Competing with Automata-based Expert Sequences

Mehryar Mohri
Courant Institute and Google Research
New York, NY 10012
mohri@cims.nyu.edu

Scott Yang[*]
D. E. Shaw & Co.
New York, NY 10036
yangs@cims.nyu.edu

Abstract

We consider a general framework of online learning with expert advice where regret is defined with respect to sequences of experts accepted by a weighted automaton. Our framework covers several problems previously studied, including competing against k-shifting experts. We give a series of algorithms for this problem, including an automata-based algorithm extending weighted-majority and more efficient algorithms based on the notion of failure transitions. We further present efficient algorithms based on an approximation of the competitor automaton, in particular n-gram models obtained by minimizing the ∞-Rényi divergence, and present an extensive study of the approximation properties of such models. Finally, we also extend our algorithms and results to the framework of sleeping experts.

1 Introduction

Online learning is a general model for sequential prediction. Within that framework, the setting of prediction with expert advice has received widespread attention [Littlestone and Warmuth, 1994, Cesa-Bianchi and Lugosi, 2006, Cesa-Bianchi et al., 2007]. In this setting, the algorithm maintains a distribution over a set of experts, or selects an expert from an implicitly maintained distribution. At each round, the loss assigned to each expert is revealed. The algorithm incurs the expected loss over the experts and then updates its distribution on the set of experts. Its objective is to minimize its expected regret, that is the difference between its cumulative loss and that of the best expert in hindsight.

[*]Work done at the Courant Institute of Mathematical Sciences.

Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84. Copyright 2018 by the author(s).

However, this benchmark is only significant when the best expert in hindsight is expected to perform well. When that is not the case, then the learner may still play poorly. As an example, it may be that no single baseball team has performed well over all seasons in the past few years. Instead, different teams may have dominated over different time periods. This has led to a definition of regret against the best sequence of experts with k shifts in the seminal work of Herbster and Warmuth [1998] on *tracking the best expert*. The authors showed that there exists an efficient online learning algorithm for this setting with favorable regret guarantees.

This work has subsequently been improved to account for broader expert classes [Gyorgy et al., 2012], to deal with unknown parameters [Monteleoni and Jaakkola, 2003], and has been further generalized [Vovk, 1999, Cesa-Bianchi et al., 2012, Koolen and de Rooij, 2013]. Vovk [1999] and Cesa-Bianchi and Lugosi [2006] showed how the work in Herbster and Warmuth [1998] can be interpreted as an efficient implementation of the classical weighted majority algorithm [Littlestone and Warmuth, 1994] over expert sequences with a specific choice of prior weights. Koolen and de Rooij [2013] described a Bayesian framework for online learning where the learner samples from a distribution of expert sequences and predicts according to the prediction of that expert sequence. In particular, they showed how algorithms designed for k-shifting regret, e.g. [Herbster and Warmuth, 1998, Monteleoni and Jaakkola, 2003], can be interpreted as specific priors in this formulation. Another approach for handling dynamic environments has consisted of designing algorithms that guarantee small regret over any subinterval during the course of play. This notion, coined as *adaptive regret* by Hazan and Seshadhri [2009], has been subsequently strengthened and generalized [Daniely et al., 2015, Adamskiy et al., 2012]. Remarkably, it was shown by Adamskiy et al. [2012] that the algorithm designed by Herbster and Warmuth [1998] is also optimal for adaptive regret. There has also been work deriving guarantees in the bandit setting when the losses are stochastic [Besbes et al., 2014, Wei et al., 2016].

The general problem of online convex optimization in the presence of non-stationary environments has also been stud-

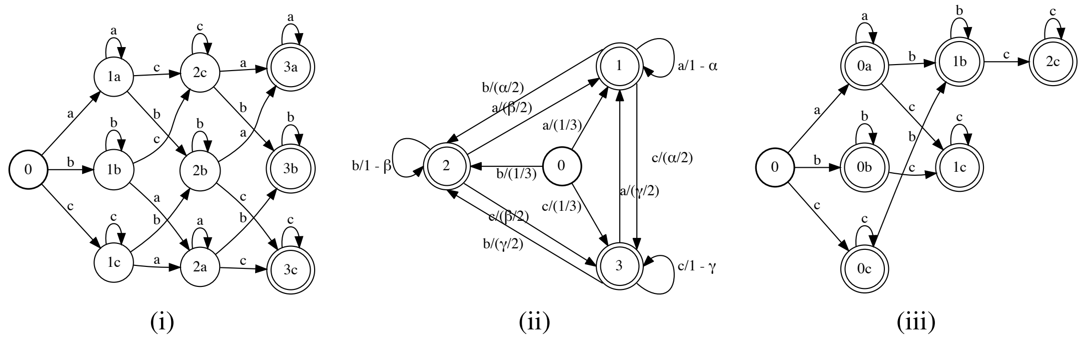

Figure 1: WFAs representing sequences of experts in $\Sigma = \{a, b, c\}$. (i) $\mathcal{C}_{k\text{-shift}}$ with $k = 2$ shifts, all weights are equal to one and not indicated; (ii) $\mathcal{C}_{\text{weighted-shift}}$ with $\alpha, \beta, \gamma \in [0, 1]$; (iii) $\mathcal{C}_{\text{hierarchy}}$ a hierarchical family of expert sequences: the learner must select expert a from the start, can only shift onto b once, and can only shift onto c twice.

ied by many researchers. One perspective has been the design of algorithms that maintain a guarantee against sequences that do not vary too much [Mokhtari et al., 2016, Shahrampour and Jadbabaie, 2016, Jadbabaie et al., 2015, Besbes et al., 2015]. Another assumes that the learner has access to a dynamical model that is able to capture the benchmark sequence [Hall and Willett, 2013]. György and Szepesvári [2016] reinterpreted the framework of Hall and Willett [2013] to recover and extend the results of Herbster and Warmuth [1998].

In this paper, we generalize the framework just described to the case where the learner's cumulative loss is compared to that of sequences accepted by a weighted finite automaton (WFA). This strictly generalizes the notion of k-shifting regret, since k-shifting sequences can be represented by an automaton (see Figure 1), and further extends it to a notion of *weighted regret* which takes into consideration the sequence weights. Our framework covers a very rich class of competitor classes, including WFAs learned from past observations.

Our contributions are mainly algorithmic but also include several new theoretical results and guarantees. We first describe an efficient online algorithm using automata operations that achieves both favorable weighted regret and unweighted regret (Section 3). Next, we present and analyze more efficient solutions based on an approximation of the WFA representing the set of competitor sequences (Section 4), including a specific analysis of approximations via n-gram models both when minimizing the ∞-Rényi divergence and the relative entropy. Finally, we extend the results above to the sleeping expert setting [Freund et al., 1997], where the learner may not have access to advice from all experts at every round (Section 5).

2 Learning setup

We consider the setting of prediction with expert advice over $T \in \mathbb{N}$ rounds. Let $\Sigma = \{a_1, \ldots, a_N\}$ denote a set of N experts. At each round $t \in [T]$, an algorithm $\mathcal{A}$ specifies a probability distribution p_t over Σ, samples an expert i_t from p_t, receives the vector of losses of all experts $\mathbf{l}_t \in [0, 1]^N$, and incurs the specific loss $l_t[i_t]$. A commonly adopted goal for the algorithm is to minimize its static (expected) regret $\text{Reg}_T(\mathcal{A}, \Sigma)$, that is the difference between its cumulative expected loss and that of the best expert in hindsight:

$$\text{Reg}_T(\mathcal{A}, \Sigma) = \max_{x \in \Sigma} \sum_{t=1}^{T} \mathsf{p}_t \cdot \mathbf{l}_t - \sum_{t=1}^{T} l_t[x]. \tag{1}$$

Here, we will consider an alternative benchmark, typically more demanding, where the cumulative loss of the algorithm is compared against the loss of the best sequence of experts $\mathbf{x} \in \Sigma^T$ among those accepted by a weighted finite automaton (WFA) $\mathcal{C}$ over the semiring $(\mathbb{R}_+ \cup \{+\infty\}, +, \times, 0, 1)$.[1] The sequences $\mathbf{x}$ accepted by $\mathcal{C}$, which we will assume to be non-empty, are those which are assigned a positive value by $\mathcal{C}$, $\mathcal{C}(\mathbf{x}) > 0$.[2] We will denote by $K \geq 1$ the cardinality of that set of sequences.

We will take into account the probability distribution q defined by the weights assigned by $\mathcal{C}$ to sequences of length T: $\mathsf{q}(\mathbf{x}) = \frac{\mathcal{C}(\mathbf{x})}{\sum_{\mathbf{x} \in \Sigma^T} \mathcal{C}(\mathbf{x})}$. This leads to the following definition of *weighted regret* at time T given a WFA $\mathcal{C}$:

$$\begin{aligned} &\text{Reg}_T(\mathcal{A}, \mathcal{C}) \qquad (2) \\ &= \max_{\substack{\mathbf{x} \in \Sigma^T \\ \mathcal{C}(\mathbf{x}) > 0}} \left\{ \sum_{t=1}^{T} \mathsf{p}_t \cdot \mathbf{l}_t - \sum_{t=1}^{T} l_t[\mathbf{x}[t]] + \log[\mathsf{q}(\mathbf{x}) K] \right\}, \end{aligned}$$

where $\mathbf{x}[t]$ denotes the tth symbol of $\mathbf{x}$. The presence of the factor K only affects the regret definition by a constant additive term $\log K$ and is only intended to make the last term vanish when the probability distribution q is uniform, i.e. $\mathsf{q}(\mathbf{x}) = \frac{1}{K}$ for all $\mathbf{x}$. The last term in the weighted regret definition can be interpreted as follows: for a given

[1]Thus, the weights in $\mathcal{C}$ are non-negative; the weight of a path is obtained by multiplying the transition weights along that path and the weight assigned to a sequence is obtained by summing the weights of all accepting paths labeled with that sequence.

[2]The weight of a sequence $\mathbf{x}$ is the sum of the weights of all paths in the automaton from the intial state to a final state that are labeled with $\mathbf{x}$.

value of an expert sequence loss $\sum_{t=1}^{T} l_t[\mathbf{x}[t]]$, the regret is larger for sequences $\mathbf{x}$ with a larger probability $\mathsf{q}(\mathbf{x})$.[3] Thus, with this definition of regret, the learning algorithm is pressed to achieve a small cumulative loss compared to expert sequences with small loss and high probability. Notice that when $\mathcal{C}$ accepts only constant sequences, that is sequences $\mathbf{x}$ with $\mathbf{x}[1] = \ldots = \mathbf{x}[T]$ and assigns the same weight to them, then the notion of weighted regret coincides with that of static regret (Formula 1).

We also define the *unweighted regret* $\mathrm{Reg}_T^0(\mathcal{A}, \mathcal{C})$ of algorithm $\mathcal{A}$ at time T given the WFA $\mathcal{C}$ as:

$$\mathrm{Reg}_T^0(\mathcal{A}, \mathcal{C}) = \max_{\substack{\mathbf{x} \in \Sigma^T \\ \mathcal{C}(\mathbf{x}) > 0}} \left\{ \sum_{t=1}^{T} \mathsf{p}_t \cdot \mathsf{l}_t - \sum_{t=1}^{T} l_t[\mathbf{x}[t]] \right\}. \tag{3}$$

The weights of the WFA $\mathcal{C}$ play no role in this notion of regret. When $\mathcal{C}$ has uniform weights, then the unweighted regret and weighted regret coincide.

As an example, the sequences of experts with k shifts studied by Herbster and Warmuth [1998] can be represented by the WFA $\mathcal{C}_{k\text{-shift}}$ of Figure 1(i). Figure 1(ii) shows an alternative weighted model of shifting experts, and Figure 1(iii) shows a hierarchical family of expert sequences.

3 Automata Weighted-Majority algorithm

In this section, we describe a simple algorithm, *Automata Weighted-Majority* (AWM), that can be viewed as an enhancement of the weighted-majority algorithm [Littlestone and Warmuth, 1994] to the setting of experts paths represented by a WFA.[4] We will show that it benefits from favorable weighted and unweighted regret guarantees.

We first present the mathematical calculations and concepts behind the algorithm before providing an efficient implementation. As with standard weighted-majority, AWM maintains a distribution q_t over the set of expert sequences $\mathbf{x} \in \Sigma^T$ accepted by $\mathcal{C}$ at any time t and admits a learning parameter $\eta > 0$. The initial distribution q_1 is defined in terms of the distribution q induced by $\mathcal{C}$ over Σ^T, and q_{t+1} is defined from q_t via an exponential update: for all $\mathbf{x} \in \Sigma^T, t \geq 1$,

$$\begin{aligned} \mathsf{q}_1[\mathbf{x}] &= \frac{\mathsf{q}[\mathbf{x}]^\eta}{\sum_{\mathbf{x} \in \Sigma^T} \mathsf{q}[\mathbf{x}]^\eta}, \\ \mathsf{q}_{t+1}[\mathbf{x}] &= \frac{e^{-\eta\, l_t[\mathbf{x}[t]]} \mathsf{q}_t[\mathbf{x}]}{\sum_{\mathbf{x} \in \Sigma^T} e^{-\eta\, l_t[\mathbf{x}[t]]} \mathsf{q}_t[\mathbf{x}]}, \end{aligned} \tag{4}$$

[3]The choice of an additive weight penalty is motivated by the log loss setting, where the loss is the log probability of a sequence and the penalty can be interpreted as a prior weight. However, a multplicative penalty is also reasonable and could serve as interesting future work.

[4]This algorithm is in fact closer to the EXP4 algorithm [Auer et al., 2002]. However, EXP4 is designed for the bandit setting, so we use the weighted-majority naming convention.

where we denote by $\mathbf{x}[t] \in \Sigma$ the tth symbol in $\mathbf{x}$. q_t induces a distribution p_t over the expert set Σ defined for all $a \in \Sigma$ by

$$\mathsf{p}_t[a] = \frac{\sum_{\mathbf{x} \in \Sigma^T} \mathsf{q}_t[\mathbf{x}] 1_{\mathbf{x}[t]=a}}{\sum_{a \in \Sigma} \sum_{\mathbf{x} \in \Sigma^T} \mathsf{q}_t[\mathbf{x}] 1_{\mathbf{x}[t]=a}}. \tag{5}$$

Thus, $\mathsf{p}_t[a]$ is obtained by summing up the q_t-weights of all sequences with the tth symbol equal to a and normalization. The distributions p_t define the AWM algorithm. Note that in contrast to a standard implementation of the weighted-majority algorithm with a fixed prior distribution, the algorithm here uses an initial distribution q_1 that is defined in terms of q^η and changes with the learning rate.

The following regret guarantees hold for AWM.

Theorem 1. *Let* q *denote the probability distribution over expert sequences of length* T *defined by* $\mathcal{C}$ *and let* K *denote the cardinality of its support. Then, the following upper bound holds for the weighted regret of* AWM*:*

$$\begin{aligned} \mathrm{Reg}_T(\mathrm{AWM}, \mathcal{C}) &\leq \frac{\eta T}{8} + \frac{1}{\eta} \log \left[K^\eta \sum_{\mathbf{x} \in \Sigma^T} \mathsf{q}[\mathbf{x}]^\eta \right] \\ &\leq \frac{\eta T}{8} + \frac{1}{\eta} \log K. \end{aligned}$$

Furthermore, when $K \geq 2$*, for any* $T > 0$*, there exists* $\eta^* > 0$*, decreasing as a function of* T*, such that:*

$$\mathrm{Reg}_T(\mathrm{AWM}, \mathcal{C}) \leq \sqrt{\frac{T H_{\eta^*}(\mathsf{q})}{2}} - H_{\eta^*}(\mathsf{q}) + \log K,$$

where $H_\eta(\mathsf{q}) = \frac{1}{1-\eta} \log \left(\sum_{\mathbf{x} \in \Sigma^T} \mathsf{q}[\mathbf{x}]^\eta \right)$ *is the* η*-Rényi entropy of* q*. The unweighted regret of* AWM *can be upper-bounded as follows:*

$$\mathrm{Reg}_T^0(\mathrm{AWM}, \mathcal{C}) \leq \frac{\eta T}{8} + \frac{1}{\eta} \log K.$$

The proof is an extension of the standard proof for the weighted-majority algorithm and is given in Appendix C. The bound in terms of the Rényi entropy shows that the regret guarantee can be substantially more favorable than standard bounds of the form $O(\sqrt{T \log K})$, depending on the properties of the distribution q. First, since the η-Rényi entropy is non-increasing in η [Van Erven and Harremos, 2014], we have $H_{\eta^*}(\mathsf{q}) \leq H_0(\mathsf{q}) = \log(|\mathrm{supp}(\mathsf{q})|) \leq \log K$. Second, if the distribution q is concentrated on a subset Δ with a small cardinality, $|\Delta| \ll K$, that is $\sum_{\mathbf{x} \notin \Delta} \mathsf{q}[\mathbf{x}]^{\eta^*} < \epsilon(1 - \eta^*) \sum_{\mathbf{x} \in \Delta} \mathsf{q}[\mathbf{x}]^{\eta^*}$ for some $\epsilon > 0$ and for $\eta^* < 1$, then, by Jensen's inequality, the following upper bound holds:

$$\begin{aligned} H_\eta^*(\mathsf{q}) &\leq \frac{1}{1-\eta^*} \log \left(\sum_{\mathbf{x} \in \Delta} \mathsf{q}[\mathbf{x}]^{\eta^*} \right) + \epsilon \\ &\leq \frac{1}{1-\eta^*} \log \left(|\Delta| \left(\frac{1}{|\Delta|} \sum_{\mathbf{x} \in \Delta} \mathsf{q}[\mathbf{x}] \right)^{\eta^*} \right) + \epsilon \\ &\leq \log(|\Delta|) + \epsilon. \end{aligned}$$

Algorithm 1: AutomataWeightedMajority(AWM).

```
Algorithm: AWM($\mathcal{C}$, $\eta$)
$\mathcal{B} \leftarrow \mathcal{C} \cap \mathcal{S}_T$
$\mathcal{A} \leftarrow$ Weight-Pushing($\mathcal{B}^\eta$)
$\boldsymbol{\beta} \leftarrow$ BwdDist($\mathcal{A}$)
$\boldsymbol{\alpha} \leftarrow 0$; $\boldsymbol{\alpha}[I_\mathcal{A}] \leftarrow 1$
for each $e \in E_\mathcal{A}^{0\to1}$ do
    $\mathsf{p}_1[\mathrm{lab}[e]] \leftarrow \mathrm{weight}[e]$.
for $t \leftarrow 1$ to $T$ do
    $i_t \leftarrow$Sample($\mathsf{p}_t$); Play($i_t$); Receive($\mathbf{l}_t$)
    $Z \leftarrow 0$; $\mathbf{w} \leftarrow 0$
    for each $e \in E_\mathcal{A}^{t\to t+1}$ do
        $\mathrm{weight}[e] \leftarrow \mathrm{weight}[e]\, e^{-\eta l_t[\mathrm{lab}[e]]}$
        $\mathbf{w}[\mathrm{lab}[e]] \leftarrow$
          $\mathbf{w}[\mathrm{lab}[e]] + \boldsymbol{\alpha}[\mathrm{src}[e]]\, \mathrm{weight}[e]\, \boldsymbol{\beta}[\mathrm{dest}[e]]$
        $Z \leftarrow Z + \mathbf{w}[\mathrm{lab}[e]]$
        $\boldsymbol{\alpha}[\mathrm{dest}[e]] \leftarrow \boldsymbol{\alpha}[\mathrm{dest}[e]] + \boldsymbol{\alpha}[\mathrm{src}[e]]\, \mathrm{weight}[e]$
    $\mathsf{p}_{t+1} \leftarrow \frac{\mathbf{w}}{Z}$
```

Efficient algorithm. We now present an efficient computation of the distributions p_t. Algorithm 1 gives the pseudocode of our algorithm. We will assume throughout that $\mathcal{C}$ is deterministic, that is it admits a single initial state and no two transitions leaving the same state share the same label. We can efficiently compute a WFA accepting the set of sequences of length T accepted by $\mathcal{C}$ by using the standard intersection algorithm for WFAs (see Appendix A for more detail on this algorithm). Let $\mathcal{S}_T$ be a deterministic WFA accepting the set of sequences of length T and assigning weight one to each (see Figure 2). Then, the intersection of $\mathcal{C}$ and $\mathcal{S}_T$ is a WFA denoted by $\mathcal{C} \cap \mathcal{S}_T$ which, by definition, assigns to each sequence $\mathbf{x} \in \Sigma^T$ the weight

$$(\mathcal{C} \cap \mathcal{S}_T)(\mathbf{x}) = \mathcal{C}(\mathbf{x}) \times \mathcal{S}_T(\mathbf{x}) = \mathcal{C}(\mathbf{x}), \qquad (6)$$

and assigns weight zero to all other sequences. Furthermore, the WFA $\mathcal{B} = (\mathcal{C} \cap \mathcal{S}_T)$ returned by the intersection algorithm is deterministic since both $\mathcal{C}$ and $\mathcal{S}_T$ are deterministic. Next, we replace each transition weight of $\mathcal{B}$ by its η-power. Since $\mathcal{B}$ is deterministic, this results in a WFA that we denote by $\mathcal{B}^\eta$ and that associates to each sequence $\mathbf{x}$ the weight $\mathcal{C}[\mathbf{x}]^\eta$. Normalizing $\mathcal{B}^\eta$ results in a WFA $\mathcal{A}$ assigning weight $\mathcal{A}[\mathbf{x}] = \frac{\mathcal{B}[\mathbf{x}]^\eta}{\sum_x \mathcal{B}[\mathbf{x}]^\eta} = \mathsf{q}_1[x]$ to any $\mathbf{x} \in \Sigma^T$. This normalization can be achieved in time that is linear in the size of the WFA $\mathcal{B}^\eta$ using the Weight-Pushing algorithm [Mohri, 1997, 2009]. For completeness, we describe this algorithm in Appendix B. Note that since $\mathcal{B}^\eta$ is acyclic, its size is in $O(|E_\mathcal{A}|)$, where $E_\mathcal{A}$ is the set of transitions of $\mathcal{A}$.[5] We will denote by $\mathcal{A}$ the resulting WFA.

[5]The Weight-Pushing algorithm has been used in many other contexts to make a directed weighted graph stochastic. This includes network normalization in speech recognition [Mohri and Riley, 2001], and online learning with large expert sets [Takimoto and Warmuth, 2003, Cortes et al., 2015], where the resulting stochastic graph enables efficient sampling. The problem setting, algorithms and objectives in the last two references are completely distinct from ours. In particular, (a) in those, each path of the graph represents a single expert, while in our case each path is a sequence of experts; (b) in those, weight-pushing is applied at every round, while in our case it is used once at the start of the algorithm; (c) the regret is with respect to a static expert, while in our case it is with respect to a WFA of expert sequences.

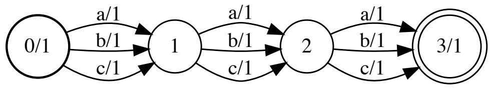

Figure 2: WFA $\mathcal{S}_T$, for $\Sigma = \{a, b, c\}$ and $T = 3$.

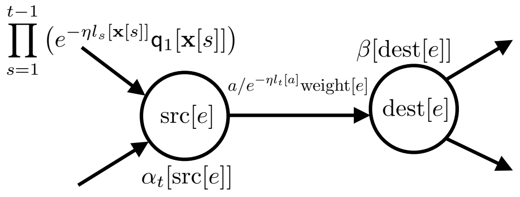
$\prod_{s=1}^{t-1} (e^{-\eta l_s[\mathbf{x}[s]]}\mathsf{q}_1[\mathbf{x}[s]])$
$\beta[\mathrm{dest}[e]]$
$a/e^{-\eta l_t[a]}\mathrm{weight}[e]$
$\mathrm{src}[e]$
$\mathrm{dest}[e]$
$\alpha_t[\mathrm{src}[e]]$

Figure 3: Illustration of algorithm AWM.

For any state u of $\mathcal{A}$, we will denote by $\boldsymbol{\beta}[u]$ the sum of the weights of all paths from u to a final state. The vector $\boldsymbol{\beta}$ can be computed in time that is linear in the number of states and transitions of $\mathcal{A}$ using a simple *single-source shortest-distance* algorithm in the semiring $(\mathbb{R}_+ \cup \{+\infty\}, +, \times, 0, 1)$ [Mohri, 2009], or the forward-backward algorithm. We call this subroutine BwdDist in the pseudocode.

We will denote by Q_t the set of states in $\mathcal{A}$ that can be reached by sequences of length t and by $E_\mathcal{A}^{t\to t+1}$ the set of transitions from a state in Q_t to a state in Q_{t+1}. For each transition e, let $\mathrm{src}[e]$ denote its source state, $\mathrm{dest}[e]$ its destination state, $\mathrm{lab}[e] \in \Sigma$ its label, and $\mathrm{weight}[e] \geq 0$ its weight. Since $\mathcal{A}$ is normalized, the expert probabilities $\mathsf{p}_1[a]$ for $a \in \Sigma$ can be read off the transitions leaving the initial state: $\mathsf{p}_1[a]$ is the weight of the transition in $E_\mathcal{A}^{0\to1}$ labeled with a.

Let $\boldsymbol{\alpha}_t[u]$ denote the *forward weights*, that is the sum of the weights of all paths from the initial state to state u just before the tth round. At round t, the weight of each transition e in $E_\mathcal{A}^{t\to t+1}$ is multiplied by $e^{-\eta l_t[\mathrm{lab}[e]]}$. This results in new forward weights $\boldsymbol{\alpha}_{t+1}[u]$ at the end of the t-th iteration. $\boldsymbol{\alpha}_{t+1}$ can be straightforwardly derived from $\boldsymbol{\alpha}_t$ since for $u \in Q_{t+1}$, $\boldsymbol{\alpha}_{t+1}[u]$ is given by $\boldsymbol{\alpha}_{t+1}[u] = \sum_{e\colon \mathrm{dest}[e]=u} \boldsymbol{\alpha}_t[\mathrm{src}[e]]\mathrm{weight}[e]$.

Observe that for any $t \in [T]$ and $\mathbf{x}$, $\mathsf{q}_t[\mathbf{x}]$ can be written as follows by unwrapping its recursive update definition:

$$\mathsf{q}_t[\mathbf{x}] = \frac{e^{-\eta \sum_{s=1}^{t-1} l_s[\mathbf{x}[s]]}\mathsf{q}_1[\mathbf{x}]}{\sum_{\mathbf{x}\in\Sigma^T} e^{-\eta \sum_{s=1}^{t-1} l_s[\mathbf{x}[s]]}\mathsf{q}_1[\mathbf{x}]}.$$

In view of that, for any $a \in \Sigma$, $\mathsf{p}_{t+1}[a]$ can be written as follows:

$$\mathsf{p}_{t+1}[a] = \frac{\sum_{\mathbf{x}\in\Sigma^T} e^{-\eta\sum_{s=1}^t l_s[\mathbf{x}[s]]}\mathsf{q}_1[\mathbf{x}]1_{\mathbf{x}[t]=a}}{\sum_{a\in\Sigma}\sum_{\mathbf{x}\in\Sigma^T} e^{-\eta\sum_{s=1}^t l_s[\mathbf{x}[s]]}\mathsf{q}_1[\mathbf{x}]1_{\mathbf{x}[t]=a}}.$$

Since the WFA $\mathcal{A}$ is deterministic, for any $\mathbf{x}$ accepted by $\mathcal{A}$ there is a unique accepting path π in $\mathcal{A}$ labeled with $\mathbf{x}$. The numerator of the expression of $\mathsf{p}_{t+1}[a]$ is then the sum of the weights of all paths in $\mathcal{A}$ with the tth symbol a at the end of tth iteration. This can be expressed as the sum over all transitions e in $E_{\mathcal{A}}^{t\to t+1}$ with label a of the total *flow* through e, that is the sum of the weights of all accepting path going through e: $\boldsymbol{\alpha}_t[\text{src}[e]]\,\text{weight}[e]\,\boldsymbol{\beta}[\text{dest}[e]]$ (see Figure 3). This is precisely the formula determining p_{t+1} in the pseudocode, where Z is the normalization factor.

The AWM algorithm is closely related to the Expert Hidden Markov Model of Koolen and de Rooij [2013] given for the log loss. It can be viewed as a generalization of that algorithm to arbitrary loss functions. A key difference between our setup and the perspective adopted by Koolen and de Rooij [2013] is that they assume a Bayesian setting where a prior distribution over expert sequences is given and must be used. We assume the existence of a competitor automaton $\mathcal{C}$, but do not necessarily need to sample from it for making predictions. This will be crucial in the next section, where we use a different WFA than $\mathcal{C}$ to improve computational efficiency while preserving regret performance. Also, the prior distribution in [Koolen and de Rooij, 2013] would be over $\mathcal{C}_T$ (for a large T) and not $\mathcal{C}$.

The computational complexity of AWM at each round t is $O(|E_{\mathcal{A}}^{t\to t+1}|)$, that is the time to update the weights of the transitions in $E_{\mathcal{A}}^{t\to t+1}$ and to incrementally compute $\boldsymbol{\alpha}$ for states reached by paths of length $t+1$. The total computational cost of the algorithm is thus $O\big(\sum_{t=1}^T |E_{\mathcal{A}}^{t\to t+1}|\big) = O(|E_{\mathcal{A}}|)$.[6] Note that $\mathcal{A}$ and $\mathcal{C}\cap\mathcal{S}_T$ admit the same topology, thus the total complexity of the algorithm depends on the number of transitions of the intersection WFA $\mathcal{C}\cap\mathcal{S}_T$, which is at most $|\mathcal{C}|NT$, where $|\mathcal{C}|$ is the size of the automaton $\mathcal{C}$ (i.e. the total number of states and transitions). This can be substantially more favorable than a naïve algorithm, whose worst-case complexity is exponential in T.

When the number of transitions of the intersection WFA $\mathcal{C}\cap\mathcal{S}_T$ is not too large compared to the number of experts N, the AWM algorithm is quite efficient. However, it is natural to ask whether one can design efficient algorithms even if the number of transitions $E_{\mathcal{A}}^{t\to t+1}$ to process per round is large (which may be the case even for a *minimized* WFA $\mathcal{C}\cap\mathcal{S}_T$ [Mohri, 2009]).

We will give two sets of solutions to derive a more efficient algorithm, which can be combined for further efficiency. In the next section, we discuss a solution that consists of using an approximate WFA with a smaller number of transitions. In Appendix E, we show that the notion of *failure transition*, originally used in the design of string-matching algorithms and recently employed for parameter estimation in backoff n-gram language models [Roark et al., 2013], can be used to derive a more compact representation of the WFA $\mathcal{C}\cap\mathcal{S}_T$, thereby resulting in a significantly more efficient online learning algorithm that still admits compelling regret guarantees.

[6] By the discussion above and Appendix A, the total complexity of the intersection and weight-pushing operations is also in $O(|E_{\mathcal{A}}|)$, so that they do not add any additional cost. Moreover, these two operations need only be carried out once and can be performed offline.

4 Approximation algorithms

In this section, we present approximation algorithms for the problem of online learning against a weighted sequence of experts represented by a WFA $\mathcal{C}$. Rather than using the intersection WFA $\mathcal{C}_T = \mathcal{C}\cap\mathcal{S}_T$, we will assume that AWM is run with an approximate WFA $\widehat{\mathcal{C}}_T$. The main motivation for doing so is that the algorithm can be substantially more efficient if $\widehat{\mathcal{C}}_T$ admits significantly fewer transitions than $\mathcal{C}_T$. Of course, this comes at the price of a somewhat weaker regret guarantee that we now analyze in detail.

4.1 Effect of WFA approximation

We first analyze the effect of automata approximation on the regret of AWM. As in the previous section, we denote by q the distribution defined by $\mathcal{C}_T$ over sequences of length T. We will similarly denote by $\widehat{\mathsf{q}}$ the distribution defined by $\widehat{\mathcal{C}}_T$ over the same set. The effect of the WFA approximation on the regret can be naturally expressed in terms of the ∞-Rényi divergence $D_\infty(\mathsf{q}\|\widehat{\mathsf{q}})$ between the distributions q and $\widehat{\mathsf{q}}$, which is defined by $D_\infty(\mathsf{q}\|\widehat{\mathsf{q}}) = \sup_{\mathbf{x}\in\Sigma^T}\log[\mathsf{q}(\mathbf{x})/\widehat{\mathsf{q}}(\mathbf{x})]$.

Theorem 2. *The weighted regret of the* AWM *algorithm with respect to the WFA $\mathcal{C}$ when run with $\widehat{\mathcal{C}}_T$ instead of $\mathcal{C}_T$ can be upper bounded as follows:*

$$\begin{aligned}\text{Reg}_T(\mathcal{A},\mathcal{C}) &\le \frac{\eta T}{8} + \frac{1}{\eta}\log\Big[K^\eta\sum_{\mathbf{x}}\widehat{\mathsf{q}}[\mathbf{x}]^\eta\Big] + D_\infty(\mathsf{q}\|\widehat{\mathsf{q}})\\ &\le \frac{\eta T}{8} + \frac{1}{\eta}\log K + D_\infty(\mathsf{q}\|\widehat{\mathsf{q}}).\end{aligned}$$

Its unweighted regret can be upper bounded as follows:

$$\text{Reg}_T^0(\mathcal{A},\mathcal{C}) \le \max_{\mathcal{C}(\mathbf{x})>0}\frac{\eta T}{8} + \frac{1}{\eta}\log\left[\frac{1}{\mathsf{q}[\mathbf{x}]}\right] + \frac{1}{\eta}D_\infty(\mathsf{q}\|\widehat{\mathsf{q}}).$$

The proof is given in Appendix D. Theorem 2 shows that the extra cost of using an approximate WFA $\widehat{\mathcal{C}}_T$ instead of $\mathcal{C}_T$ is $D_\infty(\mathsf{q}\|\widehat{\mathsf{q}})$ for the weighted regret and similarly $\frac{1}{\eta}D_\infty(\mathsf{q}\|\widehat{\mathsf{q}})$ for the unweighted regret. The bound is tight since the best sequence in hindsight in the regret definition may also be the one maximizing the log-ratio.

The theorem suggests a general algorithm for selecting an approximate WFA $\widehat{\mathcal{C}}$ out of a family $\mathcal{C}$ of WFAs with a relatively small number of transitions. This consists of choosing $\widehat{\mathcal{C}}$ to minimize the Rényi divergence as defined by the following program:

$$\min_{\widehat{\mathcal{C}} \in \mathcal{C}} D_\infty(\mathsf{q} \| \widehat{\mathsf{q}}), \tag{7}$$

where $\widehat{q}$ is the distribution induced by $\widehat{\mathcal{C}}$ over Σ^T (the one obtained by computing $\widehat{\mathcal{C}}_T = \widehat{\mathcal{C}} \cap \mathcal{S}_T$ and normalizing the weights). The theorem ensures that the solution benefits from the most favorable regret guarantee among the WFAs in $\mathcal{C}$. When the set of distributions associated to $\mathcal{C}$ is convex, then the set of distributions defined over Σ^T is also convex. This is then a convex optimization problem, since $\widehat{\mathsf{q}} \mapsto \log(\mathsf{q}/\widehat{\mathsf{q}})$ is a convex function and a supremum of convex functions is convex.

The choice of the family $\mathcal{C}$ is subject to a trade-off: approximation accuracy versus computational efficiency of using WFAs in $\mathcal{C}$. This raises a model selection question for which we discuss in detail a solution in Section 4.2: given a sequence of families $(\mathcal{C}_n)_{n \in \mathbf{N}}$ with growing complexity and computational cost, the problem consists of selecting the best n.

In the following, we will consider the case where the family $\mathcal{C}$ of weighted automata is that of *n-gram models*, for which we can upper bound the computational complexity.

4.2 Minimum Rényi divergence n-gram models

Let $\Sigma^{\leq n-1}$ denote the set of sequences of length at most $n-1$. An n-gram language model is a Markovian model of order $(n-1)$ defined over Σ^*, which can be compactly represented by a WFA with each state identified with a sequence $\mathbf{x} \in \Sigma^{\leq n-1}$, thereby encoding the sequence *just read* to reach that state. The WFA admits a transition from state $(\mathbf{x}[1] \cdots \mathbf{x}[n-1])$ to state $(\mathbf{x}[2] \cdots \mathbf{x}[n-1]a)$ with weight $\mathsf{w}\big[a \,|\, \mathbf{x}[1] \cdots \mathbf{x}[n-1]\big]$, for any $a \in \Sigma$, and, for any $k \leq n-1$, a transition from state $(\mathbf{x}[1] \cdots \mathbf{x}[k-1])$ to state $(\mathbf{x}[1] \cdots \mathbf{x}[k-1]a)$ with weight $\mathsf{w}\big[a \,|\, \mathbf{x}[1] \cdots \mathbf{x}[k-1]\big]$, for any $a \in \Sigma$. It admits a unique initial state which is the one labeled with the empty string ϵ (sequence of length zero) and all its states are final. The WFA is stochastic, that is outgoing transition weights sum to one at every state: thus, $\sum_{a \in \Sigma} \mathsf{w}[a|\mathbf{x}] = 1$ for all $\mathbf{x} \in \Sigma^{\leq n-1}$. Notice that this WFA is also deterministic since it admits a unique initial state and no two transitions with the same label leaving any state. Figure 4 illustrates this definition in the case of a simple bigram model.

Note that the transition weights $\mathsf{w}[a|\mathbf{x}]$, with $a \in \Sigma$ and $\mathbf{x} \in \cup_{k \leq n-1} \Sigma^k$ fully specify an n-gram model. Since for a fixed $\mathbf{x} \in \cup_{k \leq n-1} \Sigma^k$, $\mathsf{w}[\cdot|\mathbf{x}]$ is an element of the simplex, an n-gram model can be viewed as an element of the product of $\sum_{k=0}^{n-1} \Sigma^k$ simplices, a convex set. We will denote by $\mathcal{W}_n$ the family of all n-gram models.

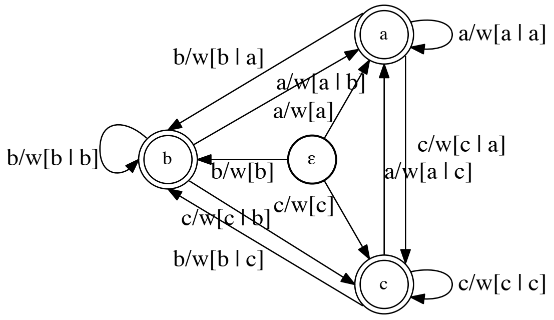

Figure 4: A bigram language model over the alphabet $\Sigma = \{a, b, c\}$.

One key advantage of n-gram models in this context is that the per-iteration complexity can be bounded in terms of the number of symbols. Since an n-gram model has at most $|\Sigma|^{n-1}$ states, its per-iteration computational cost is in $O\big(|\Sigma|^n\big)$ as each state can take one of $|\Sigma|$ possible transitions. For n small, this can be very advantageous compared to the original $\mathcal{C}_T$, since in general the maximum out-degree of states reached by sequences of length t in the latter can be very large. For instance, the automaton $\mathcal{C}_{\text{weighted-shift}}$ in Figure 1 (ii) can itself be viewed as a bigram model and admits efficient computation.

For n-gram models, our approximation algorithm (Problem 7) can be written as follows:

$$\min_{\mathsf{w} \in \mathcal{W}_n} D_\infty(\mathsf{q} \| \mathsf{q}_\mathsf{w}) = \min_{\mathsf{w} \in \mathcal{W}_n} \sup_{\mathbf{x} \in \Sigma^T} \log \left[\frac{\mathsf{q}[\mathbf{x}]}{\mathsf{q}_\mathsf{w}[\mathbf{x}]} \right], \tag{8}$$

where q_w is the distribution induced by the n-gram model w on sequences in Σ^T. By definition of the n-gram model, for any $\mathbf{x} \in \Sigma^T$, $\mathsf{q}_\mathsf{w}[\mathbf{x}]$ is given by the following:

$$\mathsf{q}_\mathsf{w}[\mathbf{x}] = \prod_{t=1}^{T} \mathsf{w}\big[\mathbf{x}[t] \big| \mathbf{x}_{\max(t-n+1,1)}^{t-1}\big],$$

since the weights of sequences of any fixed length sum to one in an n-gram model. Problem 8 is a convex optimization problem over $\mathcal{W}_n$. The problem can be solved using as an an extension of the Exponentiated Gradient (EG) algorithm of Kivinen and Warmuth [1997], which we will refer to as Prod-EG. The pseudocode of Prod-EG, a general convergence guarantee, and its convergence guarantee in the specific case of n-gram models are given in detail in Appendix F as Algorithm 6, Theorem 5, and Corollary 1 respectively.

Model selection. In practice, we seek an n-gram model that balances the tradeoff between approximation error and computational cost. Assume that we are given a maximum per-iteration computational budget B. We therefore wish to determine an n-gram approximation model affordable within our budget and with the most favorable regret guarantee. Let $F(\mathsf{q}, \mathsf{q}_\mathsf{w})$ denote the objective

Algorithm 2: n-GRAMMODELSELECT.

```
Algorithm: $n$-GRAMMODELSELECT($\mathsf{q}$, $\tau$, $B$)
$n \leftarrow 1$; $\mathsf{q_w} \leftarrow \mathsf{q}_{\mathsf{u}_n}$; $s \leftarrow 0$
while $s \leq \tau$ do
    $\mathsf{q_w} \leftarrow$ PROD-EG-UPDATE($\mathsf{q_w}, \mathcal{W}_n$)
    $s \leftarrow s + 1$
    if $F(\mathsf{q}, \mathsf{q_w}) - \Delta(s, n) > \sqrt{T}$ and $|\Sigma|^n \leq B$ then
        $n \leftarrow 2n$; $s \leftarrow 0$; $\mathsf{q_w} \leftarrow \mathsf{q}_{\mathsf{u}_n}$
$n_{\max} \leftarrow n$.
$\mathsf{q_w} \leftarrow$ BINARYSEARCH($[1, n_{\max}]$, $F(\mathsf{q}, \mathsf{q_w}) -$
    $\Delta(\tau, n) \leq \sqrt{T}$)
return $\mathsf{q_w}$
```

function of Problem (8): $F(\mathsf{q}, \mathsf{q_w}) = D_\infty(\mathsf{q} \| \mathsf{q_w})$. By the convergence guarantee of Corollary 1, if $\mathsf{q_w}$ is the n-gram model returned by PROD-EG after τ iterations, we can write $F(\mathsf{q}, \mathsf{q_w}) - F(\mathsf{q}, \mathsf{q_{w^*}}) \leq \Delta(\tau, n)$, where w^* is the n-gram model minimizing Problem (8) over $\mathcal{W}_n$ and $\Delta(\tau, n)$ the upper bound given by Corollary 1. Thus, if $F(\mathsf{q}, \mathsf{q_w}) - \Delta(\tau, n) > \sqrt{T}$ for some n, then, even the optimal n-gram model for this n will cause an increase in the regret.

Let n^* be the smallest n such that $F(\mathsf{q}, \mathsf{q_w}) - \Delta(\tau, n) \leq \sqrt{T}$ (or the smallest value that exceeds our budget). We can find this value in $\log(n^*)$ time using a two-stage process. In the first stage, we double n after every violation until we find an upper bound on n^*, which we denote by $n_{\max}$. In the second stage, we perform a binary search within $[1, n_{\max}]$ to determine n^*. Each stage takes $\log(n^*)$ iterations, and each iteration is the cost of running PROD-EG for that specific value of n. Thus, the overall complexity of the algorithm is $O(\log(n^*)\,\text{Cost}(\text{PROD-EG}))$, where Cost(PROD-EG) is the cost of a call to PROD-EG. The full pseudocode of this algorithm, n-GRAMMODELSELECT, is presented as Algorithm 2, where u_n denotes the uniform n-gram model and PROD-EG-UPDATE($\mathsf{q_w}, \mathcal{W}_n$) denotes one update made by PROD-EG when optimizing over $\mathcal{W}_n$.

In the simple case of a unigram automaton model over two symbols and when the distribution q defined by the intersection WFA $\mathcal{C}_T$ is uniform, we can give an explicit form of the solution of Problem 8. The solution is obtained from the paths with the smallest number of occurrences of each symbol, which can be straightforwardly found via a shortest-path algorithm in linear time.

Theorem 3. *Assume that $\mathcal{C}_T$ admits uniform weights over all paths and $\Sigma = \{a_1, a_2\}$. For $j \in \{1, 2\}$, let $n(a_j)$ be the smallest number of occurrences of a_j in a path of $\mathcal{C}_T$. For any $j \in \{1, 2\}$, define*

$$\mathsf{q}[a_j] = \frac{\max\left\{1, \frac{n(a_j)}{T - n(a_j)}\right\}}{1 + \max\left\{1, \frac{n(a_j)}{T - n(a_j)}\right\}}.$$

Then, the unigram model $\mathsf{w} \in \mathcal{W}_1$ solution of ∞-Rényi divergence optimization problem is defined by $\mathsf{w}[a_{j^*}] = \mathsf{q}[a_{j*}]$, $\mathsf{w}[a_{j'}] = 1 - \mathsf{w}[a_{j^*}]$, with $j^* = \operatorname{argmax}_{j \in \{1,2\}}\ n(a_j) \log \mathsf{q}[a_j] + [T - n(a_j)] \log\big[1 - \mathsf{q}[a_j]\big]$.

The proof of this result is provided in Appendix G.

Theorem 3 shows that the solutions of the ∞-Rényi divergence optimization are based on the n-gram counts of sequences in $\mathcal{C}_T$ with "high entropy". This can be very different from the maximum likelihood solutions, which are based on the average n-gram counts. For instance, suppose we are under the assumptions of Theorem 3, and specifically, assume that there are T sequences in $\mathcal{C}_T$. Assume that one of the sequences has $\left(\frac{1}{2} + \gamma\right) T$ occurrences of a_1 for some small $\gamma > 0$ and that the other $T - 1$ sequences have $T - 1$ occurrences of a_1. Then, $n(a_1) = \left(\frac{1}{2} + \gamma\right) T$, and the solution of the ∞-Rényi divergence optimization problem is given by $\mathsf{q}_\infty(a_1) = \frac{1+2\gamma}{2}$ and $\mathsf{q}_\infty(a_2) = \frac{1-2\gamma}{2}$. On the other hand, the maximum-likelihood solution would be $\mathsf{q}_1(a_1) = 1 + \frac{\gamma}{T} - \frac{3}{2T} + \frac{1}{T^2} \approx 1$ and $\mathsf{q}_1(a_2) = \frac{3}{2T} - \frac{\gamma}{T} - \frac{1}{T^2} \approx 0$ for large T.

4.3 Maximum-Likelihood n-gram models

A standard method for learning n-gram models is via Maximum-Likelihood, which is equivalent to minimizing the relatively entropy between the target distribution q and the language model, that is via

$$\min_{\mathsf{w} \in \mathcal{W}_n} D(\mathsf{q} \| \mathsf{q_w}), \tag{9}$$

where, $D(\mathsf{q} \| \mathsf{q_w})$ denotes the relative entropy, $D(\mathsf{q} \| \mathsf{q_w}) = \sum_{\mathbf{x}} \mathsf{q}[\mathbf{x}] \log \left[\frac{\mathsf{q}[\mathbf{x}]}{\mathsf{q_w}[\mathbf{x}]}\right]$. Maximum likelihood n-gram solutions are simple. For standard text data, the weight of each transition is the frequency of appearance of the corresponding n-gram in the text. For a probabilistic $\mathcal{C}_T$, the weight can be similarly obtained from the expected count of the n-gram in the paths of $\mathcal{C}_T$, where the expectation is taken over the probability distribution defined by $\mathcal{C}_T$ and can be computed efficiently [Allauzen et al., 2003]. In general, the solution of this optimization problem does not benefit from the guarantee of Theorem 2 since the ∞-Rényi divergence is an upper bound on the relative entropy. However, in some cases, maximum likelihood solutions do benefit from favorable regret guarantees. In particular, as shown by the following theorem, remarkably, the maximum-likelihood bigram approximation to the k-shifting automaton coincides with the FIXED-SHARE algorithm of Herbster and Warmuth [1998] and benefits from a constant approximation error. Thus, we can view and motivate the design of the FIXED-SHARE algorithm as that of a bigram approximation of the desired competitor automaton, which represents the family of k-shifting sequences.

Theorem 4. *Let $\mathcal{C}_T$ be the k-shifting automaton for some k. Then, the bigram model w_2 obtained by minimizing relative* *entropy is defined for all $a_1, a_2 \in \Sigma$ by*

$$\mathsf{p}_{\mathsf{w}_2}[a_1 a_2] = \frac{\left[1 - \frac{k}{(T-1)}\right] 1_{a_1 = a_2} + \left[\frac{k}{(T-1)(N-1)}\right] 1_{a_1 \neq a_2}}{N}.$$

Moreover, its approximation error can be bounded by a constant (independent of T):

$$D_\infty(\mathsf{q} \| \mathsf{q}_{\mathsf{w}_2}) \leq -\log\left[1 - 2e^{-\frac{1}{12k}}\right].$$

The proof of the theorem as well as other details about Maximum-Likelihood are given in Appendix H. The proof technique is illustrative because it reveals that the maximum likelihood n-gram model has low approximation error whenever (1) the model's distribution is proportional to the distribution of $\mathcal{C}_T$ on $\mathcal{C}_T$'s support and (2) most of the model's mass lies on the support of $\mathcal{C}_T$. When the automaton $\mathcal{C}_T$ has uniform weights, then condition (1) is satisfied when the n-gram model is uniform on $\mathcal{C}_T$. This is true whenever all sequences in $\mathcal{C}_T$ have the same set of n-gram counts, and every permutation of symbols over these counts is a sequence that lies in $\mathcal{C}_T$, which is the case for the k-shifting automaton. Condition (2) is satisfied when n is large enough, which necessarily exists since the distribution is exact for $n = T$. On the other hand, note that a unigram approximation would have satisfied condition (1) but not condition (2) for the k-shifting automaton.

To the best of our knowledge, this is the first framework that motivates the design of FIXED-SHARE with a focus on minimizing tracking regret. Other works that have recovered FIXED-SHARE (e.g. [Vovk, 1999, Cesa-Bianchi and Lugosi, 2006, Cesa-Bianchi et al., 2012, Koolen and de Rooij, 2013, György and Szepesvári, 2016]) have generally viewed the algorithm itself as the main focus and have not systematically derived it from first principles.[7]

Our derivation of FIXED-SHARE also allows us to naturally generalize the setting of standard k-shifting experts to k-shifting experts with non-uniform weights. Specifically, consider the case where $\mathcal{C}_T$ is an automaton accepting up to k-shifts but where the shifts now occur with probability $\mathsf{q}[a_2 | a_1, a_1 \neq a_2] \neq \frac{1}{N-1} 1_{\{a_2 \neq a_1\}}$. Since the bigram approximation will remain exact on $\mathcal{C}_T$, we recover the exact same guarantee as in Theorem 4.

Maximum likelihood n-gram models can further benefit from our use of failure transitions and the φ-conversion algorithm presented in Appendix E. This can reduce the size of the automaton and often dramatically improve its computational efficiency without affecting its accuracy.

[7]Section 5.2 of Cesa-Bianchi and Lugosi [2006] provides some intuition on why the choice of prior weights in the weighted majority interpretation of *Fixed-Share* is reasonable, but this is conducted a posteriori.

5 Extension to sleeping experts

In many real-world applications, it may be natural for some experts to abstain from making predictions on some of the rounds. For instance, in a bag-of-words model for document classification, only a subset of features may be active for each document. This extension of standard prediction with expert advice is also known as the *sleeping experts framework* [Freund et al., 1997]. The experts are said to be asleep when they are inactive and awake when they are active and available to be selected. This framework is distinct from the permutation-based setting adopted in [Kleinberg et al., 2010, Kanade et al., 2009, Kanade and Steinke, 2014].

Formally, at each round t, the adversary chooses an awake set $A_t \subseteq \Sigma$ from which the learner is allowed to query an expert. The algorithm then chooses an expert i_t from A_t, receives a loss vector $l_t \in [0,1]^{|\Sigma|}$ supported on A_t and incurs loss $l_t[i_t]$. Since some experts may be asleep, it is not reasonable to compare the loss against that of the best static expert. In [Freund et al., 1997], the comparison is made against the best fixed mixture of experts normalized at each round over the awake set: $\min_{\mathsf{u} \in \Delta_N} \sum_{t=1}^T \frac{\sum_{a \in A_t} \mathsf{u}[a] l_t[a]}{\sum_{a' \in A_t} \mathsf{u}[a']}$, where Δ_N is the $(N-1)$-dimensional simplex.

We extend the notion of sleeping experts to the path setting, so that instead of comparing against fixed mixtures over experts, we compare against fixed mixtures over the family of expert sequences. With some abuse of notation, let A_t also represent the automaton accepting all paths of length T whose t-th transition has label in A_t. Then, we want to design an algorithm that performs well with respect to the following quantity: $\min_{\mathsf{u} \in \Delta_K} \sum_{t=1}^T \frac{\sum_{\mathbf{x} \in \mathcal{C}_T \cap A_t} \mathsf{u}[\mathbf{x}] l_t[\mathbf{x}[t]]}{\sum_{\mathbf{x} \in \mathcal{C}_T \cap A_t} \mathsf{u}[\mathbf{x}]}$, where K is the number of accepting paths of $\mathcal{C}_T$. We describe our new algorithm, AWAKEAWM, along with its accompanying theoretical guarantee, in Appendix I.

6 Conclusion

We studied a general framework of online learning against a competitor class represented by a WFA and presented a number of algorithmic solutions for this problem achieving sublinear regret guarantees using automata approximation and failure transitions. We also extended our algorithms and results to the sleeping experts framework (Section 5). Our results can be straightforwardly extended to the adversarial bandit scenario using standard surrogate losses based on importance weighting techniques and to the case of more complex formal language families such as (probabilistic) context-free languages over expert sequences.

Acknowledgements

This work was partly funded by NSF CCF-1535987 and NSF IIS-1618662.

References

D. Adamskiy, W. M. Koolen, A. Chernov, and V. Vovk. A closer look at adaptive regret. In *ALT*, pages 290–304, 2012.

C. Allauzen, M. Mohri, and B. Roark. Generalized algorithms for constructing statistical language models. In *Proceedings of ACL*, pages 40–47, 2003.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32(1):48–77, 2002.

O. Besbes, Y. Gur, and A. Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *NIPS*, pages 199–207, 2014.

O. Besbes, Y. Gur, and A. Zeevi. Non-stationary stochastic optimization. *Operations Research*, 63(5):1227–1244, 2015.

S. Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

N. Cesa-Bianchi and G. Lugosi. *Prediction, Learning, and Games*. Cambridge University Press, 2006.

N. Cesa-Bianchi, Y. Mansour, and G. Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2-3):321–352, 2007.

N. Cesa-Bianchi, P. Gaillard, G. Lugosi, and G. Stoltz. Mirror descent meets fixed share (and feels no regret). In *NIPS*, pages 980–988, 2012.

C. Cortes, V. Kuznetsov, M. Mohri, and M. K. Warmuth. On-line learning algorithms for path experts with non-additive losses. In *COLT*, 2015.

A. Daniely, A. Gonen, and S. Shalev-Shwartz. Strongly adaptive online learning. In *Proceedings of ICML*, pages 1405–1411, 2015.

Y. Freund, R. E. Schapire, Y. Singer, and M. K. Warmuth. Using and combining predictors that specialize. In *STOC*, pages 334–343. ACM, 1997.

A. György and C. Szepesvári. Shifting regret, mirror descent, and matrices. In *ICML*, 2016.

A. Gyorgy, T. Linder, and G. Lugosi. Efficient tracking of large classes of experts. *IEEE Transactions on Information Theory*, 58(11):6709–6725, 2012.

E. C. Hall and R. M. Willett. Online optimization in dynamic environments. *arXiv:1307.5944*, 2013.

E. Hazan and C. Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of ICML*, pages 393–400. ACM, 2009.

M. Herbster and M. K. Warmuth. Tracking the best expert. *Machine Learning*, 32(2):151–178, 1998.

A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan. Online optimization: Competing with dynamic comparators. In *AISTATS*, 2015.

V. Kanade and T. Steinke. Learning hurdles for sleeping experts. *ACM Transactions on Computation Theory (TOCT)*, 6(3):11, 2014.

V. Kanade, H. McMahan, and B. Bryan. Sleeping experts and bandits with stochastic action availability and adversarial rewards. In *AISTATS*, pages 272–279, 2009.

J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

R. Kleinberg, A. Niculescu-Mizil, and Y. Sharma. Regret bounds for sleeping experts and bandits. *Machine learning*, 80(2-3):245–272, 2010.

W. M. Koolen and S. de Rooij. Universal codes from switching strategies. *IEEE Transactions on Information Theory*, 59(11):7168–7185, 2013.

N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

M. Mohri. Finite-State Transducers in Language and Speech Processing. *Computational Linguistics*, 23:2, 1997.

M. Mohri. Weighted automata algorithms. In *Handbook of weighted automata*, pages 213–254. Springer, 2009.

M. Mohri and M. Riley. A Weight Pushing Algorithm for Large Vocabulary Speech Recognition. In *Proceedings of Eurospeech*, 2001.

A. Mokhtari, S. Shahrampour, A. Jadbabaie, and A. Ribeiro. Online optimization in dynamic environments: Improved regret rates for strongly convex problems. In *Proceedings of CDC*, pages 7195–7201. IEEE, 2016.

C. Monteleoni and T. S. Jaakkola. Online learning of non-stationary sequences. In *NIPS*, page None, 2003.

B. Roark, C. Allauzen, and M. Riley. Smoothed marginal distribution constraints for language modeling. In *ACL*, pages 43–52, 2013.

S. Shahrampour and A. Jadbabaie. Distributed online optimization in dynamic environments using mirror descent. *arXiv:1609.02845*, 2016.

E. Takimoto and M. K. Warmuth. Path kernels and multiplicative updates. *Journal of Machine Learning Research*, 4(Oct):773–818, 2003.

T. Van Erven and P. Harremos. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory*, 60(7):3797–3820, 2014.

V. Vovk. Derandomizing stochastic prediction strategies. *Machine Learning*, 35(3):247–282, 1999.

C.-Y. Wei, Y.-T. Hong, and C.-J. Lu. Tracking the best expert in non-stationary stochastic environments. In *NIPS*, 2016.

A Intersection of WFAs

The intersection of two WFAs $\mathcal{A}_1$ and $\mathcal{A}_2$ is a WFA denoted by $\mathcal{A}_1 \cap \mathcal{A}_2$ that accepts the set of sequences accepted by both $\mathcal{A}_1$ and $\mathcal{A}_2$ and is defined for all $\mathbf{x}$ by

$$(\mathcal{A}_1 \cap \mathcal{A}_2)(\mathbf{x}) = \mathcal{A}_1(\mathbf{x})\, \mathcal{A}_2(\mathbf{x}).$$

There exists a standard efficient algorithm for computing the intersection WFA [Mohri, 2009]. States of $\mathcal{A}_1 \cap \mathcal{A}_2$ are identified with pairs of states Q_1 of $\mathcal{A}_1$ and Q_2 of $\mathcal{A}_2$: $Q \subseteq Q_1 \times Q_2$, as are the set of initial and final states. Transitions are obtained by matching pairs of transitions from each weighted automaton and multiplying their weights following the rule

$$\left(q_1 \xrightarrow{a/w_1} q_1',\ q_2 \xrightarrow{a/w_2} q_2'\right) \quad \Rightarrow \quad (q_1, q_2) \xrightarrow{a/(w_1 w_2)} (q_1', q_2').$$

The worst-case space and time complexity of the intersection of two deterministic weighted finite automata (WFA) is linear in the size of the automaton the algorithm returns. In the worst case, this can be as large as the product of the sizes of the WFA intersected (i.e. $O(|\mathcal{A}_1||\mathcal{A}_2|)$, where $|\mathcal{A}_1|$ is the sum of the number of states and transitions of $\mathcal{A}_1$ and similarly with $|\mathcal{A}_2|$. This corresponds to the case where every transition of $\mathcal{A}_1$ can be paired up with every transition of $\mathcal{A}_2$. In practice far fewer transitions can be matched.

Notice that when both $\mathcal{A}_1$ and $\mathcal{A}_2$ are deterministic, then $\mathcal{A}_1 \cap \mathcal{A}_2$ is also deterministic since there is a unique initial state (pair of initial states of each WFA) and since there is at most one transition leaving $q_1 \in Q_1$ or $q_2 \in Q_2$ labeled with a given symbol $a \in \Sigma$.

In the case of $\mathcal{C} \cap S_T$, the WFA returned is $\mathcal{B}$, which has the same size as $\mathcal{A}$. $\mathcal{A}$ has more transitions than states since each state admits at least on outgoing transition, so its size is dominated by its number of transitions. Therefore, the complexity of intersection here is in $O(|E_\mathcal{A}|)$, where $|E_\mathcal{A}|$ is at most $|\mathcal{C}|NT$.

B WEIGHT-PUSHING algorithm

Here, we briefly describe the WEIGHT-PUSHING algorithm for a WFA $\mathcal{A}$ in the context of this paper [Mohri, 1997, 2009]. We denote by $Q_\mathcal{A}$ the set of states of $\mathcal{A}$, by $E_\mathcal{A}$ the set of transitions of $\mathcal{A}$, by $I_\mathcal{A}$ its initial state, by $F_\mathcal{A}$ the set of its final states, and by $\rho_\mathcal{A}(q)$ the final weight at a final state q – for the WFAs considered in this paper the final weights are all equal to one.

For any state $q \in Q_\mathcal{A}$, let $d[q]$ denote the sum of the weights of all paths from q to final states:

$$d[q] = \sum_{\pi \in P(q, F_\mathcal{A})} \text{weight}[\pi]\, \rho(\text{dest}[\pi]),$$

where $P(q, F_\mathcal{A})$ denotes the set of paths from q to a state in $F_\mathcal{A}$. For an acyclic WFA $\mathcal{A}$, the weights $d[q]$ can be computed in linear time in the size of $\mathcal{A}$, that is in $O(|Q_\mathcal{A}| + |E_\mathcal{A}|)$, or $O(|E_\mathcal{A}|)$ when every state of $\mathcal{A}$ admits at least one outgoing or incoming transition. This can be done using a general shortest-distance algorithm [Mohri, 1997, 2009].

The weight-pushing algorithm then consists of the following steps. For any transition $e \in E_\mathcal{A}$ such that $d[\text{src}[e]] \neq 0$, we update its weight as follows:

$$\text{weight}[e] \leftarrow d[\text{src}[e]]^{-1}\, \text{weight}[e]\, d[\text{dest}[e]].$$

For any state $q \in F_\mathcal{A}$ with $d[q] \neq 0$, we update its final weight as follows:

$$\rho_\mathcal{A}[q] \leftarrow d[q]^{-1}\, \rho_\mathcal{A}[q].$$

The resulting WFA is guaranteed to be stochastic (at any state q, the sum of the weights of all outgoing transitions, and the final weight if q is final, is equal to one) [Mohri, 2009]. Furthermore, if $d[I_\mathcal{A}] = 1$, that is if the sum of the weights of all paths is one, then path weights are preserved by this weight-pushing operation. Otherwise, the weights of all paths starting at the initial state is divided by $d[I_\mathcal{A}]$.

C Proof of Theorem 1

Theorem 1. *Let* q *denote the probability distribution defined by* $\mathcal{C}_T = \mathcal{C} \cap \mathcal{S}_T$ *and let* K *denote the number of accepting paths of* $\mathcal{C}_T$*. Then, the following upper bound holds for the weighted regret of* AWM*:*

$$\mathrm{Reg}_T(\mathrm{AWM}, \mathcal{C}) \leq \frac{\eta T}{8} + \frac{1}{\eta} \log \left[K^\eta \sum_{\mathbf{x} \in \Sigma^T} \mathsf{q}[\mathbf{x}]^\eta \right] \leq \frac{\eta T}{8} + \frac{1}{\eta} \log K.$$

Furthermore, when $K \geq 2$*, for any* $T > 0$*, there exists* $\eta^* > 0$*, decreasing function of* T*, such that:*

$$\mathrm{Reg}_T(\mathrm{AWM}, \mathcal{C}) \leq \sqrt{\frac{T H_{\eta^*}(\mathsf{q})}{2}} - H_{\eta^*}(\mathsf{q}) + \log K,$$

where $H_\eta(\mathsf{q}) = \frac{1}{1-\eta} \log \left(\sum_{\mathbf{x} \in \Sigma^T} \mathsf{q}[\mathbf{x}] \right)$ *is the* η*-Rényi entropy of* q*. The unweighted regret of* AWM *can be upper-bounded as follows:*

$$\mathrm{Reg}^0_T(\mathrm{AWM}, \mathcal{C}) \leq \frac{\eta T}{8} + \frac{1}{\eta} \log K.$$

Proof. We will use a standard potential-based argument. For any $t \geq 1$ and sequence $\mathbf{x} \in \Sigma^T$, let $w_t[\mathbf{x}]$ denote the sequence weight defining q_t via normalization, $\mathsf{q}_t[\mathbf{x}] = \frac{w_t[\mathbf{x}]}{\sum_{\mathbf{x}} w_t[\mathbf{x}]}$, that is $w_1[\mathbf{x}] = \mathsf{q}[\mathbf{x}]^\eta$ and, for $t \geq 2$, $w_t[\mathbf{x}] = w_1[\mathbf{x}] e^{-\eta \sum_{s=1}^{t-1} l_s[\mathbf{x}[s]]}$. Let Φ_t be the potential defined by $\Phi_t = \log \left(\sum_{\mathbf{x}} w_t[\mathbf{x}] \right)$ for $t \geq 1$. Then, by Hoeffding's inequality, we can write

$$\begin{aligned}
\Phi_{t+1} - \Phi_t &= \log \left[\frac{\sum_{\mathbf{x}} w_t[\mathbf{x}] \, e^{-\eta l_t[\mathbf{x}[t]]}}{\sum_{\mathbf{x}} w_t[\mathbf{x}]} \right] \\
&= \log \left[\mathop{\mathbb{E}}_{\mathbf{x} \sim \mathsf{q}_t} \left[e^{-\eta l_t[\mathbf{x}[t]]} \right] \right] \\
&\leq -\eta \mathop{\mathbb{E}}_{\mathbf{x} \sim \mathsf{q}_t} \left[l_t[\mathbf{x}[t]] \right] + \frac{\eta^2}{8} = -\eta \mathop{\mathbb{E}}_{a \sim \mathsf{p}_t} \left[l_t[a] \right] + \frac{\eta^2}{8}.
\end{aligned}$$

Summing up these inequalities over $t \in [1, T]$ results in the following upper bound:

$$\Phi_{T+1} - \Phi_1 \leq -\eta \sum_{t=1}^T \mathop{\mathbb{E}}_{a \sim \mathsf{p}_t} [l_t[a]] + \frac{\eta^2 T}{8}.$$

We can straightforwardly derive a lower bound for the same quantity for any sequence $\mathbf{x}_0 \in \Sigma^T$:

$$\begin{aligned}
\Phi_{T+1} - \Phi_1 &= \log \Big[\sum_{\mathbf{x}} w_{T+1}[\mathbf{x}] \Big] - \log \Big[\sum_{\mathbf{x}} w_1[\mathbf{x}] \Big] \\
&\geq \log[w_{T+1}[\mathbf{x}_0]] - \log \Big[\sum_{\mathbf{x}} w_1[\mathbf{x}] \Big] \\
&= -\eta \sum_{t=1}^T l_t[\mathbf{x}_0[t]] + \log[\mathsf{q}[\mathbf{x}_0]^\eta] - \log \Big[\sum_{\mathbf{x}} \mathsf{q}[\mathbf{x}]^\eta \Big].
\end{aligned}$$

Comparing the upper and lower bounds gives

$$-\eta \sum_{t=1}^T l_t[\mathbf{x}_0[t]] + \log[\mathsf{q}[\mathbf{x}_0]^\eta] - \log \Big[\sum_{\mathbf{x}} \mathsf{q}[\mathbf{x}]^\eta \Big] \leq -\eta \sum_{t=1}^T \mathop{\mathbb{E}}_{a \sim \mathsf{p}_t} [l_t[a]] + \frac{\eta^2 T}{8},$$

which can be rearranged as

$$\begin{aligned}
&\sum_{t=1}^T \mathop{\mathbb{E}}_{a \sim \mathsf{p}_t} [l_t[a]] - \sum_{t=1}^T l_t[\mathbf{x}_0[t]] \leq \frac{\eta T}{8} - \log[\mathsf{q}[\mathbf{x}_0]] + \frac{1}{\eta} \log \Big[\sum_{\mathbf{x}} \mathsf{q}[\mathbf{x}]^\eta \Big] \\
\Leftrightarrow &\sum_{t=1}^T \mathop{\mathbb{E}}_{a \sim \mathsf{p}_t} [l_t[a]] - \sum_{t=1}^T l_t[\mathbf{x}_0[t]] + \log[K \mathsf{q}[\mathbf{x}_0]] \leq \frac{\eta T}{8} + \frac{1}{\eta} \log \Big[K^\eta \sum_{\mathbf{x}} \mathsf{q}[\mathbf{x}]^\eta \Big].
\end{aligned}$$

Since the inequality holds for any sequence $\mathbf{x}_0 \in \Sigma^T$, it implies the following upper bound on the weighted regret:

$$\mathrm{Reg}_T \leq \frac{\eta T}{8} + \frac{1}{\eta} \log \Big[K^\eta \sum_{\mathbf{x}} \mathsf{q}[\mathbf{x}]^\eta \Big].$$

By Jensen's inequality, the inequality $\frac{1}{K} \sum_{\mathbf{x}} \mathsf{q}[\mathbf{x}]^\eta \leq \Big(\frac{1}{K} \sum_{\mathbf{x}} \mathsf{q}[\mathbf{x}] \Big)^\eta = \frac{1}{K^\eta}$ holds for $\eta \in (0, 1)$. This implies the following general upper bounds on the weighted regret:

$$\mathrm{Reg}_T \leq \frac{\eta T}{8} + \frac{1}{\eta} \log K.$$

The weighted regret can also be upper bounded in terms of the Rényi entropy. Observe that

$$\frac{\eta T}{8} + \frac{1}{\eta} \log \Big[K^\eta \sum_{\mathbf{x}} \mathsf{q}[\mathbf{x}]^\eta \Big] = \frac{\eta T}{8} + \frac{1-\eta}{\eta} H_\eta(\mathsf{q}) + \log K.$$

$\eta \mapsto H_\eta(\mathsf{q})$ is known to be a non-increasing function (see e.g. [Van Erven and Harremos, 2014]). It follows that $\eta \mapsto \frac{\eta}{\sqrt{H_\eta(\mathsf{q})}}$ is an increasing function that increases at least linearly. If we assume that q is supported on more than a single sequence, then, we have $H_0(\mathsf{q}) > 0$. Thus, for any T, there exists a unique η^* such that $\frac{\eta^*}{\sqrt{H_{\eta^*}(\mathsf{q})}} = \sqrt{\frac{8}{T}}$. Furthermore, for $\eta \leq \eta^*$, the following inequality holds: $\frac{\eta}{\sqrt{H_\eta(\mathsf{q})}} \leq \sqrt{\frac{8}{T}}$. Thus, we can write

$$\begin{aligned} \frac{\eta T}{8} + \frac{1}{\eta} \log \Big[K^\eta \sum_{\mathbf{x}} \mathsf{q}[\mathbf{x}]^\eta \Big] &\leq \inf_{\eta \leq \eta^*} \frac{\eta T}{8} + \frac{1}{\eta} H_\eta(\mathsf{q}) - H_\eta(\mathsf{q}) + \log K \\ &\leq \sqrt{\frac{T H_{\eta^*}(\mathsf{q})}{2}} - H_{\eta^*}(\mathsf{q}) + \log K. \end{aligned}$$

The upper bound on the unweighted regret is obtained straightforwardly from the previous derivations using $\mathsf{q}[\mathbf{x}] = \frac{1}{K}$. □

Note that when the losses are *mixing*, we can also derive better constant-in-time regret guarantees by avoiding the use of Hoeffding's inequality.

D Proof of Theorem 2

Theorem 2. *The weighted regret of the* AWM *algorithm with respect to the WFA $\mathcal{C}$ when run with $\widehat{\mathcal{C}}_T$ instead of $\mathcal{C}_T$ can be upper bounded as follows:*

$$\mathrm{Reg}_T(\mathcal{A}, \mathcal{C}) \leq \frac{\eta T}{8} + \frac{1}{\eta} \log \Big[K^\eta \sum_{\mathbf{x}} \widehat{\mathsf{q}}[\mathbf{x}]^\eta \Big] + D_\infty(\mathsf{q} \| \widehat{\mathsf{q}}) \leq \frac{\eta T}{8} + \frac{1}{\eta} \log K + D_\infty(\mathsf{q} \| \widehat{\mathsf{q}}).$$

Its unweighted regret can be upper bounded as follows:

$$\mathrm{Reg}^0_T(\mathcal{A}, \mathcal{C}) \leq \max_{\mathcal{C}(\mathbf{x}) > 0} \frac{\eta T}{8} + \frac{1}{\eta} \log \Big[\frac{1}{\mathsf{q}[\mathbf{x}]} \Big] + \frac{1}{\eta} D_\infty(\mathsf{q} \| \widehat{\mathsf{q}}).$$

Proof. By Theorem 1 (and its proof), for any sequence $\mathbf{x}_0 \in \Sigma^T$, the following upper bound holds for the cumulative loss of AWM run with $\widehat{\mathcal{C}}_T$:

$$\sum_{t=1}^T \mathsf{p}_t \cdot \mathbf{l}_t - \sum_{t=1}^T l_t[\mathbf{x}_0[t]] + \log[\widehat{\mathsf{q}}[\mathbf{x}_0]] \leq \frac{\eta T}{8} + \frac{1}{\eta} \log \Big[\sum_{\mathbf{x}} \widehat{\mathsf{q}}[\mathbf{x}]^\eta \Big].$$

Thus, for any sequence $\mathbf{x}_0 \in \Sigma^T$ accepted by $\mathcal{C}_T$, we can write

$$\sum_{t=1}^T \mathsf{p}_t \cdot \mathbf{l}_t - \sum_{t=1}^T l_t[\mathbf{x}_0[t]] + \log[\mathsf{q}[\mathbf{x}_0] K] \leq \frac{\eta T}{8} + \frac{1}{\eta} \log \Big[K^\eta \sum_{\mathbf{x}} \widehat{\mathsf{q}}[\mathbf{x}]^\eta \Big] + \log \Big[\frac{\mathsf{q}[\mathbf{x}_0]}{\widehat{\mathsf{q}}[\mathbf{x}_0]} \Big],$$

which implies the following upper bound on the weighted regret:

$$\begin{aligned}\mathrm{Reg}_T(\mathcal{A}, \mathcal{C}) &\leq \frac{\eta T}{8} + \frac{1}{\eta} \log \left[K^\eta \sum_{\mathbf{x}} \widehat{\mathsf{q}}[\mathbf{x}]^\eta \right] + \sup_{\mathcal{C}_T(\mathbf{x}_0) > 0} \log \left[\frac{\mathsf{q}[\mathbf{x}_0]}{\widehat{\mathsf{q}}[\mathbf{x}_0]} \right] \\ &\leq \frac{\eta T}{8} + \frac{1}{\eta} \log \left[K^\eta \sum_{\mathbf{x}} \widehat{\mathsf{q}}[\mathbf{x}]^\eta \right] + D_\infty(\mathsf{q} \| \widehat{\mathsf{q}})\end{aligned}$$

As in the proof of Theorem 1, by Jensen's inequality, $\log \left[K^\eta \sum_{\mathbf{x}} \widehat{\mathsf{q}}[\mathbf{x}]^\eta \right] \leq \log K$, which implies the second inequality.

Similarly, by the proof of Theorem 1, the unweighted regret of AWM run with $\widehat{\mathcal{C}}_T$ can be upper bounded as follows:

$$\sum_{t=1}^T \mathsf{p}_t \cdot \mathbf{l}_t - \sum_{t=1}^T l_t[z_t] \leq \max_{\mathcal{C}(\mathbf{x}) > 0} \frac{\eta T}{8} + \frac{1}{\eta} \log \left[\frac{1}{\widehat{\mathsf{q}}[\mathbf{x}]} \right] = \max_{\mathcal{C}(\mathbf{x}) > 0} \frac{\eta T}{8} + \frac{1}{\eta} \log \left[\frac{1}{\mathsf{q}[\mathbf{x}]} \right] + \frac{1}{\eta} \log \left[\frac{\mathsf{q}[\mathbf{x}]}{\widehat{\mathsf{q}}[\mathbf{x}]} \right],$$

which completes the proof. □

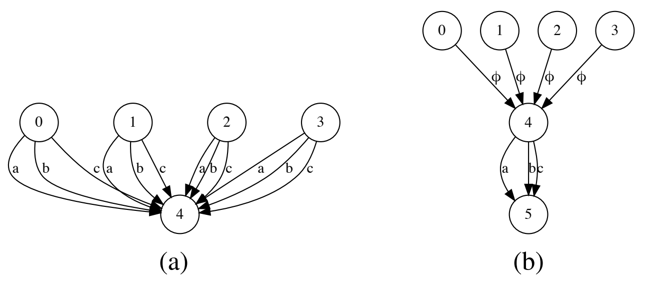

Figure 5: Example of the compression achieved by introducing a failure transition. (a) Standard automaton. (b) φ-automaton.

Algorithm 3: φ-CONVERT.

```
Algorithm: $\varphi$-CONVERT($\mathcal{C}$)
for each non-initial state $q \in \mathcal{C}$ do
    $S^*, Q^* \leftarrow \varphi$-SOURCESUBSET($\mathcal{C}, q$)
    if $|S^*| + |Q^*| < |S^*||Q^*|$ then
        $\tilde{q} \leftarrow$ NEWSTATE($\mathcal{C}$)
        $E_{\mathcal{C}} \leftarrow E_{\mathcal{C}} \cup \{(q, \varphi, 1, \tilde{q})\}$
        for each $q' \in Q^*$ do
            for each $e' \in E_{\mathcal{C}}[q']$ do
                if $(\mathrm{lab}[e'], \mathrm{weight}[e']) \in S^*$ then
                    $E_{\mathcal{C}} \leftarrow E_{\mathcal{C}} \cup \{(\tilde{q}, \mathrm{lab}[e'], \mathrm{weight}[e'], q)\}$
                    DELETETRANSITION$[E_{\mathcal{C}}, e']$
```

E Failure transition algorithms

The computational complexity of the AWM algorithm presented in Section 3 is based on the size of the composed automaton $\mathcal{C} \cap \mathcal{S}_T$, which itself is related to the original size of $\mathcal{C}$. Similarly, if we were to apply AWM to an n-gram approximation, the computational complexity of the algorithm depends on the size of the approximating automaton. In this section, we introduce a technique to improve the computational cost of AWM by reducing the size of the automaton, using the notion of *failure transition (or φ-transition)*.

φ-transitions are special transitions characterized by the semantic of “other”. If, at a state q, there is no outgoing transition labeled with $a \in \Sigma$ and there is a φ-transition leaving q and reaching q', then the failure transition is taken instead without consuming the label, and the next state is determined using the transitions leaving q'. A φ-automaton is an automaton with φ-transitions. We assume that there is no φ-cycle in any of our φ-automata, and that there is at most one failure transition leaving any state. This implies that the number of consecutive failure transitions taken is bounded.

A failure transition can often replicate the role of multiple standard transitions when there is “symmetry” within an automaton, that is when there are many transitions leading to the same state from different states that consume the same set of labels. Figure 5 illustrates such a case.

E.1 Conversion

Notice that in Figure 5, the introduction of a failure transition removed $|S|$ transitions from $|Q|$ parent states while introducing $|Q|$ φ-transitions from each of the parent states to a new state q', and $|S|$ transitions from q' to q. Thus, the change in the number of transitions is $|S^*| + |Q^*| - |S^*||Q^*|$. This fact can be exploited to design an algorithm that iterates through the states of an automaton, and for each state, determines whether it is beneficial to introduce a failure transition between that state and (a subset of) its parents. We call this algorithm, φ-Convert, which uses another algorithm, φ-SOURCESUBSET as a subroutine to greedily select a candidate set of parent states from which to introduce a φ-transition for each state. The pseudocode for φ-CONVERT and φ-SOURCESUBSET are presented in Algorithm 3 and Algorithm 4 respectively.

Recall that the two main automata operations required for AWM are intersection and shortest-distance. While these two operations are standard for weighted automata, it is not as clear how one can perform them over weighted φ-automata. We now extend both to φ-automata.

Algorithm 4: φ-SOURCESUBSET.

```
Algorithm: $\varphi$-SOURCESUBSET($\mathcal{C}, q$)
$(S_0, Q_0) \leftarrow (\emptyset, \emptyset)$
$k^* \leftarrow 1$
for $k \leftarrow 1$ to $|$ Parents $[q]|$ do
    $q_k \leftarrow \operatorname{argmax}_{q' \in \text{Parents}[q] \setminus Q_{k-1}} |(a, w) \in \Sigma \times \mathbb{R}_+ : \forall \tilde{q} \in \text{Parents}[q] \cup \{q'\}, (\tilde{q}, a, w, q) \in E_{\mathcal{C}}|$
    $S_k \leftarrow |(a, w) \in \Sigma \times \mathbb{R}_+ : \forall \tilde{q} \in \text{Parents}[q] \cup \{q_k\}, (\tilde{q}, a, w, q) \in E_{\mathcal{C}}|$
    $Q_k \leftarrow Q_{k-1} \cup \{q_k\}$
    $k^* \leftarrow \operatorname{argmax}_{j \in \{k, k^*\}} \{|S_j||Q_j| - (|S_j| + |Q_j|)\}$
return $(S_{k^*}, Q_{k^*})$
```

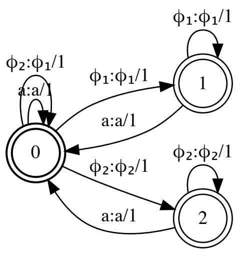

Figure 6: Illustration of the φ-filter $\mathcal{F}$.

E.2 Intersection using a φ-filter

One of the main automata operations required for AWM is intersection. The standard algorithm for intersection of automata (Appendix A), which is based on matching transitions, can return an incorrect result in the presence of φ-transitions. Specifically, the algorithm may produce multiple φ-paths between two states, which leads to redundancy and incorrect weights.

Redundant φ-paths are generated by standard intersection algorithms because when the algorithm is in state q_1 in WFA $\mathcal{C}_1$ and state q_2 in $\mathcal{C}_2$, both of which contain outgoing φ-transitions, the algorithm may take any of the following steps: (1) move forward on a φ-transition in $\mathcal{C}_1$ while staying at q_2; (2) move forward on a φ-transition in $\mathcal{C}_2$ while staying in $\mathcal{C}_1$; or (3) move forward in both $\mathcal{C}_1$ and $\mathcal{C}_2$.

To avoid this situation, we introduce the concept of a *φ-filter*, which is a *finite state transducer (FST)* that can filter out all but one φ-path between any two states.

Our φ-filter is designed to modify the two input automata in a way that will distinguish between the above cases. In $\mathcal{C}_1$, for every φ-transition, we rename the label φ as φ_2. Moreover, at the source and destination states of every φ-transition, we introduce new self-loop transitions labeled with φ_1 and with weight 1. Thus, a transition labeled with φ_2 will indicate a "move forward," while a transition labeled with φ_1 will indicate a "stay." Similarly, in $\mathcal{C}_2$, we rename the φ labels as φ_1, and we introduce self-loops labeled with φ_2 and weight 1 at the source and destination states of every φ-transition. With these modifications, any φ-path resulting from the composition algorithm will include transitions of the form: (1) $(\varphi_2 : \varphi_2)$; (2) $(\varphi_1 : \varphi_1)$; or (3) $(\varphi_2 : \varphi_1)$.

Now consider the finite-state transducer $\mathcal{F}$ illustrated in Figure 6, which will serve as our φ-filter. The composition of any two φ-automata and the φ-filter $\mathcal{F}$, $\mathcal{C}_1 \circ \mathcal{F} \circ \mathcal{C}_2$, will result in a finite-state transducer whose transitions have labels in $\{(a : a)\}_{a \in \Sigma} \cup \{(\varphi_2 : \varphi_2), (\varphi_1 : \varphi_1), (\varphi_2 : \varphi_1)\}$.[8] Moreover, we identify all label pairs in $\{(\varphi_2 : \varphi_2), (\varphi_1 : \varphi_1), (\varphi_2 : \varphi_1)\}$ using the same semantic of "other" as we did with φ. Thus, we can identify all label pairs in $\{(\varphi_2 : \varphi_2), (\varphi_1 : \varphi_1), (\varphi_2 : \varphi_1)\}$ with the single pair $(\varphi : \varphi)$ and treat the result of composition as simply a weighted finite automaton.

[8]Composition is a standard algorithm for weighted finite-state transducers which coincides with the intersection operation in the special case of WFA (see Mohri [2009]).

Algorithm 5: φ-AutomataWeightedMajority(φ-AWM).

```
Algorithm: $\varphi$-AWM($\mathcal{C}$, $\eta$)
$\mathcal{C} \leftarrow \varphi$-Convert($\mathcal{C}$)
$\mathcal{B} \leftarrow \mathcal{C} \cap \mathcal{F} \cap \mathcal{S}_T$
$\mathcal{A} \leftarrow$ Weight-Pushing($\mathcal{B}^\eta$)
$\boldsymbol{\beta} \leftarrow$ BwdDist($\mathcal{A}$)
$\boldsymbol{\alpha} \leftarrow 0$; $\boldsymbol{\alpha}[I_\mathcal{A}] \leftarrow 1$
for each $e \in E_\mathcal{A}^{0\to 1}$ do
    $\mathsf{p}_1[\mathrm{lab}[e]] \leftarrow \mathrm{weight}[e]$.
for $t \leftarrow 1$ to $T$ do
    $i_t \leftarrow$Sample($\mathsf{p}_t$); Play($i_t$); Receive($\mathbf{l}_t$)
    $Z \leftarrow 0$; $\mathbf{w} \leftarrow 0$
    for each $e \in E_\mathcal{A}^{t\to t+1}$ do
        $\mathrm{weight}[e] \leftarrow \mathrm{weight}[e]\, e^{-\eta l_t[\mathrm{lab}[e]]}$
        $\mathbf{w}[\mathrm{lab}[e]] \leftarrow \mathbf{w}[\mathrm{lab}[e]] + \boldsymbol{\alpha}[\mathrm{src}[e]]\,\mathrm{weight}[e]\,\boldsymbol{\beta}[\mathrm{dest}[e]]$
        $Z \leftarrow Z + \mathbf{w}[\mathrm{lab}[e]]$
        $\boldsymbol{\alpha}[\mathrm{dest}[e]] \leftarrow \boldsymbol{\alpha}[\mathrm{dest}[e]] + \boldsymbol{\alpha}[\mathrm{src}[e]]\,\mathrm{weight}[e]$
        if $\mathrm{lab}[e] \neq \varphi$ then
            $\tilde{q} = \mathrm{src}[e]$; $w_\varphi \leftarrow 1$
            while $\exists e_\varphi \in E[\tilde{q}]$ with $\mathrm{lab}[e_\varphi] = \varphi$ do
                $w_\varphi \leftarrow w_\varphi\,\mathrm{weight}[e_\varphi]$
                if $\exists e' \in E[\mathrm{dest}[e_\varphi]]$ with $\mathrm{lab}[e'] = \mathrm{lab}[e]$ then
                    $\alpha[\mathrm{dest}[e']] \leftarrow \alpha[\mathrm{dest}[e']] - w_\varphi \mathrm{weight}[e']$
                    Break
                else
                    $\tilde{q} \leftarrow \mathrm{dest}[e_\varphi]$
    $\mathsf{p}_{t+1} \leftarrow \frac{\mathbf{w}}{Z}$
```

E.3 Update of $\boldsymbol{\alpha}$ using a modified shortest-distance algorithm

The other key ingredient of the AWM algorithm is the update of $\boldsymbol{\alpha}$ using the shortest-distance algorithm for WFA. However, updating $\boldsymbol{\alpha}$ as we did in AWM may result in summing over 'obsolete φ-transitions'. For example, if at a given state q, there is a transition labeled with a to q' and a φ-transition whose destination state has a single outgoing transition also labeled with a to q', the second path should not be considered.

To account for these types of situations, we use the fact that the semiring $(\mathbb{R}_+, +, \times, 0, 1)$ admits a natural extension to a ring structure under the standard additive inverse -1. Specifically, upon encountering a transition e labeled with a leaving state q, we will check for φ-transitions with destination states that admit further transitions e' labeled with a. Any such transition should not be considered under the semantic of the φ-transition and thus should not contribute any weight to the distance to $\alpha[\mathrm{dest}[e']]$. To correctly account for these paths, we will *preemptively* subtract the weight of e' from its destination state. When the algorithm processes the φ-transition directly, it will add this weight back so that the total contribution of this path is zero.

E.4 φ-AWM algorithm

With the addition of the φ-filter and the modified $\boldsymbol{\alpha}$ update described above, we can present φ-AWM, an extension of AWM that can handle φ-automata. Given an input automaton (not necessarily with φ-transitions), the algorithm first calls φ-Convert to determine whether it is beneficial to introduce φ-transitions. The algorithm then composes the output with Σ^T (using the φ-filter) to compute the set of sequences of length T that are accepted by $\mathcal{C}$. Then, the algorithm updates the weights of the automaton in a similar manner as in AWM with the additional adjustment of preemptively accounting for φ-transitions. Algorithm 5 presents the pseudocode for φ-AWM.

Since the update of p_t in φ-AWM is mathematically equivalent to the one in AWM we obtain the same regret guarantees as in Theorem 1. Moreover, if we denote by $N_\varphi(Q_{\mathcal{C}_T})$ the maximum number of consecutive φ-transitions leaving states in $Q_{\mathcal{C}_T}$, the total computational cost of the algorithm is in $O\left(\sum_{t=1}^T N_\varphi(Q_{\mathcal{C}_T, t-1})|E_\mathcal{A}^{t\to t+1}|\right)$.

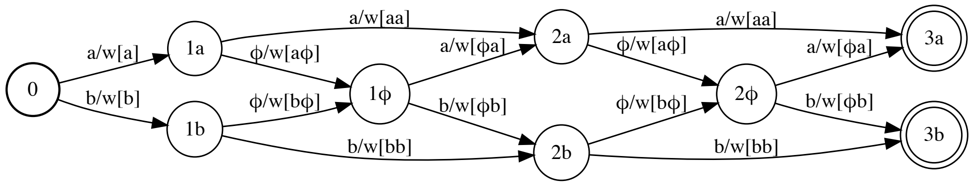

Figure 7: An illustration of a bigram model approximating the k-shifting automaton composed with $\mathcal{S}$. φ-CONVERT has been applied to the bigram model, making it smaller than a standard bigram model.

For the k-shifting automaton, the per-iteration computational complexity of φ-AWM is now $O(Nk)$, since there is at most one consecutive φ-transition in the output of φ-Convert, and we now aggregate transitions at each time using failure transitions. This is a factor of N better than that of AWM, and only a factor of k worse than the FIXED-SHARE algorithm of Herbster and Warmuth [1998]. If we intersect the k-shifting automaton with Σ^T, approximate the result with a bigram model, and then convert this model into a φ-automaton, we obtain an algorithm that runs in $O(N)$, which is the same as that of FIXED-SHARE. See Figure 7 for an illustration.

Algorithm 6: PROD-EG.

```
Algorithm: PROD-EG($\mathsf{q}_1 \in (\Delta_N)^m, \eta$)
for $s = 1, 2, \ldots, \tau$ do
    PLAY($\mathsf{q}_s$)
    RECEIVE($\nabla f(\mathsf{q}_s)$)
    for $j = 1, 2, \ldots, m$ do
        for $i = 1, 2, \ldots, N$ do
            $\mathsf{q}_{s+1,j}(i) = \mathsf{q}_{s,j} e^{-\eta \frac{\partial f}{\partial \mathsf{q}_j(i)}(\mathsf{q}_{s,j})}$
```

F PROD-EG

The pseudocode of the PROD-EG algorithm, which is based on a simple multiplicative update, is given in Algorithm 6. The following provides a general guarantee for the convergence of the algorithm.

Theorem 5 (PRODUCT-EXPONENTIATED GRADIENT (PROD-EG))**.**

Let $(\Delta_N)^m$ be the product of m $(N-1)$-dimensional simplices, and let $f\colon (\Delta_N)^m \to \mathbb{R}$ be a convex function whose partial subgradients have absolute values all bounded by L. Let $\mathsf{q}_{1,j}(i) = \frac{1}{N}$ for $i \in [N]$ and $j \in [m]$. Then, PROD-EG *benefits from the following guarantee:*

$$f\left(\frac{1}{\tau}\sum_{s=1}^{\tau} \mathsf{q}_s\right) - f(\mathsf{q}^*) \leq \frac{1}{\eta\tau} m \log(N) + 2\eta L.$$

Proof. Consider the mirror map $\psi\colon (\Delta_N)^m \to \mathbb{R}$ defined by $\psi(\mathsf{q}) = \sum_{j=1}^{m}\sum_{i=1}^{N} \mathsf{q}_j(i)\log \mathsf{q}_j(i)$. This induces the Bregman divergence:

$$B_\psi(\mathsf{q}, \mathsf{q}') = \sum_{j=1}^{m}\sum_{i=1}^{N} \mathsf{q}_j(i) \log\left(\frac{\mathsf{q}_j(i)}{\mathsf{q}'_j(i)}\right).$$

Since each relative entropy is 1-strongly convex with respect to the l_1 norm over a single simplex, the additivity of strong convexity implies that B_ψ is 1-strongly convex with respect to the l_1 norm defined over$(\Delta_N)^m$.

The update described in the theorem statement corresponds to the mirror descent update based on B_ψ:

$$\mathsf{q}_{s+1} = \operatorname*{argmin}_{\mathsf{q}\in(\Delta_N)^m} \langle g_s, \mathsf{q}\rangle + B_\psi(\mathsf{q}, \mathsf{q}_s).$$

where $g_s \in \partial(f(\mathsf{q}_s))$ is an element of the subgradient of f at q_s. Thus, the standard mirror descent regret bound (e.g. [Bubeck et al., 2015]) implies that

$$\frac{1}{\tau}\sum_{s=1}^{\tau} f(\mathsf{q}_s) - f(\mathsf{q}^*) \leq \frac{1}{\eta\tau} B_\psi(\mathsf{q}^*, \mathsf{q}_1) + \eta 2L.$$

The result now follows from the fact the observation that $B_\psi(\mathsf{q}^*, \mathsf{q}_1) \leq m\log(N)$. □

For the minimum Rényi divergence optimization problem (8), we can apply PROD-EG to the product of $m = \sum_{j=1}^{n} |\Sigma|^{n-j}$ simplices, each one corresponding to a conditional probability with a specific history. First, we remark that the subgradient of the maximum of a family of convex functions at a point can always be chosen from the subgradient of the maximizing function at that point. Specifically, let $\{f_\alpha\}_{\alpha\in\mathcal{A}}$ be a family of convex functions, and let $\alpha(x) = \operatorname{argmax}_\alpha f_\alpha(x)$. Then, it follows that

$$\max_\alpha f_\alpha(x) - \max_\alpha f_\alpha(y) \geq f_{\alpha(y)}(x) - f_{\alpha(y)}(y) \geq \langle \nabla f_{\alpha(y)}(y), x - y\rangle.$$

Let $\mathbf{x}$ be the maximizing path of the minimum Rényi divergence objective. We can then write

$$\begin{aligned}\log\left[\frac{\mathsf{q}[\mathbf{x}]}{\widehat{\mathsf{q}}_{\mathsf{w}}[\mathbf{x}]}\right] &= \mathsf{q}[\mathbf{x}] - \sum_{t=1}^{T} \log \mathsf{w}\big[\mathbf{x}[t] \big| \mathbf{x}_{\max(t-n+1,1)}^{t-1}\big] \\ &= \mathsf{q}[\mathbf{x}] - \sum_{j=1}^{n} \sum_{\mathbf{z}_1^j \in \Sigma^j} \sum_{t=1}^{T} 1_{j=\min(t,n)} 1_{\mathbf{x}_{\max(t-n+1,1)}^{t} = \mathbf{z}_1^j} \log \mathsf{w}\Big[\mathbf{z}[j] \Big| \mathbf{z}_1^{j-1}\Big].\end{aligned}$$

Thus, its partial derivative with respect to $\mathsf{w}\Big[\mathbf{z}[j] \Big| \mathbf{z}_1^{j-1}\Big]$ is:

$$\frac{\partial}{\partial \mathsf{w}\Big[\mathbf{z}[j] \Big| \mathbf{z}_1^{j-1}\Big]} \log\left[\frac{\mathsf{q}[\mathbf{x}]}{\widehat{\mathsf{q}}_{\mathsf{w}}[\mathbf{x}]}\right] = -\sum_{t=1}^{T} \frac{1_{j=\min(t,n)} 1_{\mathbf{x}_{\max(t-n+1,1)}^{t} = \mathbf{z}_1^j}}{\mathsf{w}\Big[\mathbf{z}[j] \Big| \mathbf{z}_1^{j-1}\Big]}.$$

Thus, by tuning Prod-EG with an adaptive learning rate

$$\eta_t \propto \frac{1}{\sqrt{\sum_{s=1}^{t} \left\| \nabla \log\left[\frac{\mathsf{q}[\mathbf{x}(s)]}{\widehat{\mathsf{q}}_{\mathsf{w}_s}[\mathbf{x}(s)]}\right] \right\|_\infty^2}},$$

where $\mathbf{x}(s) = \operatorname{argmax}_{\mathbf{x} \in \mathcal{C}_T} \log\left[\frac{\mathsf{q}[\mathbf{x}]}{\widehat{\mathsf{q}}_{\mathsf{w}_s}[\mathbf{x}]}\right]$, we can derive the following guarantee for Prod-EG applied to the n-gram approximation problem.

Corollary 1 (n-gram approximation guarantee)**.** *There exists an optimization algorithm outputting a sequence of conditional probabilities* $(\mathsf{q}_t)_{t=1}^{\infty}$ *such that* $\left(\frac{1}{T}\sum_{t=1}^{T} \mathsf{q}_t\right)$ *approximates the* ∞*-Rényi optimal* n*-gram solution with the following guarantee:*

$$F\left(\frac{1}{\tau}\sum_{s=1}^{\tau} \mathsf{q}_t\right) - F(\mathsf{q}^*) \leq \sqrt{\frac{2N^n \log(N) \sum_{s=1}^{\tau} \max_{\substack{j \in [n] \\ \mathbf{z}_1^j \in \Sigma^j}} \left| \sum_{t=1}^{T} \frac{1_{j=\min(t,n)} 1_{\mathbf{x}_{\max(t-n+1,1)}^{t} = \mathbf{z}_1^j}}{\mathsf{w}_s\Big[\mathbf{z}[j] \Big| \mathbf{z}_1^{j-1}\Big]} \right|}{(N-1)T^2}}.$$

Each iteration of Prod-EG admits a computational complexity that is linear in the dimension of the feature space. Since we have specified an n-gram model as the product of $\frac{N^n-1}{N-1}$ simplices, the total per-iteration cost of solving the convex optimization problem is in $O\left(\frac{N(N^n-1)}{N-1}\right) = O(N^n)$. Since the minimum Rényi divergence is not Lipschitz, the maximizing ratio in the convergence guarantee may also become large when the choice of n is too small. In all cases, observe that this approximation problem can be solved offline.

G Minimum Rényi divergence unigram models

Theorem 3. *Assume that $\mathcal{C}_T$ admits uniform weights over all paths and $\Sigma = \{a_1, a_2\}$. For $j \in \{1, 2\}$, let $n(a_j)$ be the smallest number of occurrences of a_j in a path of $\mathcal{C}_T$. For any $j \in \{1, 2\}$, define*

$$\mathsf{q}[a_j] = \frac{\max\left\{1, \frac{n(a_j)}{T-n(a_j)}\right\}}{1+\max\left\{1, \frac{n(a_j)}{T-n(a_j)}\right\}}.$$

Then, the unigram model $\mathsf{w} \in \mathcal{W}_1$ solution of ∞-Rényi divergence optimization problem is defined by $\mathsf{w}[a_{j^*}] = \mathsf{q}[a_{j*}]$, $\mathsf{w}[a_{j'}] = 1 - \mathsf{w}[a_{j^*}]$, with $j^* = \operatorname{argmax}_{j \in \{1,2\}}\ n(a_j) \log \mathsf{q}[a_j] + [T - n(a_j)] \log \big[1 - \mathsf{q}[a_j]\big]$.

Proof. We seek a unigram distribution q_w that is a solution of:

$$\min_{\mathsf{w} \in \mathcal{W}_1} \sup_{\mathbf{x} \in \mathcal{C}_T} \log\left(\frac{\mathsf{q}[\mathbf{x}]}{\mathsf{q}_\mathsf{w}[\mathbf{x}]}\right).$$

Since $\mathcal{C}_T$ admits uniform weights, $\mathsf{q}[\mathbf{x}] = \frac{1}{|\mathcal{C}_T|}$, and since q_w is the distribution induced by a unigram model, $\log \mathsf{q}_\mathsf{w}[\mathbf{x}]$ can be expressed as follows:

$$\log \mathsf{q}_\mathsf{w}[\mathbf{x}] = n_\mathbf{x}(a_1) \log p(a_1) + [T - n_\mathbf{x}(a_1)] \log\left(1 - p(a_1)\right),$$

where $p(a_j)$ is the automaton's weight on transitions labeled with a_j and $n_\mathbf{x}(a_j)$ is the count of a_j in the sequence $\mathbf{x}$. Thus, the optimization problem is equivalent to the following problem:

$$-\max_{p(a_1) \in [0,1]} \min_{\mathbf{x} \in \mathcal{C}_T} n_\mathbf{x}(a_1) \log p(a_1) + [T - n_\mathbf{x}(a_1)] \log\left(1 - p(a_1)\right).$$

Denote the objective by $F(p(a_1), n_\mathbf{x}(a_1))$. Then, the partial derivatives with respect to the label counts are given by

$$\frac{\partial F}{\partial n_\mathbf{x}(a_1)} = \log p(a_1) - \log\Big(1 - p(a_1)\Big).$$

Thus, $\frac{\partial F}{\partial n_\mathbf{x}(a_1)} \geq 0$ if and only if $p(a_1) \geq 1 - p(a_1)$. Furthermore, if $p(a_1) \geq 1 - p(a_1)$, then the sequence $\mathbf{x}$ chosen in the optimization problem is the sequence with the minimal count of symbol a_1. Similarly, if $p(a_2) \geq 1 - p(a_2)$, then the sequence $\mathbf{x}$ chosen in the optimization problem is the one with minimal count of a_2.

Since we have either $p(a_1) \geq p(a_2)$ or vice versa (potentially both), we can write the optimization problem as:

$$-\max_{k \in \{1,2\}} \max_{\substack{p(a_j) \geq 1 - p(a_j) \\ j \neq k}} \min_{\{n_\mathbf{x}(a_j)\}_{j \neq k}\,:\, \mathbf{x} \in \mathcal{C}_T} n_\mathbf{x}(a_j) \log p(a_j) + [T - n_\mathbf{x}(a_j)] \log\left(1 - p(a_j)\right).$$

Given $k \in \{1, 2\}$, let $\mathbf{x}(k)$ be the sequence that minimizes $n_\mathbf{x}(a_j)$ over all $\mathbf{x}$ for $j \neq k$. Denote these counts $n_{\mathbf{x}(k)}(a_j)$ by $n(a_j)$. Then we can rewrite the objective as:

$$-\max_{k=1,2,\ldots,N} \max_{\substack{p(a_j) \geq 1 - p(a_j) \\ j \neq k}} n(a_j) \log p(a_j) + [T - n(a_j)] \log\left(1 - p(a_j)\right).$$

Denote the objective for this new term by $\tilde{F}_k$, which is a function of $p(a_j)$. The partial derivative of $\tilde{F}_k$ with respect to $p(a_j)$ is:

$$\frac{\partial \tilde{F}_k}{\partial p(a_j)} = \frac{n(a_j)}{p(a_j)} - \frac{T - n(a_j)}{1 - p(a_j)},$$

which is equal to 0 if and only if

$$p(a_j) = \frac{n(a_j)}{T - n(a_j)}\left(1 - p(a_j)\right) = \max\left\{1, \frac{n(a_j)}{T - n(a_j)}\right\}\left(1 - p(a_j)\right).$$

The last equality follows from our assumption that $p(a_j) \geq 1 - p(a_j)$. Now, let $\mathsf{q}(a_j)$ denote the probabilities that we have just computed. Then, we can write the optimization problem of $\tilde{F}_k$ as:

$$-\max_{k \in \{1,2\}, j \in \{1,2\} \setminus \{k\}} n(a_j) \log \mathsf{q}(a_j) + [T - n(a_j)] \log\left(1 - \mathsf{q}(a_j)\right).$$

□

H Maximum likelihood n-gram models

Theorem 4. *Let $\mathcal{C}_T$ be the k-shifting automaton for some k. Then, the bigram model w_2 obtained by minimizing relative entropy is defined for all $a_1, a_2 \in \Sigma$ by*

$$\mathsf{q}_{\mathsf{w}_2}[a_1 a_2] = \frac{1}{N}\Big[1 - \frac{k}{(T-1)}\Big] 1_{a_1 = a_2} + \frac{1}{N}\Big[\frac{k}{(T-1)(N-1)}\Big] 1_{a_1 \neq a_2}.$$

Moreover, its approximation error can be bounded by a constant (independent of T):

$$D_\infty(\mathsf{q} \| \mathsf{q}_{\mathsf{w}_2}) \leq -\log\left[1 - 2e^{-\frac{1}{12k}}\right].$$

Proof. Let $a_1, a_2 \in \Sigma$. Then, we can write

$$\mathsf{q}_{\mathsf{w}_2}[a_2|a_1] = \mathsf{q}_{\mathsf{w}_2}[a_2|a_1, a_2 = a_1]\, \mathsf{q}_{\mathsf{w}_2}[a_2 = a_1] + \mathsf{q}_{\mathsf{w}_2}[a_2|a_1, a_2 \neq a_1]\, \mathsf{q}_{\mathsf{w}_2}[a_2 \neq a_1].$$

Consider first the case where $a_2 = a_1$. Then, $\mathsf{q}_{\mathsf{w}_2}[a_2|a_1, a_2 = a_1] = 1$, and $\mathsf{q}_{\mathsf{w}_2}[a_2 = a_1]$ is the expected number of times that we see label a_2 agreeing with label a_1. Since q is uniform for the k-shifting automaton, the expected counts are pure counts, and the probability that we see two consecutive labels agreeing is $1 - \frac{k}{T-1}$. Now, consider the case where $a_2 \neq a_1$. By symmetry, $\mathsf{q}_{\mathsf{w}_2}[a_2|a_1, a_2 \neq a_1] = \frac{1}{N-1}$, since a_2 is equally likely to be any of the other $N-1$ labels. Moreover, $\mathsf{q}_{\mathsf{w}_2}[a_2 \neq a_1] = \frac{k}{T-1}$. Thus, the following holds:

$$\mathsf{q}_{\mathsf{w}_2}[a_2|a_1] = \frac{1}{N-1}\frac{k}{T-1} 1_{a_1 \neq a_2} + \left[1 - \frac{k}{T-1}\right] 1_{a_1 = a_2}.$$

By symmetry, we can write $\mathsf{q}_{\mathsf{w}_2}[a_1] = \frac{1}{N}$, therefore,

$$\mathsf{q}_{\mathsf{w}_2}[a_1 a_2] = \mathsf{q}_{\mathsf{w}_2}[a_2|a_1]\mathsf{q}_{\mathsf{w}_2}[a_1] = \frac{k}{N(N-1)(T-1)} 1_{a_1 \neq a_2} + \left[\frac{T-1-k}{N(T-1)}\right] 1_{a_1 = a_2}.$$

Since the k-shifting automaton has uniform weights and $\mathsf{q}_{\mathsf{w}_2}$ is uniform on $\mathcal{C}_T$, we can write for any string $\mathbf{x}$ accepted by $\mathcal{C}_T$:

$$\begin{aligned}
\log\left[\frac{\mathsf{q}[\mathbf{x}]}{\mathsf{q}_{\mathsf{w}_2}[\mathbf{x}]}\right] &= \log\left[\frac{1}{\mathsf{q}_{\mathsf{w}_2}[\mathbf{x}]}\right] - \log(|\mathcal{C}_T|) \\
&= \log\left[\frac{1}{\mathsf{q}_{\mathsf{w}_2}[\mathbf{z} = \mathbf{x}|\mathbf{z} \in \mathcal{C}_T]\mathsf{q}_{\mathsf{w}_2}[\mathbf{z} \in \mathcal{C}_T] + \mathsf{q}_{\mathsf{w}_2}[\mathbf{z} = \mathbf{x}|\mathbf{z} \notin \mathcal{C}_T]\mathsf{q}_{\mathsf{w}_2}[\mathbf{z} \notin \mathcal{C}_T]}\right] - \log(|\mathcal{C}_T|) \\
&= \log\left[\frac{1}{\frac{1}{|\mathcal{C}_T|}\mathsf{q}_{\mathsf{w}_2}[\mathbf{z} \in \mathcal{C}_T] + \mathsf{q}_{\mathsf{w}_2}[\mathbf{z} = \mathbf{x}|\mathbf{z} \notin \mathcal{C}_T]\mathsf{q}_{\mathsf{w}_2}[\mathbf{z} \notin \mathcal{C}_T]}\right] - \log(|\mathcal{C}_T|) \\
&\leq \log\left[\frac{|\mathcal{C}_T|}{\mathsf{q}_{\mathsf{w}_2}[\mathbf{z} \in \mathcal{C}_T]}\right] - \log(|\mathcal{C}_T|) = \log\left[\frac{1}{\mathsf{q}_{\mathsf{w}_2}[\mathbf{z} \in \mathcal{C}_T]}\right].
\end{aligned}$$

The probability that a string $\mathbf{z}$ is accepted by $\mathcal{C}_T$ (under the distribution $\mathsf{q}_{\mathcal{A}_2}$) is equal to the probability that it admits exactly k shifts. Let $\xi_t = 1_{\{\mathbf{z} \text{ shifts from } t-1 \text{ to } t\}}$ be a random variable indicating whether there is a shift at the t-th symbol in sequence $\mathbf{z}$. This is a Bernoulli random variable bounded by 1 with mean $\frac{k}{T-1}$ and variance $\frac{k}{T-1}(1 - \frac{k}{T-1})$. Since each shift occurs with probability $\frac{k}{T-1}$, we can use Sanov's theorem to write the following bound:

$$\mathsf{q}_{\mathsf{w}_2}[\mathbf{z} \notin \mathcal{C}_T] = \mathsf{q}_{\mathsf{w}_2}\left[\left|\sum_{t=2}^{T} \xi_t - k\right| > \frac{1}{2}\right] \leq 2e^{-(T-1)u},$$

where $u = (T-1)\min\left\{D\left(\frac{k+\frac{1}{2}}{T-1}\middle\|\frac{k}{T-1}\right), D\left(\frac{k-\frac{1}{2}}{T-1}\middle\|\frac{k}{T-1}\right)\right\}$. We now give lower bounds on the relative entropy terms arguments of the minimum operator. For the first term, using the inequalities $\log(1+x) \geq \frac{x}{1+\frac{x}{2}}$ and $\log(1+x) < x$, we can write

$$\begin{aligned}
&- D\left(\frac{k+\frac{1}{2}}{T-1}\middle\|\frac{k}{T-1}\right)\\
&= \left(1+\frac{1}{2k}\right)\frac{k}{T-1}\log\frac{1}{1+\frac{1}{2k}} + \left(1-\frac{k}{T-1}-\frac{1}{2k}\frac{k}{T-1}\right)\log\left(1+\frac{\frac{1}{2k}\frac{k}{T-1}}{1-\frac{k}{T-1}-\frac{1}{2k}\frac{k}{T-1}}\right)\\
&\leq \left(1+\frac{1}{2k}\right)\frac{k}{T-1}\frac{-\frac{1}{2k}}{1+\frac{1}{4k}} + \left(1-\frac{k}{T-1}-\frac{1}{2k}\frac{k}{T-1}\right)\frac{\frac{1}{2k}\frac{k}{T-1}}{1-\frac{k}{T-1}-\frac{1}{2k}\frac{k}{T-1}}\\
&= \frac{1}{2k}\frac{k}{T-1}\left(1-\frac{1+\frac{1}{2k}}{1+\frac{1}{4k}}\right) = \frac{-\frac{1}{8k^2}\frac{k}{T-1}}{1+\frac{1}{4k}} = \frac{-\frac{1}{4k^2}\frac{k}{T-1}}{2+\frac{1}{4k}} \leq -\frac{1}{12k(T-1)}.
\end{aligned}$$

Similarly, we can write:

$$\begin{aligned}
&- D\left(\left(1-\frac{1}{2k}\right)\frac{k}{T-1}\middle\|\frac{k}{T-1}\right)\\
&= \left(1-\frac{1}{2k}\right)\frac{k}{T-1}\log\frac{1}{1-\frac{1}{2k}} + \left[1-\frac{k}{T-1}+\frac{1}{2k}\frac{k}{T-1}\right]\log\left[1-\frac{\frac{1}{2k}\frac{k}{T-1}}{1-\frac{k}{T-1}+\frac{1}{2k}\frac{k}{T-1}}\right]\\
&\leq \left[\frac{1}{2k}-\frac{1}{8k^2}\right]\frac{k}{T-1} + \left[1-\frac{k}{T-1}+\frac{1}{2k}\frac{k}{T-1}\right]\frac{-\frac{1}{2k}\frac{k}{T-1}}{1-\frac{k}{T-1}+\frac{1}{2k}\frac{k}{T-1}}\\
&= -\frac{\frac{1}{4k^2}\frac{k}{T-1}}{2} = -\frac{1}{8k(T-1)}.
\end{aligned}$$

Using these inequalities, we can further bound the approximation error in the regret bound by:

$$\log\left[\frac{1}{\mathsf{q}_{\mathsf{w}_2}[\mathbf{z}\in\mathcal{C}_T]}\right] \leq \log\left[\frac{1}{1-2e^{-\frac{1}{12k}}}\right] = -\log\left(1-2e^{-\frac{1}{12k}}\right),$$

which completes the proof. □

I Extension to sleeping experts

In this section, we present AWAKEAWM, a path-based weighted majority algorithm that generalizes the algorithms in [Freund et al., 1997] to arbitrary families of expert sequences. Like AWM, AWAKEAWM maintains a set of weights over all the paths in the input automaton. At each round t, the algorithm performs a weighted majority-type update. However, it normalizes the weights so that the total weight of the awake set remains unchanged. This prevents the algorithm from "overfitting" to experts that have been asleep for many rounds. The pseudocode of AWAKEAWM is provided as Algorithm 7, and it is accompanied by the following theoretical guarantee.

Theorem 6 (Regret Bound for AWAKEAWM)**.** *Let K denote the number of accepting paths of $\mathcal{C}_T = \mathcal{C} \cap \mathcal{S}_T$, and for each $t \in [T]$, let $A_t \subseteq \Sigma$ denote the set of experts that are awake at time t. Then for any distribution $\mathsf{u} \in \Delta_K$,* AWAKEAWM *admits the following unweighted regret guarantee:*

$$\sum_{t=1}^{T} \sum_{\mathbf{x} \in \mathcal{C}_T \cap A_t} \mathsf{u}[\mathbf{x}] \mathop{\mathbb{E}}_{a \sim \mathsf{p}_t^{A_t}}[l_t[a]] - \sum_{t=1}^{T} \sum_{\mathbf{x} \in \mathcal{C}_T \cap A_t} \mathsf{u}[\mathbf{x}] l_t[\mathbf{x}[t]]$$
$$\leq \frac{\eta}{8} \sum_{t=1}^{T} \mathsf{u}(A_t) + \frac{1}{\eta} \log(K).$$

Proof. As in the proof of Theorem 1, for every $t \in [T]$ and $\mathbf{x} \in \Sigma^T$, let $w_t[\mathbf{x}]$ denote the sequence weight defining q_t via normalization, $\mathsf{q}_t[\mathbf{x}] = \frac{w_t[\mathbf{x}]}{\sum_{\mathbf{x}} w_t[\mathbf{x}]}$. Moreover, let $\mathsf{q}_t^{A_t}$ be the distribution induced over sequences in with labels that awake at time t, so that for every sequence $\mathbf{x} \in \mathcal{C}_T$ with $\mathbf{x}[t] \in A_t$, $\mathsf{q}_t^{A_t}[\mathbf{x}] = \frac{\mathsf{q}_t[\mathbf{x}]}{\sum_{\mathbf{x} \in \mathcal{C}_T \colon \mathbf{x}[t] \in A_t} \mathsf{q}_t[\mathbf{x}]}$, and for every sequence $\mathbf{x} \in \mathcal{C}_T$ with $\mathbf{x}[t] \notin A_t$, $\mathsf{q}_t^{A_t}[\mathbf{x}] = 0$.

Notice that by design, if a sequence $\mathbf{x} \in \mathcal{C}_T$ has a label that isn't awake at time t, $x[t] \notin A_t$, then $\mathsf{q}_{t+1}[\mathbf{x}] = \mathsf{q}_t[\mathbf{x}]$, since we do not update that edge.

Moreover, by the normalization scheme, $\sum_{\mathbf{x} \in \mathcal{C}_T \colon \mathbf{x}[t] \notin A_t} \mathsf{q}_{t+1}[\mathbf{x}] = \sum_{\mathbf{x} \in \mathcal{C}_T \colon \mathbf{x}[t] \notin A_t} \mathsf{q}_t[x]$.

Now let $\mathsf{u} \in \Delta_K$. Then we can write

$$\begin{aligned}
&D(\mathsf{u} \| \mathsf{q}_t) - D(\mathsf{u} \| \mathsf{q}_{t+1}) \\
&= \sum_{\mathbf{x} \in \mathcal{C}_T} \mathsf{u}[\mathbf{x}] \log \frac{\mathsf{q}_{t+1}[\mathbf{x}]}{\mathsf{q}_t[\mathbf{x}]} \\
&= \sum_{x \in \mathcal{C}_T \colon \mathbf{x}[t] \in A_t} \mathsf{u}[\mathbf{x}] \log \frac{\mathsf{q}_{t+1}[\mathbf{x}]}{\mathsf{q}_t[\mathbf{x}]} \\
&= \sum_{x \in \mathcal{C}_T \cap A_t} \mathsf{u}[\mathbf{x}] \log \frac{\mathsf{q}_{t+1}^{A_t}[\mathbf{x}]}{\mathsf{q}_t^{A_t}[\mathbf{x}]} \\
&= \sum_{x \in \mathcal{C}_T \cap A_t} \mathsf{u}[\mathbf{x}] \log \frac{\mathsf{q}_t^{A_t}[\mathbf{x}] e^{-\eta l_t[\mathbf{x}[t]]}}{\mathsf{q}_t^{A_t}[\mathbf{x}] \sum_{\mathbf{y} \in \mathcal{C}_T \colon \mathbf{y}[t] \in A_t} \mathsf{q}_t^{A_t}[\mathbf{y}] e^{-\eta l_t[\mathbf{y}[t]]}} \\
&= \sum_{\mathbf{x} \in \mathcal{C}_T \cap A_t} \mathsf{u}[\mathbf{x}](-\eta l_t[\mathbf{x}[t]]) - \sum_{\mathbf{x} \in \mathcal{C}_T \colon \mathbf{x}[t] \in A_t} \mathsf{u}[\mathbf{x}] \log \left(\sum_{\mathbf{y} \in \mathcal{C}_T \colon \mathbf{y}[t] \in A_t} \mathsf{q}_t^{A_t}[\mathbf{y}] e^{-\eta l_t[\mathbf{y}[t]]} \right) \\
&\leq -\eta \sum_{\mathbf{x} \in \mathcal{C}_T \cap A_t} \mathsf{u}[\mathbf{x}] l_t[\mathbf{x}[t]] - \sum_{\mathbf{x} \in \mathcal{C}_T \colon \mathbf{x}[t] \in A_t} \mathsf{u}[\mathbf{x}] \left(\mathop{\mathbb{E}}_{\mathbf{y} \sim \mathsf{q}_t^{A_t}}[-\eta l_t[\mathbf{y}[t]]] + \frac{\eta^2}{8} \right) \\
&= -\eta \sum_{\mathbf{x} \in \mathcal{C}_T \cap A_t} \mathsf{u}[\mathbf{x}] l_t[\mathbf{x}[t]] + \eta \sum_{\mathbf{x} \in \mathcal{C}_T \colon \mathbf{x}[t] \in A_t} \mathsf{u}[\mathbf{x}] \mathop{\mathbb{E}}_{a \sim \mathsf{p}_t^{A_t}}[l_t[a]] - \mathsf{u}(A_t) \frac{\eta^2}{8}.
\end{aligned}$$

Thus, by rearranging terms and summing over t, it follows that

$$\sum_{t=1}^{T} \sum_{\mathbf{x} \in \mathcal{C}_T \cap A_t} \mathsf{u}[\mathbf{x}] \mathop{\mathbb{E}}_{a \in \mathsf{p}_t^{A_t}}[\eta l_t[a]] - \sum_{t=1}^{T} \sum_{\mathbf{x} \in \mathcal{C}_T \cap A_t} \mathsf{u}[\mathbf{x}] l_t[\mathbf{x}[t]] \leq \sum_{t=1}^{T} \mathsf{u}(A_t) \frac{\eta}{8} + D(\mathsf{u} \| \mathsf{q}_1),$$

Algorithm 7: AWAKEAUTOMATAWEIGHTEDMAJORITY(AwakeAWM).

```
Algorithm: AWAKEAWM($\mathcal{C}$, $\eta$)
$\mathcal{B} \leftarrow \mathcal{C} \cap \mathcal{S}_T$
$\mathcal{A} \leftarrow$ WEIGHT-PUSHING($\mathcal{B}^\eta$)
$\boldsymbol{\beta} \leftarrow$ BWDDIST($\mathcal{A}$)
$\boldsymbol{\alpha} \leftarrow 0$; $\boldsymbol{\alpha}[I_\mathcal{A}] \leftarrow 1$
for each $e \in E_\mathcal{A}^{0\to 1}$ do
    $\mathsf{p}_1[\text{lab}[e]] \leftarrow \text{weight}[e]$.
for $t \leftarrow 1$ to $T$ do
    RECEIVE($A_t$)
    for each $a \in A_t$ do
        $\mathsf{p}_t^A[a] \leftarrow \mathsf{p}_t[a]/\mathsf{p}_t(A_t)$
    $i_t \leftarrow$SAMPLE($\mathsf{p}_t^A$); PLAY($i_t$); RECEIVE($\mathbf{l}_t$)
    $Z \leftarrow 0$; $\mathbf{w} \leftarrow 0$; $Z^A \leftarrow 0$
    for each $e \in E_\mathcal{A}^{t\to t+1}$ do
        if $\text{lab}[e] \in A_t$ then
            $\text{weight}[e] \leftarrow \text{weight}[e]\, e^{-\eta l_t[\text{lab}[e]]}$
        $\mathbf{w}[\text{lab}[e]] \leftarrow \mathbf{w}[\text{lab}[e]] + \boldsymbol{\alpha}[\text{src}[e]]\, \text{weight}[e]\, \boldsymbol{\beta}[\text{dest}[e]]$
        $\boldsymbol{\alpha}[\text{dest}[e]] \leftarrow \boldsymbol{\alpha}[\text{dest}[e]] + \boldsymbol{\alpha}[\text{src}[e]]\, \text{weight}[e]$
        if $\text{lab}[e] \in A_t$ then
            $Z^A \leftarrow Z^A + \mathbf{w}[\text{lab}[e]]$
    $\mathsf{p}_{t+1} \leftarrow \mathbf{w}\frac{\mathsf{p}_t(A_t)}{Z^A}$
```

and since for the unweighted regret, $\mathsf{q}_1 = \frac{1}{K}$, $D(\mathsf{u}\|\mathsf{q}_1) \leq \log(K)$, which completes the proof. □

As with AWM, AWAKEAWM is an efficient algorithm with a total computational cost that is linear in the number of transitions of $\mathcal{A}$ (or equivalently, $\mathcal{C}_T$). Moreover, as in the non-sleeping expert setting, we can further improve the computational complexity by applying φ-conversion to arrive at a or n-gram approximation and then φ-conversion. All other improvements in the sleeping expert setting will similarly mirror those for the non-sleeping expert algorithms.