is greater than

\[ 1 + \frac{\Delta}{L} \left( 1 + \frac{\Delta + L - 1}{(\Delta + L - 1)^2 + \Delta} \right) . \]

In particular, when \( n = 1 \), \( \Delta = 1 \), the ratio is larger than

\[ 1 + \frac{1}{L} + \frac{1}{L^2 + 1} . \]

**Proof.** (Sketch.) The proof revolves around a \( \Delta \)-dimensional prismatic polytope construction, where each vertex corresponds to a state. The intuition is sketched in Fig. 2 for the specific setting of \( \Delta = 3 \).

### 8. Efficient Dynamic Programs

Naively computing a best path by enumerating all paths of length \( \ell \) would be computationally prohibitive since the number of such paths is exponential in \( \ell \). Fortunately, we can design efficient dynamic programs to address this issue.

Specifically, let \( S^{(\ell)}_i \), \( \ell \in \{0, \ldots, L\} \) be all state sequences...
of length $\ell + 1$ that start from state at time $i$. Thus, for instance, $S_{i}^{(0)}$ contains all states $y_i$ that can be reached in one step. We also denote the reward resulting from paths in the set $S_{i}^{(\ell)}$ that end in sequence $y_{n,a,b}$ by $P_{i}(\ell, y_{a,b})$. Our efficient implementation for Peek Search in outlined in Algorithm 1. We now establish the proof of correctness, and the efficiency of Algorithm 1.

**Theorem 9.** Algorithm 1 computes a best $\gamma$-discounted path for the next $L + 1$ steps, in $n^{th}$ order Markov models, in time $O(L|K|^{n})$, where $K$ is the set of states.

**Proof.** (Sketch.) The proof is based on the fact that, for every $\ell \in [L]$, the reward on the optimal discounted path of length $\ell$ can be recursively computed from that on an optimal path of length $\ell - 1$ using $O(|K|^{n})$ computations.

It is immediate that our other two algorithms, Randomized Peek Search and Peek Reset, can compute their appropriate paths efficiently by using Algorithm 1 as a subroutine. For instance, Randomized Peek Search could invoke Algorithm 1 at each reset point with $\gamma$ set to 1, and follow this path until the next reset point.

**References**

Altun, Y., Tsochantaridis, I., and Hofmann, T. Hidden markov support vector machines. In *ICML*. 2003.

Backurs, A. and Tzamos, C. Improving viterbi is hard: Better runtimes imply faster clique algorithms. In *ICML*, volume 70, pp. 311–321, 2017.

Bengio, S. An asynchronous hidden markov model for audio-visual speech recognition. In *NIPS*, pp. 1237–1244, 2003.

Bulla, J. and Bulla, I. Stylized facts of financial time series and hidden semi-markov models. *Comput. Stat. Data Anal.*, 51(4), 2006.

Cairo, M., Farina, G., and Rizzi, R. Decoding hidden markov models faster than viterbi via online matrix-vector (max, +)-multiplication. In *AAAI*, pp. 1484–1490, 2016.

Chenand, B. and Zhou, X.-H. Non-homogeneous markov process models with informative observations with an application to alzheimer’s disease. *Biometrical Journal*, 53(3):444–463, 2011.

Chu, W., Ghahramani, Z., and Wild, D. L. A graphical model for protein secondary structure prediction. In *ICML*, 2004.

Churbanov, A. and Winters-Hilt, S. Implementing em and viterbi algorithms for hidden markov model in linear memory. *BMC Bioinformatics*, 9(1):224, 2008.

Daumé, III, H. and Marcu, D. Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML*, pp. 169–176, 2005.

Felzenszwalb, P. F., Huttenlocher, D. P., and Kleinberg, J. M. Fast algorithms for large-state-space hms with applications to web usage analysis. In *NIPS*, pp. 409–416, 2003.

Garg, V. K., Dekel, O., and Xiao, L. Learning small predictors. In *NIPS*, 2018.

Gotoh, O. Modeling one thousand intron length distributions with fitild. *Bioinformatics*, 34(19):3258–3264, 2018.

Gupta, C., Suggala, A. S., Goyal, A., Simhadri, H. V., Paranjape, B., Kumar, A., Goyal, S., Udupa, R., Varma, M., and Jain, P. ProtoNN: compressed and accurate kNN for resource-scarce devices. In *ICML*, pp. 1331–1340, 2017.

Heafield, K., Pouzyrevsky, I., Clark, J. H., and Koehn, P. Scalable modified kneser-ney language model estimation. In *Association for Computational Linguistics (ACL)*, pp. 690–696, 2013.

Jayram, T. S., Kimbrel, T., Krauthgamer, R., Schieber, B., and Sviridenko, M. Online server allocation in a server farm via benefit task systems. In *STOC*, pp. 540–549, 2001.

Kaji, N., Fujiwara, Y., Yoshinaga, N., and Kitsuregawa, M. Efficient staggered decoding for sequence labeling. In *Association for Computational Linguistics (ACL)*, pp. 485–494, 2010.

Kumar, A., Goyal, S., and Varma, M. Resource-efficient machine learning in 2 kb ram for the internet of things. In *ICML*, pp. 1935–1944, 2017.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pp. 282–289, 2001.

Lifshits, Y., Mozes, S., Weimann, O., and Ziv-Ukelson, M. Speeding up hmm decoding and training by exploiting sequence repetitions. *Algorithmica*, 54(3):379–399, 2009.

McCallum, A., Freitag, D., and Pereira, F. C. N. Maximum entropy markov models for information extraction and segmentation. In *ICML*, 2000.

Narasimhan, M., Viola, P., and Shilman, M. Online decoding of markov models under latency constraints. In *ICML*, pp. 657–664, 2006.
We provide here detailed proofs of all the theorems stated in the main text. Note that the set $M$ may have more than one path that maximizes the discounted sum. Breaking ties arbitrarily, let the online algorithm choose $\hat{y}_i \in \{\hat{y}_i, \hat{y}_{i+1}, \ldots, \hat{y}_{i+L}\} \in M_i$ (and reach the state $\hat{y}_i$). Let $\{y^*_t \mid t \in [T]\}$ be the optimal path over the entire horizon. Since $\Delta = 1$, one of the candidate paths considered by the online algorithm is the optimal segment $(y^*_i, y^*_{i+1}, \ldots, y^*_{i+L})$. Since $\hat{y}_i \in M_i$, we must have

$$R(\hat{y}_i|\hat{y}_{i-1}) + \gamma R(\hat{y}_i^*|\hat{y}_i) + \sum_{j=2}^{L} \gamma^j R(\hat{y}_i^j|\hat{y}_i^{j-1})$$

$$\geq R(y^*_i|\hat{y}_{i-1}) + \sum_{j=1}^{L} \gamma^j R(y^*_i|\hat{y}_{i+j-1})$$

$$\geq \sum_{j=1}^{L} \gamma^j R(y^*_i|\hat{y}_{i+j-1}) , \quad (3)$$

where the last inequality follows since all rewards are non-negative, and thus in particular, $R(y^*_i|\hat{y}_{i-1}) \geq 0$.

An alternate path considered by the online algorithm is $(\tilde{y}_i^1, \ldots, \tilde{y}_i^{L-1}, \tilde{y}_i^L)$, where $(\tilde{y}_i^1, \ldots, \tilde{y}_i^{L-1})$ are the last $L$ steps of the path $\tilde{y}_i \in M_{i-1}$ (i.e. the path chosen at time $i-1$) and $\tilde{y}_i^L$ is an arbitrary valid transition from state $\tilde{y}_i^{L-1}$. Again, since this transition fetches a non-negative reward, we must have

$$R(\tilde{y}_i^1|\hat{y}_{i-1}) + \gamma R(\tilde{y}_i^2|\hat{y}_i) + \sum_{j=2}^{L} \gamma^j R(\tilde{y}_i^j|\hat{y}_i^{j-1})$$

$$\geq R(\tilde{y}_i^{L-1}|\hat{y}_{i-1}) + \sum_{j=1}^{L-1} \gamma^j R(\tilde{y}_i^{L-1}|\hat{y}_i^{L-1}) . \quad (4)$$

Multiplying (3) by $1 - \gamma$ and (4) by $\gamma$, and adding the resulting inequalities, we get

$$R(\hat{y}_i|\hat{y}_{i-1}) + \gamma R(\hat{y}_i^*|\hat{y}_i) + \sum_{j=2}^{L} \gamma^j R(\hat{y}_i^j|\hat{y}_i^{j-1})$$

$$\geq \sum_{j=1}^{L} (1 - \gamma) \gamma^j R(y^*_i|\hat{y}_{i+j-1}) + \gamma R(\hat{y}_i^1|\hat{y}_i-1)$$

$$+ \gamma R(\tilde{y}_i^{L-1}|\hat{y}_{i-1}) + \sum_{j=1}^{L-1} \gamma^{j+1} R(\tilde{y}_i^{L-1}|\hat{y}_i^{L-1})$$

$$= \sum_{j=1}^{L} (1 - \gamma) \gamma^j R(y^*_i|\hat{y}_{i+j-1}) + \gamma R(\tilde{y}_i^{L-1}|\hat{y}_{i-1})$$

$$+ \sum_{k=2}^{L} \gamma^k R(\tilde{y}_i^{k-1}|\hat{y}_i^{k-1}) , \quad (5)$$

Supplementary Material

We provide here detailed proofs of all the theorems stated in the main text.

A. First order Markov models with $\Delta = 1$

**Theorem 1.** The competitive ratio of Peek Search on first order Markov models with unit diameter, using peek $L \geq 1$ is bounded as $$\rho \leq \left( 1 + \frac{1}{L} \right) \sqrt[L]{L+1} .$$

**Proof.** Recall that at each time step $i$, our online algorithm solves the following optimization problem over variables $y \in S(i, L)$, i.e. the set of valid paths of length $L + 1$ that emanate from the state at time $i$:

$$M_i = \arg \max_{\hat{y} \in S(i, L)} R(\hat{y}_i|\hat{y}_{i-1}) + \sum_{j=1}^{L} \gamma^j R(\hat{y}_i+j|\hat{y}_{i+j-1}).$$

Ocana-Riola, R. Non-homogeneous markov processes for biomedical data analysis. *Biometrical Journal*, 47:369–376, 2005.

Peng, J., Bo, L., and Xu, J. Conditional neural fields. In *NIPS*, pp. 1419–1427, 2009.

Perez-Ocon, R., Ruiz-Castro, J. E., and Gamiz-Perez, M. L. Non-homogeneous markov models in the analysis of survival after breast cancer. *Journal of the Royal Statistical Society, Series C*, 50:111–124, 2001.

Rabiner, L. R. Readings in speech recognition. chapter A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pp. 267–296. 1990.

Šrámek, R., Brejová, B., and Vinař, T. On-line viterbi algorithm for analysis of long biological sequences. In *Algorithms in Bioinformatics*, pp. 240–251. Springer Berlin Heidelberg, 2007.

Taskar, B., Guestrin, C., and Koller, D. Max-margin markov networks. In *NIPS*, pp. 25–32, 2003.

Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. Support vector machine learning for interdependent and structured output spaces. In *ICML*, 2004.

Viterbi, A. J. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Information Theory*, 13(2):260–269, 1967.

Wang, S. B., Quattoni, A., Morency, L.-P., and Demirdjian, D. Hidden conditional random fields for gesture recognition. In *CVPR*, pp. 1521–1527, 2006.
where the last inequality follows from a change of variable, namely, \(k = j + 1\). Summing across all time steps \(i\),

\[
\sum_i R(\hat{y}_i|\hat{y}_{i-1}) + \sum_i \left( \gamma R(\hat{y}_i^1|\hat{y}_i) + \sum_{j=2}^L \gamma^j R(\hat{y}_i^j|\hat{y}_i^{j-1}) \right)
\]

\[
\geq \sum_i \sum_{j=1}^L (1 - \gamma) \gamma^j R(\hat{y}_i^j|\hat{y}_i^{j-1})
\]

\[
+ \sum_i \left( \gamma R(\hat{y}_i^1|\hat{y}_i) + \sum_{j=2}^L \gamma^j R(\hat{y}_i^j|\hat{y}_i^{j-1}) \right)
\].

Without loss of generality, we can assume that all transitions between states in the first \(L + 1\) time steps and the last \(L + 1\) steps fetch zero reward. Equivalently, we can introduce a dummy start state that transitions to itself \(L\) times with zero reward and produces a fake output in each transition, and then makes a zero reward transition into the true start state, whence actual decoding happens for \(T\) steps followed by repeated transitions into a dummy end state that again fetches zero reward. Then, by telescoping over \(i\), we have

\[
\sum_i R(\hat{y}_i|\hat{y}_{i-1}) \geq \sum_i \sum_{j=1}^L (1 - \gamma) \gamma^j R(\hat{y}_i^j|\hat{y}_i^{j-1}).
\]

Defining a variable \(s = i + j\), and interchanging the two sums, we note that the right side turns out to be

\[
(1 - \gamma) \sum_{j=1}^L \gamma^j \sum_s R(\hat{y}_s^j|\hat{y}_s^{j-1}).
\]

That is, every reward subsequent to \(L + 1\) steps appears with discount \(\gamma, \gamma^2, \ldots, \gamma^L\). Summing the geometric series, we note that the ratio of the total reward obtained by the optimal offline algorithm to that by the online algorithm, i.e. the competitive ratio \(\rho\) is at most \(\gamma^{-1}(1 - \gamma^L)^{-1}\). The result follows by setting \(\gamma = \sqrt[1/(L+1)]{1}\).

\[
\square
\]

**B. \(n^{th}\) order Markov models with \(\Delta = 1\)**

**Theorem 2.** The competitive ratio of Peek Search on Markov models of order \(n \geq 1\) with unit diameter, using peek \(L \geq n\) is bounded as

\[
\rho \leq \frac{L + 1}{L - n + 1} \left( \frac{(L-n+1)}{L+1} \right) \left( \frac{L+1}{n} \right)
\]

\[
= 1 + \Theta \left( \frac{\log L}{L - n + 1} \right).
\]

**Proof.** For \(n = 1\), the result follows from Theorem 1. Therefore, we will assume \(n > 1\). The online algorithm finds, at time \(i\), some \(\hat{y}_i = (\hat{y}_i^1, \hat{y}_i^1, \ldots, \hat{y}_i^L)\) that maximizes the following objective over valid paths \(y = (y_i, \ldots, y_{i+L})\):

\[
f(y) \triangleq \sum_{n-1} R(y_{i-n-i-1}) + \sum_{j=1}^L \gamma^j R(y_{i+j}|y_{i+n+j-i-1}, y_{i+j-i-1}) + \sum_{j=n}^L \gamma^j R(y_{i+j}|y_{i+n+j-i-1}, y_{i+j-i-1}).
\]

One candidate path for the online algorithm (a) makes a transition to \(y_i^*\) worth \(R(y_i^*|y_{i-n-i-1}) \geq 0\), (b) then follows the sequence of \(n - 1\) states \(y_{i+1}^*, y_{i+2}^*, \ldots, y_{i+L}^*\) where transition \(i + j, j \in [n - 1]\) is worth

\[
\gamma^j R(y_{i+j}^*|y_{i+n+j-i-1}, y_{i+j-i-1}) \geq 0,
\]

and (c) finally follows a sequence \(L - n + 1\) states \(y_{i+1}^*, y_{i+2}^*, \ldots, y_{i+L}^*\) where transition \(i + j, j \in \{n, n + 1, \ldots, L\}\) is worth

\[
\gamma^j R(y_{i+j}^*|y_{i+n+j-i-1}, y_{i+j-i-1}).
\]

Since \(\hat{y}_i \in \operatorname{argmax}_y f(y)\) and the rewards in (a) and (b) are all non-negative, we must have

\[
f(\hat{y}_i) \geq \sum_{j=n}^L \gamma^j R(y_{i+j}^*|y_{i+n+j-i-1}, y_{i+j-i-1}).
\]

Another option available with the online algorithm is to continue following the path selected at time \(i - 1\) for \(L\) steps, and then make an additional arbitrary transition with a non-negative reward. Therefore, we must also have

\[
f(\hat{y}_i) \geq R(\hat{y}_{i-1}^1|y_{i-n-i-1}) + \sum_{j=1}^n \gamma^j R(\hat{y}_{i-1}^j|y_{i-n+j-i-1}, y_{i-1}^j) + \sum_{j=n}^{L-1} \gamma^j R(\hat{y}_{i-1}^j|y_{i-n+j-i-1}, y_{i-1}^j).
\]

Multiplying (6) by \(1 - \gamma\) and (7) by \(\gamma\), and adding the
resulting inequalities, we get

\[
f(\tilde{y}_i) \geq (1 - \gamma) \sum_{j=n}^{L} \gamma^j R(y_{i+j}^* | y_{i+j-n,i+j-1}^*) + \gamma R(\tilde{y}_{i-1}^* | y_{i-j-n,i-j-1}) + \sum_{j=2}^{n} \gamma^j R(\tilde{y}_{i-1}^* | y_{i-j-n+1,i-j}, y_{i}^{[j-1]}_i) + \sum_{j=n+1}^{L} \gamma^j R(\tilde{y}_{i-1}^* | y_{i-n-j+1,i-j-1}^*) .
\]  

(8)

Expanding the terms of \( f(\tilde{y}_i) \), we note

\[
f(\tilde{y}_i) = R(\tilde{y}_i | y_{i-n,i-1}) + \gamma R(\tilde{y}_i^* | y_{i-n+1,i}) + \sum_{j=2}^{n} \gamma^j R(\tilde{y}_i^* | y_{i-j-n,i-j}, y_{i}^{[j-1]}_i) + \sum_{j=n+1}^{L} \gamma^j R(\tilde{y}_i^* | y_{i-n-j+1,i-j-1}) .
\]  

(9)

Substituting \( f(\tilde{y}_i) \) from (9) in (8), assuming zero padding as in the proof of Theorem 1, and summing over all time steps \( i \), we get the inequality

\[
\sum_i \sum_{j=n}^{L} R(\tilde{y}_i | y_{i-n,i-1}) \geq \sum_i \sum_{j=n}^{L} (1 - \gamma) \gamma^j R(y_{i+j}^* | y_{i+j-n,i+j-1}^*) .
\]

Defining \( s = i + j \) and interchanging the two sums, we note that the right side simplifies to

\[
(1 - \gamma) \sum_{j=n}^{L} \gamma^j \sum_s R(y_{i+j}^* | y_{i+j-n,i+j-1}^*) .
\]

The sum of this geometric series is given by \( \gamma^n - \gamma^{L+1} \), and thus setting

\[
\gamma = \left( \frac{n}{L + 1} \right)^{1/(L-n+1)},
\]

we immediately conclude that the total reward obtained by the optimal offline algorithm exceeds that of the online algorithm by at most \( \Theta \left( \frac{\log L}{L - n + 1} \right) \) times the reward of the online algorithm, and hence we have the following bound on the competitive ratio

\[
\rho \leq 1 + \Theta \left( \frac{\log L}{L - n + 1} \right).
\]

C. \( n^{th} \) order Markov models with finite \( \Delta \)

**Theorem 3.** The competitive ratio of Peek Search on Markov models of order \( n \geq 1 \) with diameter \( \Delta \geq 1 \), using peek \( L \geq \Delta + n - 1 \) is bounded as

\[
\rho \leq \frac{L + 1}{L - \Delta - n + 2} \sqrt{\frac{L + 1}{\Delta + n - 1}} + 1 + \Theta \left( \frac{\log L}{L - \Delta - n + 2} \right).
\]

Proof. For \( \Delta = 1 \), the result follows from Theorem 2. Therefore, we will assume \( \Delta > 1 \). As in the proof of Theorem 2, the online algorithm finds at time \( i \) some \( \tilde{y}_i \triangleq (\tilde{y}_i, \ldots, \tilde{y}_{i+L}) \) that maximizes the following objective over valid paths \( y = (y_i, \ldots, y_{i+L}) \):

\[
f(y) \triangleq R(y_i | y_{i-n,i-1}) + \sum_{j=1}^{n-1} \gamma^j R(y_{i+j} | y_{i+n+j,i-1}, y_{i,j+1}) + \sum_{j=n+1}^{L} \gamma^j R(y_{i+j} | y_{i+n+j,i+j}) .
\]

Since \( \Delta > 1 \), the online algorithm may not be able to jump to the desired state on the optimal offline path in one step unlike in the setting of Theorem 2, and may require \( \Delta \) steps in the worst case.\(^3\) Therefore, let \( (\tilde{y}_i, \ldots, \tilde{y}_{i+\Delta-2}) \) be an intermediate sequence of states before the online algorithm could transition to the optimal offline path and then follow the optimal algorithm for the remaining steps. Therefore, we have \( f(\tilde{y}_i) \)

\[
\begin{align*}
\geq & \quad R(\bar{y}_i | y_{i-n,i-1}) \\
& + \sum_{j=1}^{\Delta-2} \gamma^j R(\tilde{y}_{i+j} | y_{i-n+j,i-1}, y_{i,i+j-1}) \\
& + \gamma^{\Delta-1} R(y_{i+\Delta-1} | y_{i+n-\Delta-1,i+\Delta-2}) \\
& + \sum_{j=\Delta}^{\Delta+n-2} \gamma^j R(y_{i+j} | y_{i+n+j-\Delta-1,i+j-\Delta-2}) \\
& + \sum_{j=\Delta+n-1}^{L} \gamma^j R(y_{i+j}^* | y_{i+j-n,i+j-1}^*) \\
\geq & \quad \sum_{j=\Delta+n-1}^{L} \gamma^j R(y_{i+j}^* | y_{i+j-n,i+j-1}^*),
\end{align*}
\]

\(^3\)The online algorithm may require less than \( \Delta \) steps depending on its current state, however, we perform a worst case analysis and therefore, our result holds even if fewer than \( \Delta \) steps may suffice to reach the optimal path at some point during the execution of the online algorithm.
where we have leveraged the non-negativity of rewards to obtain the last inequality.

Another option available with the online algorithm is to continue following the path selected at time \( i - 1 \) for \( L \) steps, and then make an additional arbitrary transition with a non-negative reward. Therefore, we must also have

\[
f(\hat{y}_i) \geq R(\hat{y}_{i-1}, \hat{y}_{i-n,i-1}) + \sum_{j=1}^{n-1} \gamma^j R(\hat{y}_{i-1}^{[j]}, y_{i-n+j,i-1}^{[j]}) + \sum_{j=n}^{L-1} \gamma^j R(\hat{y}_{i-1}^{[j]}, y_{i-n+j,i-1}^{[j]}) .
\]

Multiplying (10) by \( 1 - \gamma \) and (11) by \( \gamma \), and adding the resulting inequalities, we get

\[
f(\hat{y}_i) \geq (1 - \gamma) \sum_{j=\Delta+1}^{\Delta+n} \gamma^j R(y_{i+j+n,i+j}) + \gamma R(\hat{y}_{i-1}^{[1]}, y_{i-n,i-1}^{[1]}) + \sum_{j=n}^{L-1} \gamma^{j+1} R(\hat{y}_{i-1}^{[j+1]}, y_{i-n+j,i-1}^{[j+1]}) .
\]

Expanding \( f(\hat{y}_i) \), telescoping over \( i \), and defining \( s = i + j \) as in Theorem 2, we get that the total reward accumulated by the online algorithm is at least

\[
(1 - \gamma) \sum_{j=\Delta+1}^{\Delta+n} \gamma^j = (\gamma^{n+\Delta-1} - \gamma^{L+1})
\]
times the total reward collected by the optimal offline algorithm. Setting

\[
\gamma = \left(\frac{L+1}{L-\Delta-n+2}\right)^{L-n-1} \frac{\Delta+n-1}{L+1} ,
\]

we immediately get

\[
\rho \leq \frac{L+1}{L-\Delta-n+2} \left(\frac{L+1}{L-\Delta-n+2}\right)^{(L-n+2)/(L+1)} \frac{\Delta+n-1}{\Delta+n-1} \left(\frac{L+1}{L-\Delta-n+2}\right)^{\log L} .
\]

Note that with peek search, the per-step reward of the online algorithm in the sense that it may not receive any reward in these steps. However, the remaining steps fetch nearly the same reward as the optimal offline algorithm. In particular, the competitive ratio \( \rho \) gets arbitrarily close to 1, as \( L \) is set sufficiently large compared to \( \Delta + n \). That is, the performance of the online algorithm is asymptotically optimal in the peek \( L \).

### D. Non-homogeneous Markov Models

Note that there might be multiple transitions between a pair of states during any peek window. Such transitions are considered distinct and may indeed have different rewards during the same window. We only require that the non-discounted reward committed for every transition is “honored” at all times during the window.

**Theorem 4.** The competitive ratio of Peek Search on non-homogeneous (time-varying) Markov models of order \( n \geq 1 \) with diameter \( \Delta \geq 1 \), using peek \( L \geq \Delta+1 \) is bounded as

\[
\rho \leq \frac{L+1}{L-\Delta-n+2} \left(\frac{L+1}{L-\Delta-n+2}\right)^{(L-n+2)/(L+1)} \frac{\Delta+n-1}{\Delta+n-1} \left(\frac{L+1}{L-\Delta-n+2}\right)^{\log L} ,
\]

provided the reward associated with any transition does not change for (at least) \( L+1 \) steps from the time it is revealed as peek information to the online algorithm.

**Proof.** The online algorithm maximizes the following non-stationary objective at time \( i \):

\[
f_i(y) \triangleq \sum_{j=1}^{n-1} \gamma^j R_i(y_{i+j}, y_{i-n+j,i-1}) + \sum_{j=n}^{L-1} \gamma^{j+1} R_i(y_{i+j}, y_{i-n+j,i-1}) + \gamma R_i(y_{i-1}, y_{i-n,i-1}) .
\]

where the subscript \( i \) shown with \( f \) and \( R \) indicates that the rewards associated with a transition may change with time
i. Proceeding as in the proof of Theorem 3, we get

\[
\begin{align*}
    f_i(\hat{y}_i) &\geq (1 - \gamma) \sum_{j=\Delta+n-1}^L \gamma^j R_i(y^*_n y^*_i j | y^*_{[i+j-n,i+j-1]} \\
&\quad + \gamma R_{i-1}(\hat{y}_{i-1}^* | \hat{y}_{i-n,i-1}) \\
&\quad + \sum_{j=1}^{n-1} \gamma^{j+1} R_i(y^*_{[i+j-n,j-i]} | y^*_{i-n,j-i-1}) \hat{y}^*_j \\
&\quad + \sum_{j=n}^{L-1} \gamma^{j+1} R_{i-1}(\hat{y}_{i-1}^* | \hat{y}_{i-n,j-i-1}) \\
&\quad + \sum_{j=n}^L \gamma^{j+1} R_i(y^*_{[i+j-n+1,j]} | y^*_{i-n,j-i-1}) \\
\end{align*}
\]

However, by our assumption, we can equivalently write

\[
\begin{align*}
    f_i(\hat{y}_i) &\geq (1 - \gamma) \sum_{j=\Delta+n-1}^L \gamma^j R_i(y^*_n y^*_i j | y^*_{[i+j-n,i+j-1]} \\
&\quad + \gamma R_{i-1}(\hat{y}_{i-1}^* | \hat{y}_{i-n,i-1}) \\
&\quad + \sum_{j=1}^{n-1} \gamma^{j+1} R_i(y^*_{[i+j-n,j-i]} | y^*_{i-n,j-i-1}) \hat{y}^*_j \\
&\quad + \sum_{j=n}^{L-1} \gamma^{j+1} R_{i-1}(\hat{y}_{i-1}^* | \hat{y}_{i-n,j-i-1}) \\
&\quad + \sum_{j=n}^L \gamma^{j+1} R_i(y^*_{[i+j-n+1,j]} | y^*_{i-n,j-i-1}) \\
\end{align*}
\]

Expanding \( f(\hat{y}_i) \), summing over all \( i \), and defining \( s = i+j \) as in Theorem 2, we get

\[
\begin{align*}
    \sum_i R_i(\hat{y}_i | \hat{y}_{[i-n,i-1]}) &\geq \sum_i \sum_{j=\Delta+n-1}^L (1 - \gamma) \gamma^j R_i(y^*_{[i+j-n, i+j-1]} \\
&= (1 - \gamma) \sum_{j=\Delta+n-1}^L \gamma^j \sum_s R_s(y^*_n | y^*_{[i-n,s-1]} \\
&= (1 - \gamma) \sum_{j=\Delta+n-1}^L \gamma^j \sum_s R_s(y^*_n | y^*_{[i-n,s-1]}),
\end{align*}
\]

where we have again made use of the fact that reward due to any transition does not change for \( L+1 \) steps once revealed. The rest of the proof is identical to the analysis near the end of proof for Theorem 3.

---

### E. Randomized Peek Search

**Theorem 5.** Randomized Peek Search achieves, in expectation over its internal randomness, on Markov models of order \( n \geq 1 \) with diameter \( \Delta \geq 1 \), a competitive ratio

\[
\begin{align*}
    \Delta &\geq \Delta + n - 1 \\
&\leq 1 + \frac{\Delta + n - 1}{L + 1 - (\Delta + n - 1)} \\
&= 1 + \Theta \left( \frac{1}{L - \Delta + 1} \right),
\end{align*}
\]

when peek is available, and the effective diameter is

\[
\Delta \triangleq \Delta + n - 1. 
\]

**Proof.** Recall that the randomized algorithm recomputes and follows a path that optimizes the non-discounted reward once every \( L+1 \) steps (which we call an epoch). Since the starting or reset point is chosen uniformly at random from \( \{1, 2, \ldots, L+1\} \), we define a random variable \( X \) that denotes the outcome of an unbiased \( (L+1) \)-sided dice.

Let \( (X = x) \) be any particular realization. Then, during epoch \( i \), one option available with the online algorithm is to give up rewards in steps

\[
[i \ast (L+1) + x, i \ast (L+1) + x + \Delta + n - 2]
\]

to reach a state on the optimal offline path and follow it for the remainder of the epoch. Let \( ON_x \) denote the total reward of the online randomized algorithm conditioned on realization \( x \), and let \( OPT \) be the optimal reward. Then, letting \( r^*_t \) be the reward obtained by the optimal offline algorithm at time \( t \) we must have

\[
ON_x \geq OPT - \sum_i \sum_{t=i(L+1)+x}^{i(L+1)+x+\Delta+n-2} r^*_t. \quad (12)
\]

Since \( x \) is chosen uniformly at random from \( [L+1] \), we also note the expected value of the second term on the right

\[
\begin{align*}
&= \mathbb{E}_X \left( \sum_i \sum_{t=i(L+1)+x}^{i(L+1)+x+\Delta+n-2} r^*_t | X = x \right) \\
&= \frac{1}{L+1} \sum_{x=1}^{L+1} \sum_i \sum_{t=i(L+1)+x}^{i(L+1)+x+\Delta+n-2} r^*_t \\
&= \frac{1}{L+1} \sum_{x=1}^{L+1} \sum_i \sum_{z=0}^{\Delta+n-2} r^*_z | x \geq i(L+1) + x \\
&= \frac{1}{L+1} \sum_{z=0}^{\Delta+n-2} \left( \sum_i \sum_{x=1}^{L+1} r^*_z | x \geq i(L+1) + x \right) \\
&= \frac{1}{L+1} \sum_{z=0}^{\Delta+n-2} OPT \\
&= \frac{\Delta + n - 1}{L + 1} OPT.
\end{align*}
\]
Therefore, taking expectations on both sides of (12),

\[ \mathbb{E}_x(ON_x) \geq \text{OPT} \left( 1 \frac{\Delta + n - 1}{L + 1} \right), \]

whence the result follows immediately.

\[ \square \]

**F. Peek Reset**

**Theorem 6.** The competitive ratio of Peek Reset on Markov models of order \( n \geq 1 \) with diameter \( \Delta \geq 1 \), using peek \( L \) is bounded as

\[ \rho \leq 1 + \frac{2(\Delta + n)(\Delta + n - 1)}{L - 8(\Delta + n - 1) + 1} \]

\[ = 1 + \Theta \left( \frac{1}{L - 8\Delta + 1} \right), \]

where the effective diameter

\[ \hat{\Delta} \triangleq \Delta + n - 1. \]

**Proof.** We will assume for now that \( L \) is a multiple of \( 4(\Delta + n - 1) \). Recall that the Peek Reset algorithm works in phases of varying lengths, and takes multiple steps in each phase. Let \( \tau(i) \) denote the time at which phase \( i \) begins. Then, the algorithm follows, in phase \( i \), a sequence of states \( \hat{y}(i) \triangleq (\hat{y}(i), \hat{y}(i+1), \ldots, \hat{y}_{T_i-1}) \) that maximizes the following objective over valid paths \( y = (y(i), \ldots, y_{T_i-1}) \):

\[ f(y) \triangleq R(y(i), y_{y(i)-n}, \ldots, y_{y(i)-1}) \]

\[ + \sum_{j=1}^{n-1} R(y(i)+j, y_{y(i)+j-n}, \ldots, y_{y(i)+j-1}) \]

\[ + \sum_{j=n}^{T_i-1} R(y(i)+j, y_{y(i)+j-n}, \ldots, y_{y(i)+j-1}), \]

where \( T_i \) is chosen arbitrarily from the set

\[ \arg \min_{t \in [(i) + L/2 + 1, (i) + L]} \max_{y_{y(i)-n}, \ldots, y_{y(i)-1}} R(y(i), y_{t-n}, \ldots, y_{t-1}). \]

We define the corresponding reward

\[ x_{T_i} = \min_{t \in [(i) + L/2 + 1, (i) + L]} \max_{y_{y(i)-n}, \ldots, y_{y(i)-1}} R(y(i), y_{t-n}, \ldots, y_{t-1}). \]

Consider the portion of the path traced by the online algorithm from \( \hat{y}(i)+L/2 \) to \( \hat{y}_{T_i-1} \). Total number of edges on this path is \( z_i = T_i - ((i) + L/2 + 1) \). We claim that the reward resulting from this sequence is at least

\[ a_i = \frac{z_i - (\Delta + n - 1)}{\Delta + n} \cdot x_{T_i}. \]

This is true since, by definition of \( x_{T_i} \), at each time \( t \in [(i) + L/2 + 1, (i) + L] \), there is a state \( y_{t-1} \) such that moving to some state \( y_t \) will fetch a reward at least \( x_{T_i} \). Note that a maximum of \( \Delta + n - 1 \) steps might have to be wasted to reach another state that fetches at least \( x_{T_i} \). Thus, a reward of \( x_{T_i} \) is guaranteed every \( \Delta + n \) steps. While there are \( z_i \) steps in this sequence, at most \( \Delta + n - 1 \) steps may be left over as residual edges that do not fetch any reward if \( z_i \) is not a multiple of \( \Delta + n \). Since the online algorithm optimized for total non-discounted reward, it must have considered this alternative sequence of steps for the interval pertaining to \( z_i \).

Next consider the portion traversed by the online algorithm from \( \hat{y}_{T_i} \) to \( \hat{y}(i)+L \) in the next phase \( (i + 1) \). This phase starts at time \( T_i \). By an argument analogous to previous paragraph, the online algorithm collects from this sequence an aggregate no less than

\[ b_i = \frac{(i) + L - T_i - (\Delta + n - 1)}{\Delta + n} \cdot x_{T_i}. \]

Thus, the reward accumulated by the online algorithm in these two portions is at least

\[ a_i + b_i = \frac{L - 4(\Delta + n - 1)}{2(\Delta + n)} \cdot x_{T_i}. \]

Summing over all phases, we note that the total reward gathered by the online algorithm is

\[ \sum_i f(\hat{y}(i)) \geq \frac{L - 4(\Delta + n - 1)}{2(\Delta + n)} \sum_i x_{T_i}. \quad (13) \]

Let \( f(y^*(i)) \) be the reward collected by the optimal offline algorithm in phase \( i \). Since the online algorithm optimizes for the total reward, one possibility it considers is to forego reward in the first \( (\Delta + n - 1) \) steps in each phase in order to traverse the same sequence as the optimal algorithm in the remaining steps. Thus, we must have

\[ \sum_i f(\hat{y}(i)) \geq \sum_i f(y^*(i)) - (\Delta + n - 1) \sum_i x_{T_i}. \quad (14) \]

Combining (13) and (14), we note for even \( L \)

\[ \frac{\sum_i f(y^*(i))}{\sum_i f(\hat{y}(i))} \leq 1 + \frac{2(\Delta + n)(\Delta + n - 1)}{L - 4(\Delta + n - 1)}. \]

Accounting for \( L \) that are not multiples of \( 4(\Delta + n - 1) \), we conclude the competitive ratio of Peek Reset

\[ \rho \leq 1 + \frac{2(\Delta + n)(\Delta + n - 1)}{L - 8(\Delta + n - 1) + 1}. \]

\[ \square \]
G. Lower Bounds

Theorem 7. The competitive ratio of any deterministic online algorithm, that has access to peek $L$, for decoding with $n^{th}$ order (time-varying) Markov models that have diameter $Δ$ is greater than

$$1 + \frac{\bar{Δ}}{L} \left( 1 + \frac{\bar{Δ} + L - 1}{(\bar{Δ} + L - 1)^2 + \Delta} \right),$$

where the effective diameter $\bar{Δ} = Δ + n - 1$. In particular, when $n = 1$ and $Δ = 1$, the ratio is larger than

$$1 + \frac{1}{L} + \frac{1}{L^2 + 1}.$$  

Proof. We motivate the main ideas of the proof for the specific setting of $n = 2$ and unit diameter. The extension to general $n$ and unit diameter is then straightforward. Finally, we conjure an example to prove the lower bound for arbitrary $n$ and $Δ$ via a prismatic polytope construction.

First consider the case $n = 2$ and $Δ = 1$. We design a $3 \times (L + 3)$ matrix initialized as shown below: each row corresponds to a different state, each column corresponds to a time, "?" indicates that the corresponding entry is not known since it lies outside the current peek window of length $L + 1$, and $a > 0$ is a variable that will be optimized later.

$$
\begin{bmatrix}
0 & a & a & \ldots & a & ? & ? \\
0 & 1 & a & a & \ldots & a & ? & ? \\
0 & 1 & a & a & \ldots & a & ? & ? \\
\end{bmatrix}_{(L-1) \text{ terms}} (15)
$$

The box in front of the first entry indicates that the online algorithm made a transition to state 1 from a dummy start state "*" and is ready to make a decision in the current step $t = 0$ whether to continue staying in state 1, or transition to either state 2 or 3. At time $t = 0$, the rewards for the next $L + 1$ steps are identical, so without loss of generality, let the online algorithm choose the first state, get 0 as reward, and move to the next time $t = 1$. An additional column is revealed and we get the following snapshot.

$$
\begin{bmatrix}
0 & 1 & a & a & \ldots & a & 0 & ? \\
0 & 1 & a & a & \ldots & a & 2a & ? \\
\end{bmatrix}_{(L-1) \text{ terms}} (16)
$$

Since $n = 2$, we may enforce the following second order Markov dependencies for $t \geq 1$: any state $s \in \{1, 2, 3\}$ yields zero reward unless the previous two states $s', s'' \in \{1, 2, 3, *\}$ were such that $s' \in \{*, s\}$ and $s'' = s$. If this condition is true, then the algorithm receives the current entry pertaining to $s$ as the reward. In other words, other than the special case of dummy start state being one of the states, the algorithm receives the reward only if $s$ is same as the previous two states.

Suppose the online algorithm selects state 1 again at $t = 1$. Then it collects reward 1, and another column is revealed as shown below.

$$
\begin{bmatrix}
0 & 1 & a & a & \ldots & a & 0 & 0 \\
0 & 1 & a & a & \ldots & a & 2a & 0 \\
\end{bmatrix} (17)
$$

In this scenario, the maximum reward the online algorithm can fetch, during its entire execution, is at most $1 + \frac{(L - 1)a}{1}$. To see this note that this is exactly the reward the algorithm gets if it sticks to state 1 at all subsequent times $t$. If, however, it were to jump to any other state and continue with it for at least one step, then it would lose rewards in successive steps due to second order dependency, for a total loss of reward $2a$. All other possibilities incur a loss greater than $2a$. This loss offsets the additional $2a$ reward available with states other than 1. On the other hand, the offline algorithm would select a state $s \in \{2, 3\}$ from the very beginning, and thus receive $1 + \frac{(L + 1)a}{2}\text{total reward}$. The competitive ratio in this scenario, therefore, turns out to be

$$r_1 = 1 + \frac{(L + 1)a}{1 + (L - 1)a} = 1 + \frac{2a}{1 + (L - 1)a}.$$  

Suppose instead the online algorithm transitions to some state $s \in \{2, 3\}$ at $t = 1$. We assume without loss of generality that the algorithm transitions to state 2. The last column is then revealed as follows.

$$
\begin{bmatrix}
0 & 1 & a & a & \ldots & a & 0 & 0 \\
0 & 1 & a & a & \ldots & a & 2a & a \\
\end{bmatrix}_{(L-1) \text{ terms}} (18)
$$

Note that the online algorithm loses on rewards 1 and $a$ in successive steps due to transition. The maximum total reward possible in this case is $La$ regardless of whether the online algorithm makes a transition to other states, or sticks with state 2 subsequently. The offline algorithm, in contrast,}

\footnote{Note that since our objective here is to prove a lower bound, we would like the competitive ratio to be as high as possible. It might be tempting to set a reward larger than $a$ for state 3 in the last column. That would imply both the online and the offline algorithms could receive an additional reward worth $a$. This, however, would not improve the competitive ratio for the simple reason that for positive $x$, $y$, and $c$, $x + c > x + y$ only if $x < y$ (while since $r_2 > 1$, we instead have $x > y$).}
would receive all rewards available in state 3. Thus, the ratio in this scenario is
\[ r_2 = \frac{1 + (L + 2)a}{La} = 1 + \frac{1 + 2a}{La}. \]
Combining the two cases, the competitive ratio of the online algorithm is at least \( \min \{ r_1, r_2 \} \), and thus we could set \( r_1 = r_2 \) and solve for \( a \).

We can extend this analysis to the general \( n \geq 1 \) setting with unit diameter easily. We design a \( 3 \times (L + 3) \) matrix with the same row initialization as in (15). Also, we assume that prior to time \( t = 0 \), only zero reward transitions were available between some dummy states\(^5\) for both the online and the offline algorithms. We denote the set of these dummy states by **. We enforce the following \( n^{th} \) order Markov dependencies for \( t \geq 1 \): any state \( s \in [m] \) yields zero reward unless the previous \( n \) states were same as \( s \) or had a prefix consisting only of states in ** followed by \( s \) in the remaining time steps. If this condition is satisfied, the algorithm receives the current entry pertaining to \( s \) as reward.

The evolution of the reward matrix is as follows. Assuming state 1 was selected at \( t = 0 \), we let column \( L + 2 \) have all entries in rows 2 and 3 to \( na \) (instead of 2\( na \) that we set in (16)) at \( t = 1 \). Finally, if the online algorithm selects state 1 at \( t = 1 \), we set the last column to all zeros at time \( t = 2 \) as in (17); otherwise, we set first two entries in the last column to 0, and \( a \) in the last row as in (18).

Reasoning along the same lines as before, the competitive ratio of the online algorithm is at least
\[
\min \left\{ 1 + \frac{na}{1 + (L - 1)a} , 1 + \frac{1 + na}{La} \right\}. \tag{19}
\]
We set
\[
1 + \frac{na}{1 + (L - 1)a} = 1 + \frac{1 + na}{La},
\]
whereby
\[
a = \frac{n + L - 1 + \sqrt{(n + L - 1)^2 + 4n}}{2n}.
\]
Substituting this value for \( a \) in (19), and leveraging that
\[
a < \frac{n + L - 1}{n} + \frac{1}{n + L - 1},
\]
we note the competitive ratio is at least
\[
1 + \frac{n}{L} \left(1 + \frac{n + L - 1}{(n + L - 1)^2 + n}\right). \tag{20}
\]

The foregoing analysis may be visualized geometrically in terms of a triangle, with each vertex corresponding to a state. The rewards for initial \( L + 1 \) steps are all same, and thus the online algorithm does not have preference for any state initially. Without loss of generality, as soon as it selects state 1 (with all rewards at time \( t = 0 \) being 0), the rewards for time step \( L + 2 \) are chosen at \( t = 1 \) such that states 2 and 3 would fetch reward \( na \) while state 1 will fetch none. The online algorithm could either stay with state 1 and get a sub-optimal total reward or jump to an adjacent vertex or state, which would not yield reward for \( n \) steps.

We now extend this analysis to accommodate any finite \( \Delta \geq 1 \). Toward that goal, we consider a \( \Delta \)-dimensional prismatic polytope\(^6\) with a triangular base (i.e. having 3 vertices). Each vertex of the polytope corresponds to a state, and the maximum distance between two vertices is exactly \( \Delta \). Moreover, for every vertex there is some vertex at distance \( d \) for each \( d \in [\Delta] \). The polytope is completely symmetric with respect to all the vertices, and we again set rewards for the first \( L + 1 \) steps at all vertices to be the same as before.

Without loss of generality, we again assume that the online algorithm starts at some state 1 (arbitrary labeled). At the next time step, the reward at all vertices that are at a distance \( d \) from this vertex is set to \((n + d - 1)a\). Thus, the vertices adjacent to state 1 have reward \( na \) in column \( L + 2 \) since they lie at distance \( d = 1 \), while the reward for state 1 in this column is 0. Thus, the maximum reward is available at distance \( d = \Delta \) from state 1, however, the online algorithm will need to make \( \Delta \) steps to reach such a state, and then wait another \( n - 1 \) steps before availing this reward. Thus, effectively, \( \Delta = n + \Delta - 1 \) steps are wasted that the offline algorithm could fully exploit due to prescience. Proceeding along the same lines as before, and replacing \( n \) with \( \Delta \) in (20), we conclude that the competitive ratio of any deterministic online algorithm on our construction is at least
\[
1 + \frac{\Delta}{L} \left(1 + \frac{\Delta + L - 1}{(\Delta + L - 1)^2 + \Delta}\right).
\]

\[\square\]

**Theorem 8.** For any \( \epsilon > 0 \), the competitive ratio of any randomized online algorithm, that has access to peek \( L \), for decoding with \( n^{th} \) order (time-varying) Markov models that have unit diameter is at least \( 1 + \frac{(1 - \epsilon)n}{L + cn} \).

In general, for diameter \( \Delta \), the competitive ratio is at least
\[
1 + \frac{(2^{\Delta - 1}[1/\epsilon] - 1)n}{2^{\Delta - 1}[1/\epsilon]/(L + n)}. \tag{21}
\]
\(^6\) Note that a \( d \)-dimensional prismatic polytope is constructed from two \((d - 1)\)-dimensional polytopes, translated into the next dimension.
We design a matrix with \( \left[ \frac{1}{e} \right] \) rows and \( L+2 \) columns. The first column consists of all zeros, the next \( L \) columns contain all ones, and the last column contains all zeros except one randomly chosen row that contains \( n \). We again enforce the Markov dependency structure described in the proof of Theorem 7 for all states (or rows) in \( \left[ \frac{1}{e} \right] \).

The optimal offline algorithm knows beforehand which row \( q \) contains \( n \) in the other column, and thus collects a total reward \( L + n \). On the other hand, any randomized online algorithm chooses this row at \( t = 0 \) with only the probability \( e \). Selecting any other row at \( t = 0 \) may fetch a maximum reward of \( L \) accounting for all the possibilities including sticking to this row subsequently, or moving to \( q \) in one or more transitions. Since the randomized algorithm is assigned at time \( t = 0 \) with the remaining probability \( (1-e) \) to some row other than \( q \), its expected reward cannot exceed
\[
e \cdot (L + n) + (1 - e) \cdot L = L + en.
\]

Thus, when \( \Delta = 1 \), the competitive ratio for any randomized online algorithm is at least \( \frac{L + n}{L + en} \) as claimed.

For the general setting, we consider a \( \Delta \)-dimensional prismatic polytope with the base containing \( \left[ \frac{1}{e} \right] \) vertices. In addition to the usual prismatic polytope topology (assuming bidirectional edges between any pair of adjacent vertices), we add edges so that vertices on each face are strongly connected, i.e., directed edges in both directions connect all pairs of vertices that lie on a face. The polytope contains \( u = 2^{\Delta-1} \left[ \frac{1}{e} \right] \) states in total. We design a matrix having these many rows and \( L+2 \) columns as before. Any randomized online algorithm has only a \( 1/u \) probability of getting the maximum possible \( L + n \) reward (due to selecting \( q \) and sticking with it), and must forfeit a reward no less than \( n \) with the remaining probability. Thus, the expected reward cannot exceed
\[
\frac{(L + n)}{u} + (1 - 1/u) \cdot L = L + n/u,
\]
while the maximum possible reward is \( L + n \). Thus the competitive ratio is at least \( \frac{L + n}{L + n/u} \) which simplifies to the result stated in the problem statement.

\[ \Box \]

### H. Efficient Dynamic Programs

**Theorem 9.** Algorithm 1 computes a best \( \gamma \)-discounted path for the next \( L + 1 \) steps, in the \( n \)th order Markov models, in time \( O(L|K|^n) \), where \( K \) is the set of states.

**Proof.** Let \( S_i(\ell, \ell, v[a,b]) \) denote the set of all valid paths of length \( \ell + 1 \) emanating from the state \( \hat{y}_{i-1} \) at time \( i \), where \( \ell \in \{0, 1, \ldots, L\} \), that end in the state sequence \((v_a, \ldots, v_b)\). Thus, e.g., if the directed edge \( e = (\hat{y}_{i-1}, v_n) \) exists, then
\[
S_i(0, v[2, n]) = \begin{cases} \{e\} & \text{if } v_{n-j} = \hat{y}_{i-j}, \forall j \in [n-2] \\ \emptyset & \text{otherwise} \end{cases}
\]
where \( \emptyset \) is the empty set. We also denote the reward resulting from valid paths of length \( \ell + 1 \) that end in sequence \( v[a,b] \) by \( \Pi_i(\ell, v[a,b]) \). That is,
\[
\Pi_i(\ell, v[a,b]) = \max_{(y_1, \ldots, y_{\ell+1}) \in S_i(\ell, v[a,b])} f_\ell(y_i[i,i+\ell]),
\]
where we define \( f_\ell(y_i[i,i+\ell]) \) recursively as
\[
\begin{align*}
R(y_i)[\hat{y}_{i-n,i-1}] & = \ell = 0, \\
( f_{\ell-1}(y_i[i,i+\ell-1]) & + \gamma^\ell R(y_{i+\ell}[\hat{y}_{i-n+\ell,i-1}], y_{i,i+\ell-1}) ) & \ell \in [n-1], \\
& + \gamma^\ell R(y_{i+\ell}[y_{i+n+\ell,i+\ell-1}], y_{i,i+\ell-1}) & \ell \in [n,L].
\end{align*}
\]

Note that \( f_L(y_i[i,L]) \) is precisely the objective optimized by Peek Search at time \( i \). Now, suppose \( \ell \in [n,L] \). Then, for any end sequence \( v[2,n] \),
\[
\Pi_i(\ell, v[2,n]) = \max_{y_i[i,i+\ell]} f_\ell(y_i[i,i+\ell]) = \max_{y_i[i,i+\ell] \in S_i(\ell, v[1,n])} f_\ell(y_i[i,i+\ell]),
\]
which may be expanded as
\[
\max_{y_i[i,i+\ell] \in S_i(\ell, v[1,n])} \left( f_{\ell-1}(y_i[i,i+\ell-1]) + \gamma^\ell R(y_{i+\ell}[y_{i-n+\ell,i+\ell-1}], y_{i,i+\ell-1}) \right)
\]
where we simply write \( S_i(\ell, v) \) instead of \( y_i[i,i+\ell] \in S_i(\ell, v) \) in order to improve readability at the expense of abuse of notation.
A similar analysis can be done for $\ell \in [n - 1]$. Then, the maximizing path of length $\ell + 1$ is in the set

$$\arg \max_{v_{[n, n]}} \max_{v_{[n \setminus 1], K}} \left( \Pi_i(\ell - 1, v_{[n \setminus 1]}) + \gamma^\ell R(v_n | v_{[n \setminus 1]}) \right),$$

which requires checking $O(|K|^n)$ values for $v_{[n]}$. We conclude by noting that $\Pi_i$ is updated for each $\ell \in \{0, \ldots, L\}$, and thus the total complexity is $O(L|K|^n)$.

---

Footnote: In addition to backpointer information that is required to determine a maximizing path as in the Viterbi algorithm once the construction of table for bookkeeping $\Pi_i$ is completed. Construction of table requires $O(L|K|^n)$ time which dominates the $O(L)$ time required for computing the path from the backpointers.