Online Non-Additive Path Learning under Full and Partial Information

Corinna Cortes CORINNA@GOOGLE.COM
Google Research, New York, NY

Vitaly Kuznetsov VITALYK@GOOGLE.COM
Google Research, New York, NY

Mehryar Mohri MOHRI@CIMS.NYU.EDU
Google Research and Courant Institute, New York, NY

Holakou Rahmanian[*] HOLAKOU@MICROSOFT.COM
Microsoft Corporation, Redmond, WA

Manfred K. Warmuth MANFRED@UCSC.EDU
Google Inc., Zürich and UC Santa Cruz, CA

Editor: TBD

Abstract

We study the problem of online path learning with non-additive gains, which is a central problem appearing in several applications, including ensemble structured prediction. We present new online algorithms for path learning with non-additive count-based gains for the three settings of full information, semi-bandit and full bandit with very favorable regret guarantees. A key component of our algorithms is the definition and computation of an intermediate context-dependent automaton that enables us to use existing algorithms designed for additive gains. We further apply our methods to the important application of ensemble structured prediction. Finally, beyond count-based gains, we give an efficient implementation of the EXP3 algorithm for the full bandit setting with an arbitrary (non-additive) gain.

Keywords: online learning, non-additive gains, finite-state automaton

1. Introduction

One of the core combinatorial online learning problems is that of learning a minimum loss path in a directed graph. Examples can be found in structured prediction problems such as machine translation, automatic speech recognition, optical character recognition and computer vision. In these problems, predictions (or predictors) can be decomposed into possibly overlapping substructures that may correspond to words, phonemes, characters, or image patches. They can be represented in a directed graph where each edge represents a different substructure.

The number of paths, which serve as *experts*, is typically exponential in the size of the graph. Extensive work has been done to design efficient algorithms when the loss is

[*]. Research done in part while the author was visiting Courant Institute and interning at Google Research, NYC as a Ph.D. student from UCSC.

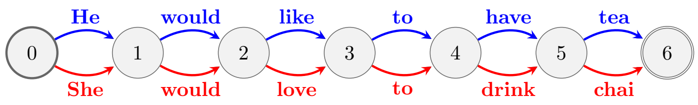

Figure 1: Combining outputs of two different translators (blue and red). There are 64 interleaved translations represented as paths. The BLEU score measures the overlap in n-grams between sequences. Here, an example of a 4-gram is “like-to-drink-tea”.

additive, that is when the loss of the path is the sum of the losses of the edges along that path. Several efficient algorithms with favorable guarantees have been designed both for the full information setting (Takimoto and Warmuth, 2003; Kalai and Vempala, 2005; Koolen et al., 2010) and different bandit settings (György et al., 2007; Cesa-Bianchi and Lugosi, 2012) by exploiting the additivity of the loss.

However, in modern machine learning applications such as machine translation, speech recognition and computational biology, the loss of each path is often not additive in the edges along the path. For instance, in machine translation, the BLEU score similarity determines the loss. The BLEU score can be closely approximated by the inner product of the count vectors of the n-gram occurrences in two sequences, where typically $n = 4$ (see Figure 1). In computational biology tasks, the losses are determined based on the inner product of the (discounted) count vectors of occurrences of n-grams with gaps (gappy n-grams). In other applications, such as speech recognition and optical character recognition, the loss is based on the edit-distance. Since the performance of the algorithms in these applications is measured via non-additive loss functions, it is natural to seek learning algorithms optimizing these losses directly. This motivates our study of online path learning for non-additive losses.

One of the applications of our algorithm is *ensemble structured prediction*. Online learning of ensembles of structured prediction experts can significantly improve the performance of algorithms in a number of areas including machine translation, speech recognition, other language processing areas, optical character recognition, and computer vision (Cortes et al., 2014). In general, ensemble structured prediction is motivated by the fact that one particular expert may be better at predicting one substructure while some other expert may be more accurate at predicting another substructure. Therefore, it is desirable to interleave the substructure predictions of all experts to obtain the more accurate prediction. This application becomes important, particularly in the bandit setting. Suppose one wishes to combine the outputs of different translators as in Figure 1. Instead of comparing oneself to the outputs of the best translator, the comparator is the best “interleaved translation” where each word in the translation can come from a different translator. However, computing the loss or the gain (such as BLEU score) of each path can be costly and may require the learner to resort to learning from partial feedback only.

Online path learning with non-additive losses has been previously studied by Cortes et al. (2015). That work focuses on the full information case providing an efficient implementations of Expanded Hedge (Takimoto and Warmuth, 2003) and Follow-the-Perturbed-Leader (Kalai and Vempala, 2005) algorithms under some technical assumptions on the outputs of the experts.

In this paper, we design algorithms for online path learning with non-additive gains or losses in the full information, as well as in several bandit settings specified in detail in Section 2. In the full information setting, we design an efficient algorithm that enjoys regret guarantees that are more favorable than those of Cortes et al. (2015), while not requiring any additional assumption. In the bandit settings, our algorithms, to the best of our knowledge, are the first efficient methods for learning with non-additive losses.

The key technical tools used in this work are weighted automata and transducers (Mohri, 2009). We transform the original path graph $\mathcal{A}$ (e.g. Figure 1) into an intermediate graph $\mathcal{A}'$. The paths in $\mathcal{A}$ are mapped to the paths in $\mathcal{A}'$, but now the losses in $\mathcal{A}'$ are additive along the paths. Remarkably, the size of $\mathcal{A}'$ does not depend on the size of the alphabet (word vocabulary in translation tasks) from which the output labels of edges are drawn. The construction of $\mathcal{A}'$ is highly non-trivial and is our primary contribution. This alternative graph $\mathcal{A}'$, in which the losses are additive, enables us to extend many well-known algorithms in the literature to the path learning problem.

The paper is organized as follows. We introduce the path learning setup in Section 2. In Section 3, we explore the wide family of non-additive count-based gains and introduce the alternative graph $\mathcal{A}'$ using automata and transducers tools. We present our algorithms in Section 4 for the full information, semi- and full bandit settings for the count-based gains. Next, we extend our results to *gappy* count-based gains in Section 5. The application of our method to the ensemble structured prediction is detailed in Appendix A. In Appendix B, we go beyond count-based gains and consider *arbitrary* (non-additive) gains. Even with no assumption about the structure of the gains, we can efficiently implement the EXP3 algorithm in the full bandit setting. Naturally, the regret bounds for this algorithm are weaker, however, since no special structure of the gains can be exploited in the absence of any assumption.

2. Basic Notation and Setup

We describe our path learning setup in terms of finite automata. Let $\mathcal{A}$ denote a fixed acyclic finite automaton. We call $\mathcal{A}$ the *expert automaton*. $\mathcal{A}$ admits a single *initial state* and one or several *final states* which are indicated by bold and double circles, respectively, see Figure 2(a). Each transition of $\mathcal{A}$ is labeled with a unique *name*. Denote the set of all transition names by E. An automaton with a single initial state is *deterministic* if no two outgoing transitions from a given state admit the same name. Thus, our automaton $\mathcal{A}$ is deterministic by construction since the transition names are unique. An *accepting path* is a sequence of transitions from the initial state to a final state. The expert automaton $\mathcal{A}$ can be viewed as an indicator function over strings in E^* such that $\mathcal{A}(\pi) = 1$ iff π is an accepting path. Each accepting path serves as an expert and we equivalently refer to it as a *path expert*. The set of all path experts is denoted by $\mathcal{P}$. At each round $t = 1, \ldots, T$, each transition $e \in E$ outputs a *symbol* from a finite non-empty alphabet Σ, denoted by $\text{out}_t(e) \in \Sigma$. The *prediction* of each path expert $\pi \in E^*$ at round t is the sequence of output symbols along its transitions at that round and is denoted by $\text{out}_t(\pi) \in \Sigma^*$. We also denote by $\text{out}_t(\mathcal{A})$ the automaton with the same topology as $\mathcal{A}$ where each transition e is labeled with $\text{out}_t(e)$, see Figure 2(b). At each round t, a *target sequence* $y_t \in \Sigma^*$ is presented to the learner. The *gain/loss* of each path expert π is $\mathcal{U}(\text{out}_t(\pi), y_t)$ where $\mathcal{U}\colon \Sigma^* \times \Sigma^* \longrightarrow \mathbb{R}_{\geq 0}$. Our focus is the $\mathcal{U}$ functions that are not necessarily additive along the transitions in $\mathcal{A}$. For

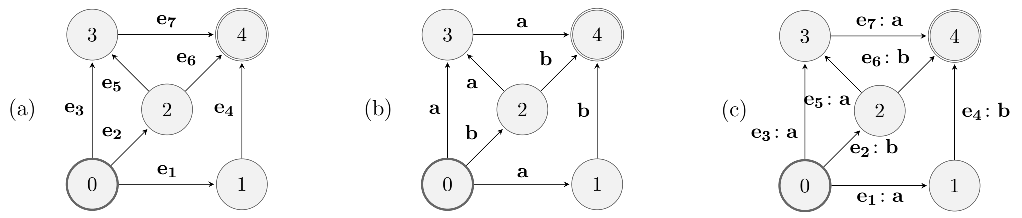

Figure 2: (a) The expert automaton denoted by $\mathcal{A}$ labeled with transition names. (b) The output of expert automaton at round t denoted by $\mathrm{out}_t(\mathcal{A})$ labeled with the outputs $\mathrm{out}_t(e)$ for each transition e. (c) The name and output of each transition together separated by a ':'.

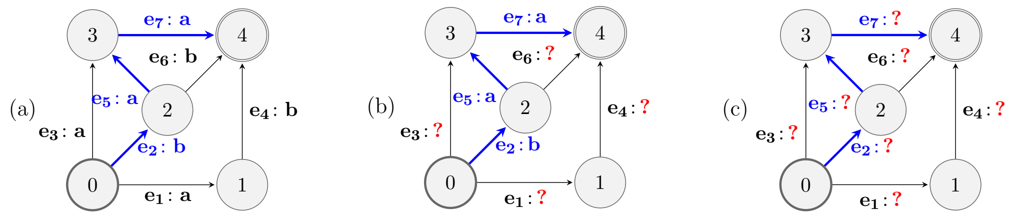

Figure 3: Information revealed in different settings: (a) full information (b) semi-bandit (c) full bandit. The name of each transition e and its output symbol (if revealed) are shown next to it separated by a ':'. The blue path indicates the path expert predicted by the learner at round t.

example, $\mathcal{U}$ can be either a *distance function* (e.g. edit-distance) or a *similarity function* (e.g. n-gram gain with $n \geq 2$).

We consider standard online learning scenarios of prediction with path experts. At each round $t \in [T]$, the *learner* picks a path expert π_t and predicts with its prediction $\mathrm{out}_t(\pi_t)$. The learner receives the gain of $\mathcal{U}(\mathrm{out}_t(\pi_t), y_t)$. Depending on the setting, the *adversary* may reveal some information about y_t and the output symbols of the transitions (see Figure 3). In the *full information* setting, y_t and $\mathrm{out}_t(e)$ are revealed to the learner for every transition e in $\mathcal{A}$. In the *semi-bandit* setting, the adversary reveals y_t and $\mathrm{out}_t(e)$ for every transition e along π_t. In *full bandit* setting, $\mathcal{U}(\mathrm{out}_t(\pi_t), y_t)$ is the only information that is revealed to the learner. The goal of the learner is to minimize the *regret* which is defined as the cumulative gain of the best path expert chosen in hindsight minus the cumulative expected gain of the learner.

3. Count-Based Gains

Many of the most commonly used non-additive gains in applications belong to the broad family of *count-based gains*, which are defined in terms of the number of occurrences of a fixed set of patterns, $\theta_1, \theta_2, \ldots, \theta_p$, in the sequence output by a path expert. These patterns may be n-grams, that is sequences of n consecutive symbols, as in a common approximation of the BLEU score in machine translation, a set of relevant subsequences of variable-length in computational biology, or patterns described by complex regular expressions in pronunciation modeling.

For any sequence $y \in \Sigma^*$, let $\Theta(y) \in \mathbb{R}^p$ denote the vector whose kth component is the number of occurrences of θ_k in y, $k \in [p]$.[1] The count-based gain function $\mathcal{U}$ at round t for a path expert π in $\mathcal{A}$ given the target sequence y_t is then defined as a dot product:

$$\mathcal{U}(\mathrm{out}_t(\pi), y_t) := \Theta(\mathrm{out}_t(\pi)) \cdot \Theta(y_t) \geq 0. \tag{1}$$

Such gains are not additive along the transitions and the standard online path learning algorithms for additive gains cannot be applied. Consider, for example, the special case of 4-gram-based gains in Figure 1. These gains cannot be expressed additively if the target sequence is, for instance, "He would like to eat cake" (see Appendix F). The challenge of learning with non-additive gains is even more apparent in the case of *gappy* count-based gains which allow for gaps of varying length in the patterns of interest. We defer the study of gappy-count based gains to Section 5.

How can we design algorithms for online path learning with such non-additive gains? Can we design algorithms with favorable regret guarantees for all three settings of full information, semi- and full bandit? The key idea behind our solution is to design a new automaton $\mathcal{A}'$ whose paths can be identified with those of $\mathcal{A}$ and, crucially, whose gains are additive. We will construct $\mathcal{A}'$ by defining a set of *context-dependent rewrite rules*, which can be compiled into a finite-state transducer $T_{\mathcal{A}}$ defined below. The *context-dependent automaton* $\mathcal{A}'$ can then be obtained by composition of the transducer $T_{\mathcal{A}}$ with $\mathcal{A}$. In addition to playing a key role in the design of our algorithms (Section 4), $\mathcal{A}'$ provides a compact representation of the gains since its size is substantially less than the dimension p (number of patterns).

3.1 Context-Dependent Rewrite Rules

We will use *context-dependent rewrite rules* to map $\mathcal{A}$ to the new representation $\mathcal{A}'$. These are rules that admit the following general form:

$$\phi \to \psi / \lambda \underline{\quad} \rho,$$

where ϕ, ψ, λ, and ρ are regular expressions over the alphabet of the rules. These rules must be interpreted as follows: ϕ is to be replaced by ψ whenever it is preceded by λ and followed by ρ. Thus, λ and ρ represent the left and right contexts of application of the rules. Several types of rules can be considered depending on their being obligatory or optional, and on their direction of application, from left to right, right to left or simultaneous application (Kaplan and Kay, 1994). We will be only considering rules with simultaneous applications. Such context-dependent rules can be efficiently compiled into a *finite-state transducer* (FST), under the technical condition that they do not rewrite their non-contextual part (Mohri and Sproat, 1996; Kaplan and Kay, 1994).[2] An FST $\mathcal{T}$ over an input alphabet Σ and output alphabet Σ' defines an indicator function over the pairs of strings in $\Sigma^* \times \Sigma'^*$. Given $x \in \Sigma^*$ and $y \in \Sigma'^*$, we have $\mathcal{T}(x, y) = 1$ if there exists a path from an initial state to a final state with input label x and output label y, and $\mathcal{T}(x, y) = 0$ otherwise.

1. This can be extended to the case of weighted occurrences where more emphasis is assigned to some patterns θ_k whose occurrences are then multiplied by a factor $\alpha_k > 1$, and less emphasis to others.
2. Additionally, the rules can be augmented with weights, which can help us cover the case of weighted count-based gains, in which case the result of the compilation is a weighted transducer (Mohri and Sproat, 1996). Our algorithms and theory can be extended to that case.

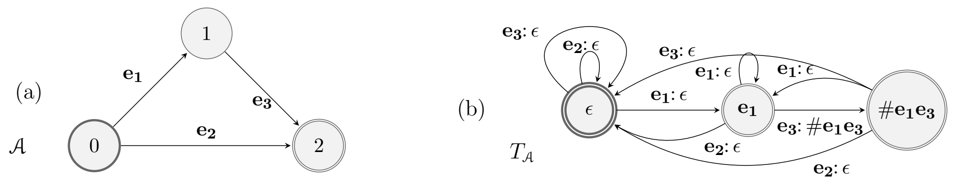

Figure 4: (a) An expert automaton $\mathcal{A}$; (b) associated context-dependent transducer $T_{\mathcal{A}}$ for bigrams. ϵ denotes the empty string. Inputs and outputs are written next to the transitions separated by a ':'.

To define our rules, we first introduce the alphabet E' as the set of transition names for the target automaton $\mathcal{A}'$. These capture all possible contexts of length r, where r is the length of pattern θ_k:

$$E' = \Big\{ \# e_1 \cdots e_r \mid e_1 \cdots e_r \text{ is a path segment of length } r \text{ in } \mathcal{A}, r \in \big\{ |\theta_1|, \ldots, |\theta_p| \big\} \Big\},$$

where the '#' symbol "glues" $e_1, \ldots, e_r \in E$ together and forms one single symbol in E'. We will have one context-dependent rule of the following form for each element $\# e_1 \cdots e_r \in E'$:

$$e_1 \cdots e_r \to \# e_1 \cdots e_r / \epsilon \underline{\quad\quad} \epsilon. \tag{2}$$

Thus, in our case, the left- and right-contexts are the empty strings[3], meaning that the rules can apply (simultaneously) at every position. In the special case where the patterns θ_k are the set of n-grams, then r is fixed and equal to n. Figure 4 shows the result of the rule compilation in that case for $n = 2$. This transducer inserts $\# e_1 e_3$ whenever e_1 and e_3 are found consecutively and otherwise outputs the empty string. We will denote the resulting FST by $T_{\mathcal{A}}$.

3.2 Context-Dependent Automaton $\mathcal{A}'$

To construct the context-dependent automaton $\mathcal{A}'$, we will use the *composition* operation. The composition of $\mathcal{A}$ and $T_{\mathcal{A}}$ is an FST denoted by $\mathcal{A} \circ T_{\mathcal{A}}$ and defined as the following product of two 0/1 outcomes for all inputs:

$$\forall x \in E^*,\ \forall y \in E'^*: \quad (\mathcal{A} \circ T_{\mathcal{A}})(x, y) := \mathcal{A}(x) \cdot T_{\mathcal{A}}(x, y).$$

There is an efficient algorithm for the composition of FSTs and automata (Pereira and Riley, 1997; Mohri et al., 1996; Mohri, 2009), whose worst-case complexity is in $O(|\mathcal{A}|\,|T_{\mathcal{A}}|)$. The automaton $\mathcal{A}'$ is obtained from the FST $(\mathcal{A} \circ T_{\mathcal{A}})$ by *projection*, that is by simply omitting the input label of each transition and keeping only the output label. Thus if we denote by Π the projection operator, then $\mathcal{A}'$ is defined as $\mathcal{A}' = \Pi(\mathcal{A} \circ T_{\mathcal{A}})$.

Observe that $\mathcal{A}'$ admits a fixed topology (states and transitions) at any round $t \in [T]$. It can be constructed in a pre-processing stage using the FST operations of composition and projection. Additional FST operations such as ϵ-removal and minimization can help further

3. Context-dependent rewrite rules are powerful tools for identifying different patterns using their left- and right-contexts. For our application of count-based gains, however, identifying these patterns are independent of their context and we do not need to fully exploit the strength of these rewrite rules.

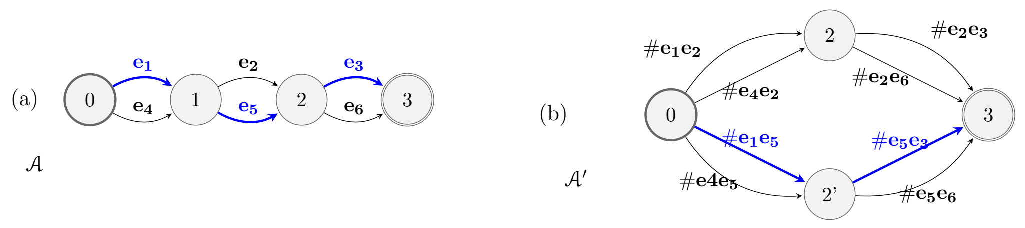

Figure 5: (a) An example of the expert automaton $\mathcal{A}$. (b) the associated context-dependent automaton $\mathcal{A}'$ with bigrams as patterns. The path $\pi = e_1 e_5 e_3$ in $\mathcal{A}$ and its corresponding path $\pi' = \#e_1 e_5 \#e_5 e_3$ in $\mathcal{A}'$ are marked in blue.

optimize the automaton obtained after projection (Mohri, 2009). Proposition 1, proven in Appendix D, ensures that for every accepting path π in $\mathcal{A}$, there is a unique corresponding accepting path in $\mathcal{A}'$. Figure 5 shows the automata $\mathcal{A}$ and $\mathcal{A}'$ in a simple case and how a path π in $\mathcal{A}$ is mapped to another path π' in $\mathcal{A}'$.

Proposition 1 *Let $\mathcal{A}$ be an expert automaton and let $T_{\mathcal{A}}$ be a deterministic transducer representing the rewrite rules (2). Then, for each accepting path π in $\mathcal{A}$, there exists a unique corresponding accepting path π' in $\mathcal{A}' = \Pi(\mathcal{A} \circ T_{\mathcal{A}})$.*

The size of the context-dependent automaton $\mathcal{A}'$ depends on the expert automaton $\mathcal{A}$ and the lengths of the patterns. Notice that, crucially, its size is independent of the size of the alphabet Σ. Appendix A analyzes more specifically the size of $\mathcal{A}'$ in the important application of ensemble structure prediction with n-gram gains.

At any round $t \in [T]$ and for any $\#e_1 \cdots e_r \in E'$, let $\text{out}_t(\#e_1 \cdots e_r)$ denote the sequence $\text{out}_t(e_1) \cdots \text{out}_t(e_r)$, that is the sequence obtained by concatenating the outputs of $e_1, \ldots, e_r$. Let $\text{out}_t(\mathcal{A}')$ be the automaton with the same topology as $\mathcal{A}'$ where each label $e' \in E'$ is replaced by $\text{out}_t(e')$. Once y_t is known, the representation $\Theta(y_t)$ can be found, and consequently, the additive contribution of each transition of $\mathcal{A}'$ can be computed. The following theorem, which is proved in Appendix D, shows the additivity of the gains in $\mathcal{A}'$. See Figure 6 for an example.

Theorem 2 *At any round $t \in [T]$, define the gain $g_{e',t}$ of the transition $e' \in E'$ in $\mathcal{A}'$ by $g_{e',t} := [\Theta(y_t)]_k$ if $out_t(e') = \theta_k$ for some $k \in [p]$ and $g_{e',t} := 0$ if no such k exists. Then, the gain of each path π in $\mathcal{A}$ at trial t can be expressed as an additive gain of the corresponding unique path π' in $\mathcal{A}'$:*

$$\forall t \in [T],\ \forall \pi \in \mathcal{P}: \quad \mathcal{U}(out_t(\pi), y_t) = \sum_{e' \in \pi'} g_{e',t}\ .$$

4. Algorithms

In this section, we present algorithms and associated regret guarantees for online path learning with non-additive count-based gains in the full information, semi-bandit and full bandit settings. The key component of our algorithms is the context-dependent automaton $\mathcal{A}'$. In what follows, we denote the length of the longest path in $\mathcal{A}'$ by K, an upper-bound on the gain of each transition in $\mathcal{A}'$ by B, the number of path experts by N, and the number of transitions and states in $\mathcal{A}'$ by M and Q, respectively. We note that K is at most the length of the longest path in $\mathcal{A}$ since each transition in $\mathcal{A}'$ admits a unique label.

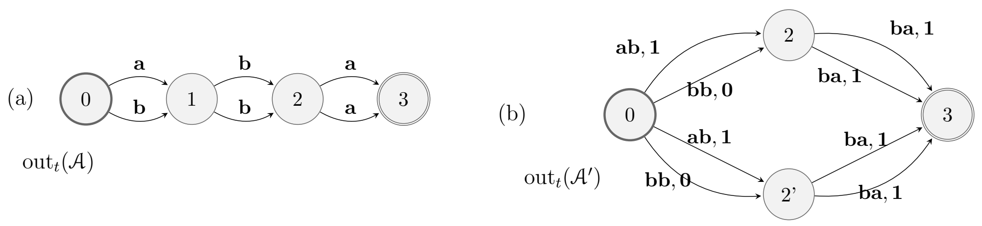

Figure 6: (a) the automaton $\mathrm{out}_t(\mathcal{A})$ of $\mathcal{A}$ in Figure 5(a), with bigram gains and $\Sigma = \{a, b\}$. (b) the automaton $\mathrm{out}_t(\mathcal{A}')$ given $y_t = aba$. Here, the patterns are $(\theta_1, \theta_2, \theta_3, \theta_4) = (aa, ab, ba, bb)$, and thus, $\Theta(y_t) = [0, 1, 1, 0]^T$. The additive gain contributed by each transition $e' \in E'$ in $\mathcal{A}'$ is written on it separated by a comma from $\mathrm{out}_t(e')$.

Remark. The number of accepting paths in $\mathcal{A}'$ is often equal to but sometimes less than the number of accepting paths in $\mathcal{A}$. In some degenerate cases, several paths $\pi_1, \ldots, \pi_k$ in $\mathcal{A}$ may correspond to one single path[4] π' in $\mathcal{A}'$. This implies that $\pi_1, \ldots, \pi_k$ in $\mathcal{A}$ will always consistently have the same gains in every round and that is the additive gain of π' in $\mathcal{A}'$. Thus, if π' is predicted by the algorithm in $\mathcal{A}'$, any of the paths $\pi_1, \ldots, \pi_k$ can be equivalently used for prediction in the original expert automaton $\mathcal{A}$.

4.1 Full Information: Context-dependent Component Hedge Algorithm

Koolen et al. (2010) gave an algorithm for online path learning with non-negative additive losses in the full information setting, the Component Hedge (CH) algorithm. For count-based losses, Cortes et al. (2015) provided an efficient Rational Randomized Weighted Majority (RRWM) algorithm. This algorithm requires the use of determinization (Mohri, 2009) which is only shown to have polynomial computational complexity under some additional technical assumptions on the outputs of the path experts. In this section, we present an extension of CH, the *Context-dependent Component Hedge* (CDCH), for the online path learning problem with non-additive count-based gains. CDCH admits more favorable regret guarantees than RRWM and can be efficiently implemented without any additional assumptions.

Our CDCH algorithm requires a modification of $\mathcal{A}'$ such that all paths admit an equal number K of transitions (same as the longest path). This modification can be done by adding at most $(K-2)(Q-2)+1$ states and zero-gain transitions (György et al., 2007). Abusing the notation, we will denote this new automaton by $\mathcal{A}'$ in this subsection. At each iteration t, CDCH maintains a weight vector $\boldsymbol{w}_t$ in the unit-flow polytope P over $\mathcal{A}'$, which is a set of vectors $\boldsymbol{w} \in \mathbb{R}^M$ satisfying the following conditions: (1) the weights of the outgoing transitions from the initial state sum up to one, and (2) for every non-final state, the sum of the weights of incoming and outgoing transitions are equal. For each $t \in \{1, \ldots, T\}$, we observe the gain of each transition $g_{t,e'}$, and define the loss of that transition as $\ell_{e'} = B - g_{t,e'}$. After observing the loss of each transition e' in $\mathcal{A}'$, CDCH updates each component of $\boldsymbol{w}$ as $\widehat{w}(e') \leftarrow w_t(e') \exp(-\eta\, \ell_{t,e'})$ (where η is a specified learning rate), and sets $\boldsymbol{w}_{t+1}$ to the relative entropy projection of the updated $\widehat{\boldsymbol{w}}$ back to the unit-flow polytope, i.e. $\boldsymbol{w}_{t+1} = \mathrm{argmin}_{\boldsymbol{w} \in P} \sum_{e' \in E'} w(e') \ln \frac{w(e')}{\widehat{w}(e')} + \widehat{w}(e') - w(e')$.

4. For example, in the case of n-gram gains, all the paths in $\mathcal{A}$ with a length less than n correspond to path with empty output in $\mathcal{A}'$ and will always have a gain of zero.

CDCH predicts by decomposing $\boldsymbol{w}_t$ into a convex combination of at most $|E'|$ paths in $\mathcal{A}'$ and then sampling a single path according to this mixture as described below. Recall that each path in $\mathcal{A}'$ identifies a path in $\mathcal{A}$ which can be recovered in time K. Therefore, the inference step of the CDCH algorithm takes at most time polynomial in $|E'|$ steps. To determine a decomposition, we find a path from the initial state to a final state with non-zero weights on all transitions, remove the largest weight on that path from each transition on that path and use it as a mixture weight for that path. The algorithm proceeds in this way until the outflow from initial state is zero. The following theorem from (Koolen et al., 2010) gives a regret guarantee for the CDCH algorithm.

Theorem 3 *With proper tuning of the learning rate η, the regret of CDCH is bounded as below:*

$$\forall\, \pi^* \in \mathcal{P}\colon \quad \sum_{t=1}^{n} \mathcal{U}(out_t(\pi^*), y_t) - \mathcal{U}(out_t(\pi_t), y_t) \leq \sqrt{2\,T\,B^2 K^2 \log(K\,M)} + B\,K\,\log(KM).$$

The regret bounds of Theorem 3 are in terms of the count-based gain $\mathcal{U}(\cdot,\cdot)$. Cortes et al. (2015) gave regret guarantees for the RRWM algorithm with count-based losses defined by $-\log \mathcal{U}(\cdot,\cdot)$. In Appendix E, we show that the regret associated with $-\log \mathcal{U}$ is upper-bounded by the regret bound associated with $\mathcal{U}$. Observe that, even with this approximation, the regret guarantees that we provide for CDCH are tighter by a factor of K. In addition, our algorithm does not require additional assumptions for an efficient implementation compared to the RRWM algorithm of Cortes et al. (2015).

4.2 Semi-Bandit: Context-dependent Semi-Bandit Algorithm

György et al. (2007) gave an efficient algorithm for online path learning with additive losses in the semi-bandit setting. In this section, we present a *Context-dependent Semi-Bandit* (CDSB) algorithm extending that work to solving the problem of online path learning with count-based gains in a semi-bandit setting. To the best of our knowledge, this is the first efficient algorithm with favorable regret bounds for this problem.

As with the algorithm of György et al. (2007), CDSB makes use of a set C of *covering paths* with the property that, for each $e' \in E'$, there is an accepting path π' in C such that e' belongs to π'. At each round t, CDSB keeps track of a distribution p_t over all N path experts by maintaining a weight $w_t(e')$ on each transition e' in $\mathcal{A}'$ such that the weights of outgoing transitions for each state sum up to 1 and $p_t(\pi') = \prod_{e' \in \pi'} w_t(e')$, for all accepting paths π' in $\mathcal{A}'$. Therefore, we can sample a path π' from p_t in at most K steps by selecting a random transition at each state according to the distribution defined by $\boldsymbol{w}_t$. To make a prediction, we sample a path in $\mathcal{A}'$ according to a mixture distribution $(1-\gamma)p_t + \gamma\mu$, where μ is a uniform distribution over paths in C. We select p_t with probability $1-\gamma$ or μ with probability γ and sample a random path π' from the randomly chosen distribution.

Once a path π'_t in $\mathcal{A}'$ is sampled, we observe the gain of each transition e' of π'_t, denoted by $g_{t,e'}$. CDSB sets $\widehat{w}_t(e') = w_t(e') \exp(\eta \widetilde{g}_{t,e'})$, where $\widetilde{g}_{t,e'} = (g_{t,e'} + \beta)/q_{t,e'}$ if $e' \in \pi'_t$ and $\widetilde{g}_{t,e'} = \beta/q_{t,e'}$ otherwise. Here, $\eta, \beta, \gamma > 0$ are parameters of the algorithm and $q_{t,e'}$ is the flow through e' in $\mathcal{A}'$, which can be computed using a standard shortest-distance algorithm over the probability semiring (Mohri, 2009). The updated distribution is $p_{t+1}(\pi') \propto \prod_{e' \in \pi'} \widehat{w}_t(e')$. Next, the *weight pushing* algorithm (Mohri, 1997) is applied (see Appendix C), which results in new transition weights $\boldsymbol{w}_{t+1}$ such that the total outflow out of each state is again one and the updated probabilities are $p_{t+1}(\pi') = \prod_{e' \in \pi'} w_{t+1}(e')$, thereby facilitating sampling. The computational complexity of each of the steps above is polynomial in the size of $\mathcal{A}'$. The following theorem from György et al. (2007) provides a regret guarantee for CDSB algorithm.

Theorem 4 *Let C denote the set of "covering paths" in $\mathcal{A}'$. For any $\delta \in (0,1)$, with proper tuning of the parameters η, β, and γ, the regret of the CDSB algorithm can be bounded as follows with probability $1-\delta$:*

$$\forall\, \pi^* \in \mathcal{P}\colon \quad \sum_{t=1}^{n} \mathcal{U}(out_t(\pi^*), y_t) - \mathcal{U}(out_t(\pi_t), y_t) \leq 2\, B\, \sqrt{TK}\Big(\sqrt{4K|C| \ln N} + \sqrt{M \ln \tfrac{M}{\delta}}\Big).$$

4.3 Full Bandit: Context-dependent ComBand Algorithm

Here, we present an algorithm for online path learning with count-based gains in the full bandit setting. Cesa-Bianchi and Lugosi (2012) gave an algorithm for online path learning with additive gains, ComBand. Our generalization, called *Context-dependent ComBand* (CDCB), is the first efficient algorithm with favorable regret guarantees for learning with count-based gains in this setting. For the full bandit setting with *arbitrary* gains, we develop an efficient execution of EXP3, called EXP3-AG, in Appendix B.

As with CDSB, CDCB maintains a distribution p_t over all N path experts using weights $\boldsymbol{w}_t$ on the transitions such that the outflow of each state is one and the probability of each path experts is the product of the weights of the transitions along that path. To make a prediction, we sample a path in $\mathcal{A}'$ according to a mixture distribution $q_t = (1-\gamma)p_t + \gamma\mu$, where μ is a uniform distribution over the paths in $\mathcal{A}'$. Note that this sampling can be efficiently implemented as follows. As a pre-processing step, define μ using a separate set of weights $\boldsymbol{w}^{(\mu)}$ over the transitions of $\mathcal{A}'$ in the same form. Set all the weights $\boldsymbol{w}^{(\mu)}$ to one and apply the weight-pushing algorithm to obtain a uniform distribution over the path experts. Next, we select p_t with probability $1-\gamma$ or μ with probability γ and sample a random path π' from the randomly chosen distribution.

After observing the scalar gain $g_{\pi'}$ of the chosen path, CDCB computes a surrogate gain vector for all transitions in $\mathcal{A}'$ via $\widetilde{\boldsymbol{g}}_t = g_{\pi'} P \boldsymbol{v}_{\pi'}$, where P is the pseudo-inverse of $\mathbb{E}[\boldsymbol{v}_{\pi'} \boldsymbol{v}_{\pi'}^T]$ and $\boldsymbol{v}_{\pi'} \in \{0,1\}^M$ is a bit representation of the path π'. As for CDSB, we set $\widehat{w}(e') = w_t(e') \exp(-\eta \widetilde{g}_{t,e'})$ and update $\mathcal{A}'$ via weighted-pushing to compute $\boldsymbol{w}_{t+1}$. We obtain the following regret guarantees from Cesa-Bianchi and Lugosi (2012) for CDCB:

Theorem 5 *Let λ_{min} denote the smallest non-zero eigenvalue of $\mathbb{E}[\boldsymbol{v}_{\pi'} \boldsymbol{v}_{\pi'}^T]$ where $\boldsymbol{v}_{\pi'} \in \{0,1\}^M$ is the bit representation of the path π' which is distributed according to the uniform distribution μ. With proper tuning of the parameters η and γ, the regret of CDCB can be bounded as follows:*

$$\forall\, \pi^* \in \mathcal{P}\colon \quad \sum_{t=1}^{n} \mathcal{U}(out_t(\pi^*), y_t) - \mathcal{U}(out_t(\pi_t), y_t) \leq 2\, B \sqrt{\left(\frac{2K}{M\lambda_{min}} + 1\right) TM \ln N}.$$

5. Extension to Gappy Count-Based Gains

Here, we generalize the results of Section 3 to a broader family of non-additive gains called *gappy count-based gains*: the gain of each path depends on the *discounted* counts of *gappy* occurrences of a fixed set of patterns $\theta_1, \ldots, \theta_p$ in the sequence output by that path. In a gappy occurrence, there can be "gaps" between symbols of the pattern. The count of a gappy occurrence is discounted multiplicatively by γ^k where $\gamma \in [0, 1]$ is a fixed *discount rate* and k is the total length of gaps. For example, the gappy occurrences of the pattern $\theta = aab$ in a sequence $y = babbaabaa$ with discount rate γ are

- $b\,a\,b\,b\,\underline{a\,a\,b}\,a\,a$, length of gap $= 0$, discount factor $= 1$;
- $b\,\underline{a}\,b\,b\,\underline{a}\,a\,\underline{b}\,a\,a$, length of gap $= 3$, discount factor $= \gamma^3$;
- $b\,\underline{a}\,b\,b\,a\,\underline{a\,b}\,a\,a$, length of gap $= 3$, discount factor $= \gamma^3$,

which makes the total discounted count of gappy occurrences of θ in y to be $1 + 2 \cdot \gamma^3$. Each sequence of symbols $y \in \Sigma^*$ can be represented as a discounted count vector $\Theta(y) \in \mathbb{R}^p$ of gappy occurrences of the patterns whose ith component is "the discounted number of gappy occurrences of θ_i in y". The gain function $\mathcal{U}$ is defined in the same way as in Equation (1).[5] A typical instance of such gains is gappy n-gram gains where the patterns are all $|\Sigma|^n$-many n-grams.

The key to extending our results in Section 3 to gappy n-grams is an appropriate definition of the alphabet E', the rewrite rules, and a new context-dependent automaton $\mathcal{A}'$. Once $\mathcal{A}'$ is constructed, the algorithms and regret guarantees presented in Section 4 can be extended to gappy count-based gains. To the best of our knowledge, this provides the first efficient online algorithms with favorable regret guarantees for gappy count-based gains in full information, semi-bandit and full bandit settings.

Context-Dependent Rewrite Rules. We extend the definition of E' so that it also encodes the total length k of the gaps: $E' = \Big\{(\#e_1 \cdots e_r)_k \mid e_1 \cdots e_r \in E,\ r \in \{|\theta_1|, \ldots, |\theta_p|\},\ k \in \mathbb{Z},\ k \geq 0\Big\}$. Note that the discount factor in gappy occurrences does not depend on the position of the gaps. Exploiting this fact, for each pattern of length n and total gap length k, we reduce the number of output symbols by a factor of $\binom{k+n-2}{k}$ by encoding the number of gaps as opposed to the position of the gaps.

We extend the rewrite rules in order to incorporate the gappy occurrences. Given $e' = (\#e_{i_1} e_{i_2} \ldots e_{i_n})_k$, for all path segments $e_{j_1} e_{j_2} \ldots e_{j_{n+k}}$ of length $n + k$ in $\mathcal{A}$ where $\{i_s\}_{s=1}^{n}$ is a subsequence of $\{j_r\}_{r=1}^{n+k}$ with $i_1 = j_1$ and $i_n = j_{n+k}$, we introduce the rule:

$$e_{j_1} e_{j_2} \ldots e_{j_{n+k}} \longrightarrow (\#e_{i_1} e_{i_2} \ldots e_{i_n})_k / \epsilon \_\_\_ \epsilon.$$

As with the non-gappy case in Section 3, the simultaneous application of all these rewrite rules can be efficiently compiled into a FST $T_{\mathcal{A}}$. The context-dependent transducer $T_{\mathcal{A}}$ maps any sequence of transition names in E into a sequence of corresponding gappy occurrences.

5. The regular count-based gain can be recovered by setting $\gamma = 0$.

The example below shows how $T_{\mathcal{A}}$ outputs the gappy trigrams given a path segment of length 5 as input:

$$\begin{aligned}e_1, e_2, e_3, e_4, e_5 \xrightarrow{T_{\mathcal{A}}} &(\#e_1e_2e_3)_0, (\#e_2e_3e_4)_0, (\#e_3e_4e_5)_0, (\#e_1e_2e_4)_1, (\#e_1e_3e_4)_1, (\#e_2e_3e_5)_1,\\ &(\#e_2e_4e_5)_1, (\#e_1e_2e_5)_2, (\#e_1e_4e_5)_2, (\#e_1e_3e_5)_2.\end{aligned}$$

Context-Dependent Automaton $\mathcal{A}'$. As in Section 3.2, we construct the context-dependent automaton as $\mathcal{A}' := \Pi(\mathcal{A} \circ T_{\mathcal{A}})$, which admits a fixed topology through trials. The rewrite rules are constructed in a way such that different paths in $\mathcal{A}$ are rewritten differently. Therefore, $T_{\mathcal{A}}$ assigns a unique output to a given path expert in $\mathcal{A}$. Proposition 1 ensures that for every accepting path π in $\mathcal{A}$, there is a unique corresponding accepting path in $\mathcal{A}'$.

For any round $t \in [T]$ and any $e' = (\#e_{i_1}e_{i_2}\cdots e_{i_n})_k$, define $\text{out}_t(e') := \text{out}_t(e_{i_1})\ldots\text{out}_t(e_{i_n})$. Let $\text{out}_t(\mathcal{A}')$ be the automaton with the same topology as $\mathcal{A}'$ where each label $e' \in E'$ is replaced by $\text{out}_t(e')$. Given y_t, the representation $\Theta(y_t)$ can be found, and consequently, the additive contribution of each transition of $\mathcal{A}'$. Again, we show the additivity of the gain in $\mathcal{A}'$ (see Appendix D for the proof).

Theorem 6 *Given the trial t and discount rate $\gamma \in [0,1]$, for each transition $e' \in E'$ in $\mathcal{A}'$, define the gain $g_{e',t} := \gamma^k\, [\Theta(y_t)]_i$ if $out_t(e') = (\theta_i)_k$ for some i and k and $g_{e',t} := 0$ if no such i and k exist. Then, the gain of each path π in $\mathcal{A}$ at trial t can be expressed as an additive gain of π' in $\mathcal{A}'$:*

$$\forall t \in [1,T],\ \forall \pi \in \mathcal{P}, \quad \mathcal{U}(out_t(\pi), y_t) = \sum_{e' \in \pi'} g_{e',t}\ .$$

We can extend the algorithms and regret guarantees presented in Section 4 to gappy count-based gains. To the best of our knowledge, this provides the first efficient online algorithms with favorable regret guarantees for gappy count-based gains in full information, semi-bandit and full bandit settings.

6. Conclusion and Open Problems

We presented several new algorithms for online non-additive path learning with very favorable regret guarantees for the full information, semi-bandit, and full bandit scenarios. We conclude with two open problems: (1) Non-acyclic expert automata: we assumed here that the expert automaton $\mathcal{A}$ is acyclic and the language of patterns $\mathcal{L} = \{\theta_1, \ldots, \theta_p\}$ is finite. Solving the non-additive path learning problem with cyclic expert automaton together with (infinite) regular language $\mathcal{L}$ of patterns remains an open problem; (2) Incremental construction of $\mathcal{A}'$: in this work, regardless of the data and the setting, the context-dependent automaton $\mathcal{A}'$ is constructed in advance as a pre-processing step. Is it possible to construct $\mathcal{A}'$ gradually as the learner goes through trials? Can we build $\mathcal{A}'$ incrementally in different settings and keep it as small as possible as the algorithm is exploring the set of paths and learning about the revealed data?

Acknowledgments

The work of MM was partly funded by NSF CCF-1535987 and NSF IIS-1618662. Part of this work was done while MKW was at UC Santa Cruz, supported by NSF grant IIS-1619271.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. OpenFst: a general and efficient weighted finite-state transducer library. In *Proceedings of CIAA*, pages 11–23. Springer, 2007.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77, 2002.

Nicolo Cesa-Bianchi and Gábor Lugosi. Combinatorial bandits. *Journal of Computer and System Sciences*, 78(5):1404–1422, 2012.

Corinna Cortes, Vitaly Kuznetsov, and Mehryar Mohri. Ensemble methods for structured prediction. In *Proceedings of ICML*, 2014.

Corinna Cortes, Vitaly Kuznetsov, Mehryar Mohri, and Manfred K. Warmuth. On-line learning algorithms for path experts with non-additive losses. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 424–447, 2015.

András György, Tamás Linder, Gábor Lugosi, and György Ottucsák. The on-line shortest path problem under partial monitoring. *Journal of Machine Learning Research*, 8(Oct): 2369–2403, 2007.

Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *Journal of Computer and System Sciences*, 71(3):291–307, 2005.

Ronald M. Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20(3):331–378, 1994.

Wouter M. Koolen, Manfred K. Warmuth, and Jyrki Kivinen. Hedging structured concepts. In *Proceedings of COLT*, pages 93–105, 2010.

Mehryar Mohri. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311, 1997.

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

Mehryar Mohri and Richard Sproat. An efficient compiler for weighted rewrite rules. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 231–238. Association for Computational Linguistics, 1996.

Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted automata in text and speech processing. In *Proceedings of ECAI-96 Workshop on Extended finite state models of language*, 1996.

Fernando Pereira and Michael Riley. Speech recognition by composition of weighted finite automata. In *Finite-State Language Processing*, pages 431–453. MIT Press, 1997.

Gilles Stoltz. *Information incomplete et regret interne en prédiction de suites individuelles*. PhD thesis, Ph. D. thesis, Univ. Paris Sud, 2005.

Eiji Takimoto and Manfred K. Warmuth. Path kernels and multiplicative updates. *JMLR*, 4:773–818, 2003.

Appendix A. Applications to Ensemble Structured Prediction

The algorithms discussed in Section 4 can be used for the online learning of ensembles of structured prediction experts, and consequently, significantly improve the performance of algorithms in a number of areas including machine translation, speech recognition, other language processing areas, optical character recognition, and computer vision. In structured prediction problems, the output associated with a model h is a structure y that can be decomposed and represented by ℓ substructures $y_1, \ldots, y_\ell$. For instance, h may be a machine translation system and y_i a particular word.

The problem of ensemble structured prediction can be described as follows. The learner has access to a set of r experts $h_1, \ldots, h_r$ to make an ensemble prediction. Therefore, at each round $t \in [1, T]$, the learner can use the outputs of the r experts $\text{out}_t(h_1), \ldots, \text{out}_t(h_r)$. As illustrated in Figure 7(a), each expert h_j consists of ℓ substructures $h_j = (h_{j,1}, \ldots, h_{j,\ell})$.

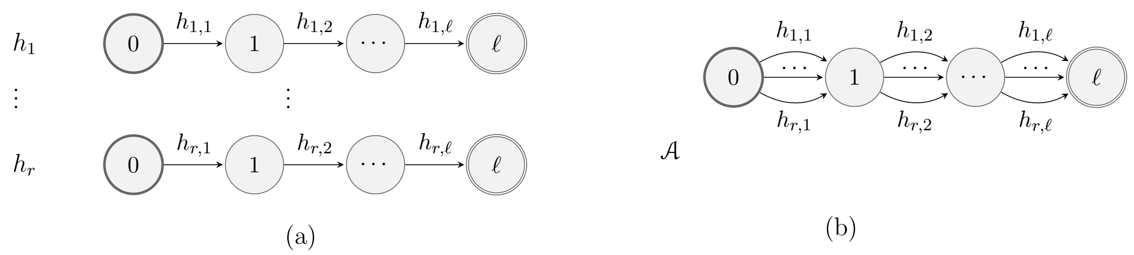

Figure 7: (a) the structured experts $h_1, \ldots, h_r$. (b) the expert automaton $\mathcal{A}$ allowing all combinations.

Represented by paths in an automaton, the substructures of these experts can be *combined together*. Allowing all combinations, Figure 7(b) illustrates the expert automaton $\mathcal{A}$ induced by r structured experts with ℓ substructures. The objective of the learner is to find the best path expert which is the combination of substructures with the best expected gain. This is motivated by the fact that one particular expert may be better at predicting one substructure while some other expert may be more accurate at predicting another substructure. Therefore, it is desirable to combine the substructure predictions of all experts to obtain the more accurate prediction.

Consider the online path learning problem with expert automaton $\mathcal{A}$ in Figure 7(b) with non-additive n-gram gains described in Section 3 for typical small values of n (e.g. $n = 4$). We construct the context-dependent automaton $\mathcal{A}'$ via a set of rewrite rules. The rewrite rules are as follows:

$$h_{j_1,i+1},\ h_{j_2,i+2},\ \ldots, h_{j_n,i+n} \rightarrow \#h_{j_1,i+1}h_{j_2,i+2}\ldots h_{j_n,i+n}\,/\,\epsilon \underline{\quad} \epsilon,$$

for all $j_1, \ldots, j_n \in [1, r]$, $i \in [0, \ell - n]$. The number of rewrite rules is $(\ell - n + 1)\, r^n$. We compile these rewrite rules into the context-dependent transducer $T_{\mathcal{A}}$, and then construct the context-dependent automaton $\mathcal{A}' = \Pi(\mathcal{A} \circ T_{\mathcal{A}})$.

The context-dependent automaton $\mathcal{A}'$ is illustrated in Figure 8. The transitions in $\mathcal{A}'$ are labeled with n-grams of transition names $h_{i,j}$ in $\mathcal{A}$. The context-dependent automaton $\mathcal{A}'$ has $\ell - n + 1$ layers of states each of which acts as a "memory" indicating the last observed

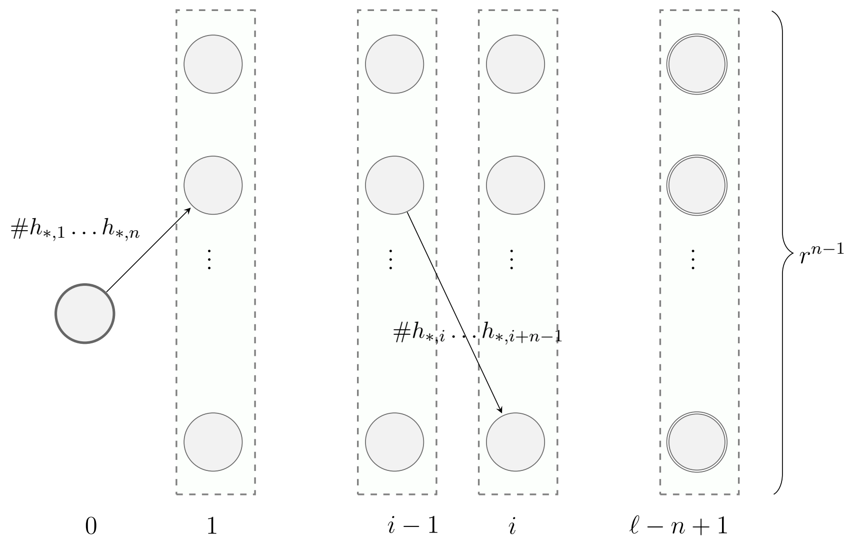

Figure 8: The context-dependent automaton $\mathcal{A}'$ for the expert automaton $\mathcal{A}$ depicted in Figure 7(b).

$(n-1)$-gram of transition names $h_{i,j}$. With each intermediate state (i.e. a state which is neither the initial state nor a final state), a $(n-1)$-gram is associated. Each layer contains r^{n-1} many states encoding all combinations of $(n-1)$-grams ending at that state. Each intermediate state has r incoming transitions which are the n-grams ending with $(n-1)$-gram associated with the state. Similarly each state has r outgoing transitions which are the n-grams starting with $(n-1)$-gram associated with the state.

The number of states and transitions in $\mathcal{A}'$ are $Q = 1 + r^n(\ell - n)$ and $M = r^n(\ell - n + 1)$, respectively. Note that the size of $\mathcal{A}'$ does not depend on the size of the output alphabet Σ. Also notice that all paths in $\mathcal{A}'$ have equal length of $K = \ell - n + 1$. Furthermore the number of paths in $\mathcal{A}'$ and $\mathcal{A}$ are the same and equal to $N = r^\ell$.

We now apply the algorithms introduced in Section 4.

A.1 Full Information: Context-dependent Component Hedge Algorithm

We apply the CDCH algorithm to this application in full information setting. The context-dependent automaton $\mathcal{A}'$ introduced in this section is highly structured. We can exploit this structure and obtain better bounds comparing to the general bounds of Theorem 3 for CDCH.

Theorem 7 *Let B denote an upper-bound for the gains of all the transitions in $\mathcal{A}'$, and T be the time horizon. The regret of CDCH algorithm on ensemble structured prediction with r predictors consisting of ℓ substructures with n-gram gains can be bounded as*

$$\mathit{Regret}_{CDCH} \leq \sqrt{2\,T\,B^2\,(\ell-n+1)^2\,n\,\log r} + B\,(\ell-n+1)\,n\,\log r.$$

Proof First, note that all paths in $\mathcal{A}'$ have equal length of $K = \ell - n + 1$. Therefore there is no need of modifying $\mathcal{A}'$ to make all paths of the same length. At each trial $t \in [T]$, we define the loss of each transition as $\ell_{t,e'} := B - g_{t,e'}$. Extending the results of Koolen et al. (2010), the general regret bound of CDCH is

$$\text{Regret}_{CDCH} \leq \sqrt{2\,T\,K\,B^2\,\Delta(\boldsymbol{v}_{\pi^*}||\boldsymbol{w}_1)} + B\,\Delta(\boldsymbol{v}_{\pi^*}||\boldsymbol{w}_1), \tag{3}$$

where $\boldsymbol{v}_{\pi^*} \in \{0,1\}^M$ is a bit vector representation of the best comparator π^*, $\boldsymbol{w}_1 \in [0,1]^M$ is the initial weight vector in the unit-flow polytope, and

$$\Delta(\boldsymbol{w}||\widehat{\boldsymbol{w}}) := \sum_{e' \in E'} \left(w_{e'} \ln \frac{w_{e'}}{\widehat{w}_{e'}} + \widehat{w}_{e'} - w_{e'} \right).$$

Since the initial state has r^n outgoing transitions, and all the intermediate states have r incoming and outgoing transitions, the initial vector $\boldsymbol{w}_1 = \frac{1}{r^n}\mathbf{1}$ falls into the unit-flow polytope, where $\mathbf{1}$ is a vector of all ones. Also $\boldsymbol{v}_{\pi^*}$ has exactly $K = \ell - n + 1$ many ones. Therefore:

$$\Delta(\boldsymbol{v}_{\pi^*}||\boldsymbol{w}_1) = (\ell - n + 1)\, n\, \log r \tag{4}$$

Combining the Equations (3) and (4) gives us the desired regret bound. ■

A.2 Semi-Bandit: Context-dependent Semi-Bandit Algorithm

In order to apply the algorithm CDSB in semi-bandit setting in this application, we need to introduce a set of "covering paths" C in $\mathcal{A}'$. We introduce C by partitioning all the transitions in $\mathcal{A}'$ into r^n paths of length $\ell - n + 1$ iteratively as follows. At each iteration, choose an arbitrary path π from the initial state to a final state. Add π to the set C and remove all its transitions from $\mathcal{A}'$. Notice that the number of incoming and outgoing transitions for each intermediate state are always equal throughout the iterations. Also note that in each iteration, the number of outgoing edges from the initial state decreases by one. Therefore after r^n iterations, C contains a set of r^n paths that partition the set of transitions in $\mathcal{A}'$.

Furthermore, observe that the number of paths in $\mathcal{A}'$ and $\mathcal{A}$ are the same and equal to $N = r^\ell$. The Corollary below is a direct result of Theorem 4 with $|C| = r^n$.

Corollary 8 *For any $\delta \in (0,1)$, with proper tuning, the regret of the CDSB algorithm can be bounded, with probability $1-\delta$, as:*

$$\textit{Regret}_{CDSB} \leq 2\,B\,(\ell - n + 1)\,\sqrt{T}\left(\sqrt{4\,r^n\,\ell\,\ln r} + \sqrt{r^n \ln \frac{r^n(\ell - n + 1)}{\delta}}\right).$$

A.3 Full Bandit: Context-dependent ComBand Algorithm

We apply the CDCB algorithm to this application in full bandit setting. The Corollary below, which is a direct result of Theorem 5, give regret guarantee for CDCB algorithm.

Corollary 9 *Let λ_{min} denote the smallest non-zero eigenvalue of $\mathbb{E}[\boldsymbol{v}_\pi \boldsymbol{v}_\pi^T]$ where $\boldsymbol{v}_\pi \in \{0,1\}^M$ is the bit representation of the path π which is distributed according to the uniform* *distribution μ. With proper tuning, the regret of CDCB can be bounded as follows:*

$$\mathit{Regret}_{CDCB} \leq 2\,B \sqrt{T\left(\frac{2(\ell-n+1)}{r^n(\ell-n+1)\lambda_{min}}+1\right) r^n\,(\ell-n+1)\,\ell\,\ln r}.$$

Appendix B. Path Learning for Full Bandit and Arbitrary Gain

So far, we have discussed the count-based gains as a wide family of non-additive gains in full information, semi- and full bandit settings. We developed learning algorithms that exploit the special structure of count-based gains. In this section, we go beyond the count-based gains in the full bandit setting and consider the scenario where the gain function is arbitrary and admits no known structure. In other words, any two paths can have completely independent gains regardless of the number of overlapping transitions they may share. Clearly, the CDCB algorithm of Section 4 cannot be applied to this case as it is specialized to (gappy) count-based gains. However, we present a general algorithm for path learning in the full bandit setting, when the gain function is arbitrary. This algorithm (called EXP3-AG) admits weaker regret bounds since no special structure of the gains can be exploited in the absence of any assumption. The algorithm is essentially an efficient implementation of EXP3 for path learning with arbitrary gains using weighted automata and graph operations.

We start with a brief description of the EXP3 algorithm of Auer et al. (2002), which is an online learning algorithm designed for the full bandit setting over a set of N experts. The algorithm maintains a distribution $\boldsymbol{w}_t$ over the set of experts, with $\boldsymbol{w}_1$ initialized to the uniform distribution. At each round $t \in [T]$, the algorithm samples an expert I_t according to $\boldsymbol{w}_t$ and receives (only) the gain g_{t,I_t} associated to that expert. It then updates the weights multiplicatively via the rule $w_{t+1,i} \propto w_{t,i} \exp(\eta\, \widetilde{g}_{t,i})$ for all $i \in [N]$, where $\widetilde{g}_{t,i} = \frac{g_{t,i}}{w_{t,i}} \mathbf{1}\{I_t = i\}$ is an unbiased surrogate gain associated with expert i. The weights $w_{t+1,i}$ are then normalized to sum to one.[6]

In our learning scenario, each expert is a path in $\mathcal{A}$. Since the number of paths is exponential in the size of $\mathcal{A}$, maintaining a weight per path is computationally intractable. We cannot exploit the properties of the gain function since it does not admit any known structure. However, we can make use of the graph representation of the experts. We will show that the weights of the experts at round t can be compactly represented by a deterministic *weighted finite automaton* (WFA) $\mathcal{W}_t$. We will further show that sampling a path from $\mathcal{W}_t$ and updating $\mathcal{W}_t$ can be done efficiently.

A deterministic WFA $\mathcal{W}$ is a deterministic finite automaton whose transitions and final states carry weights. Let $w(e)$ denote the weight of a transition e and $w_f(q)$ the weight at a final state q. The weight $\mathcal{W}(\pi)$ of a path π ending in a final state is defined as the product of its constituent transition weights and the weight at the final state: $\mathcal{W}(\pi) := \left(\prod_{e\in\pi} w(e)\right) \cdot w_f(\mathrm{dest}(\pi))$, where $\mathrm{dest}(\pi)$ denotes the destination state of π.

Sampling paths from a deterministic WFA $\mathcal{W}$ is straightforward when it is *stochastic*, that is when the weights of all outgoing transitions and the final state weight (if the state

6. The original EXP3 algorithm of Auer et al. (2002) mixes the weight vector with the uniform distribution in each trial. Later Stoltz (2005) showed that the mixing step is not necessary.

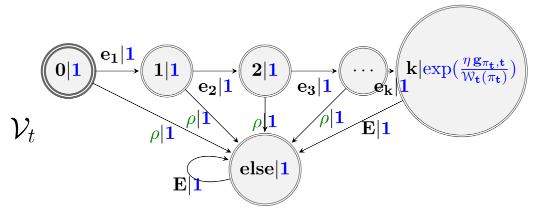

Figure 9: The update WFA $\mathcal{V}_t$. The weight of each state and transition is written next to its name separated by "|": $\xrightarrow{e|\text{weight}}$

```
1: $\mathcal{W}_1 \longleftarrow \mathcal{A}$
2: For $t = 1, \ldots, T$
3:     $\mathcal{W}_t \longleftarrow$ WeightPush($\mathcal{W}_t$)
4:     $\pi_t \longleftarrow$ Sample($\mathcal{W}_t$)
5:     $g_{\pi_t,t} \longleftarrow$ ReceiveGain($\pi_t$)
6:     $\mathcal{V}_t \longleftarrow$ UpdateWFA($\pi_t, \mathcal{W}_t(\pi_t), g_{t,\pi_t}$)
7:     $\mathcal{W}_{t+1} \longleftarrow \mathcal{W}_t \circ \mathcal{V}_t$
```

Figure 10: Algorithm EXP3-AG

is final) sum to one at every state: starting from the initial state, we can randomly draw a transition according to the probability distribution defined by the outgoing transition weights and proceed similarly from the destination state of that transition, until a final state is reached. The WFA we obtain after an update may not be stochastic, but we can efficiently compute an equivalent stochastic WFA $\mathcal{W}'$ from any $\mathcal{W}$ using the *weight-pushing* algorithm (Mohri, 1997, 2009; Takimoto and Warmuth, 2003): $\mathcal{W}'$ admits the same states and transitions as $\mathcal{W}$ and assigns the same weight to a path from the initial state to a final state; but the weights along paths are redistributed so that $\mathcal{W}'$ is stochastic. For an acyclic input WFA such as those we are considering, the computational complexity of weight-pushing is linear in the sum of the number of states and transitions of $\mathcal{W}$, see the Appendix C for details.

We now show how $\mathcal{W}_t$ can be efficiently updated using the standard WFA operation of *intersection* (or *composition*) with a WFA $\mathcal{V}_t$ representing the multiplicative weights that we will refer to as the *update WFA* at time t.

$\mathcal{V}_t$ is a deterministic WFA that assigns weight $\exp(\eta\, \widetilde{g}_{t,\pi})$ to path π. Thus, since $\widetilde{g}_{t,\pi} = 0$ for all paths but the path π_t sampled at time t, $\mathcal{V}_t$ assigns weight 1 to all paths $\pi \neq \pi_t$ and weight $\exp\left(\frac{\eta g_{t,\pi_t}}{\mathcal{W}_t(\pi_t)}\right)$ to π_t. $\mathcal{V}_t$ can be constructed deterministically as illustrated in Figure 9, using *ρ-transitions* (marked with ρ in green). A ρ-transition admits the semantics of the *rest*: it matches any symbol that is not labeling an existing outgoing transition at that state. For example, the ρ-transition at state 1 matches any symbol other than e_2. ρ-transitions lead to a more compact representation not requiring the knowledge of the full alphabet. This further helps speed up subsequent intersection operations (Allauzen et al., 2007).

To update the weights $\mathcal{W}_t$, we use the *intersection* (or *composition*) of WFAs. By definition, the intersection of $\mathcal{W}_t$ and $\mathcal{V}_t$ is a WFA denoted by $(\mathcal{W}_t \circ \mathcal{V}_t)$ that assigns to each path expert π the product of the weights assigned by $\mathcal{W}_t$ and $\mathcal{V}_t$:[7]

$$\forall \pi \in \mathcal{P}: \quad (\mathcal{W}_t \circ \mathcal{V}_t)(\pi) = \mathcal{W}_t(\pi) \cdot \mathcal{V}_t(\pi).$$

There exists a general an efficient algorithm for computing the intersection of two WFAs (Pereira and Riley, 1997; Mohri et al., 1996; Mohri, 2009): the states of the intersection

7. The terminology of intersection is motivated by the case where the weights are either 0 or 1, in which case the set of paths with non-zero weights in $\mathcal{W}_t \circ \mathcal{V}_t$ is the intersection of the sets of paths with with weight 1 in $\mathcal{W}_t$ and $\mathcal{V}_t$.

WFA are formed by pairs of a state of the first WFA and a state of the second WFA, and the transitions obtained by matching pairs of transitions from the original WFAs, with their weights multiplied, see Appendix C for more details. Since both $\mathcal{W}_t$ and $\mathcal{V}_t$ are deterministic, their intersection $(\mathcal{W}_t \circ \mathcal{V}_t)$ is also deterministic (Cortes et al., 2015).

The following lemma (proven in Appendix D) shows that the weight assigned by EXP3-AG to each path expert coincides with those defined by EXP3.

Lemma 10 *At each round $t \in [T]$ in* EXP3-AG*, the following properties hold for $\mathcal{W}_t$ and $\mathcal{V}_t$:*

$$\mathcal{W}_{t+1}(\pi) \propto \exp(\eta \sum_{s=1}^{t} \widetilde{g}_{s,\pi}), \quad \mathcal{V}_t(\pi) = \exp(\eta\, \widetilde{g}_{t,\pi}), \quad s.t.\ \widetilde{g}_{s,\pi} = (g_{s,\pi}/\mathcal{W}_s(\pi)) \cdot \mathbf{1}\{\pi = \pi_s\}.$$

Figure 10 gives the pseudocode of EXP3-AG. The time complexity of EXP3-AG is dominated by the cost of the intersection operation (line 7). The worst-case space and time complexity of the intersection of two deterministic WFA is linear in the size of the automaton the algorithm returns. Due to the specific structure of $\mathcal{V}_t$, the size of $\mathcal{W}_t \circ \mathcal{V}_t$ can be shown to be at most $O(|\mathcal{W}_t| + |\mathcal{V}_t|)$ where $|\mathcal{W}_t|$ is the sum of the number of states and transitions in $\mathcal{W}_t$. This is significantly better than the worst case size of the intersection in general (i.e. $O(|\mathcal{W}_t||\mathcal{V}_t|)$. Recall that $\mathcal{W}_{t+1}$ is deterministic. Thus, unlike the algorithms of Cortes et al. (2015), no further determinization is required. The following Lemma guarantees the efficiency of EXP3-AG algorithm. See Appendix D for the proof.

Lemma 11 *The time complexity of* EXP3-AG *at round t is in $O(|\mathcal{W}_t| + |\mathcal{V}_t|)$. Moreover, in the worst case, the growth of $|\mathcal{W}_t|$ over time is at most linear in K where K is the length of the longest path in $\mathcal{A}$.*

The following upper bound holds for the regret of EXP3-AG, as a direct consequence of existing guarantees for EXP3 (Auer et al., 2002).

Theorem 12 *Let $U > 0$ be an upper bound on all path gains: $g_{t,\pi} \leq U$ for all $t \in [T]$ and all path π. Then, the regret of* EXP3-AG *with N path experts is upper bounded by $U\sqrt{2\,T\,N \log N}$.*

The $\sqrt{N}$ dependency of the bound suggests that the guarantee will not be informative for large values of N. However, the following known lower bound shows that, in the absence of any assumption about the structure of the gains, this dependency cannot be improved in general (Auer et al., 2002).

Theorem 13 *Let $U > 0$ be an upper bound on all path gains: $g_{t,\pi} \leq U$ for all $t \in [T]$ and all path π. Then, For any number of path experts $N \geq 2$ there exists a distribution over the assignment of gains to path experts such that the regret of any algorithm is at least $\frac{1}{20} U \min\{\sqrt{T\,N}, T\}$.*

Appendix C. Weighted Finite Automata

In this section, we formally describe several WFA operations relevant to this paper, as well as their properties.

C.1 Intersection of WFAs

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

$$(\mathcal{A}_1 \circ \mathcal{A}_2)(\pi) = \mathcal{A}_1(\pi) \cdot \mathcal{A}_2(\pi).$$

There exists a standard efficient algorithm for computing the intersection WFA (Pereira and Riley, 1997; Mohri et al., 1996; Mohri, 2009). States $Q \subseteq Q_1 \times Q_2$ of $\mathcal{A}_1 \circ \mathcal{A}_2$ are identified with pairs of states Q_1 of $\mathcal{A}_1$ and Q_2 of $\mathcal{A}_2$, as are the set of initial and final states. Transitions are obtained by matching pairs of transitions from each WFA and multiplying their weights:

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

The worst-case space and time complexity of the intersection of two deterministic WFAs 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 that are intersected (i.e. $O(|\mathcal{A}_1||\mathcal{A}_2|)$. 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 $\mathcal{A}_1 \circ \mathcal{A}_2$ 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 $a \in \Sigma$.

C.2 Weight Pushing

Given a WFA $\mathcal{W}$, the weight pushing algorithm (Mohri, 1997, 2009) computes an equivalent stochastic WFA. The weight pushing algorithm is defined as follows. For any state q in $\mathcal{W}$, let $d[q]$ denote the sum of the weights of all paths from q to final states:

$$d[q] = \sum_{\pi \in \mathcal{P}(q)} \left(\prod_{e \in \pi} w(e) \right) \cdot w_f(\mathrm{dest}(\pi)),$$

where $\mathcal{P}(q)$ denotes the set of paths from q to final states in $\mathcal{W}$. The weights $d[q]$s can be computed be simultaneously for all qs using standard shortest-distance algorithms over the probability semiring (**?**). The weight pushing algorithm performs the following steps. For any transition $(q, q') \in E$ such that $d[q] \neq 0$, its weight is updated as below:

$$w(q, q') \leftarrow d[q]^{-1}\, w(q, q')\, d[q'].$$

For any final state q, the weight is updated as follows:

$$w_f(q) \leftarrow w_f(q)\, d[q]^{-1}.$$

The resulting WFA is guaranteed to preserve the path expert weights and to be stochastic (Mohri, 2009).

Appendix D. Proofs

Lemma 10 *At each round $t \in [T]$, the following properties hold for $\mathcal{W}_t$ and $\mathcal{V}_t$:*

$$\mathcal{W}_{t+1}(\pi) \propto \exp(\eta \sum_{s=1}^{t} \widetilde{g}_{s,\pi}), \quad \mathcal{V}_t(\pi) = \exp(\eta\, \widetilde{g}_{t,\pi}), \quad s.t.\ \widetilde{g}_{s,\pi} = \begin{cases} \frac{g_{s,\pi}}{\mathcal{W}_s(\pi)} & \pi = \pi_s \\ 0 & \textit{otherwise.} \end{cases}$$

Proof Consider $\mathcal{V}_t$ in Figure 9 and let $\pi_t = e_1 e_2 \dots e_k$ be the path chosen by the learner. Every state in $\mathcal{V}_t$ is a final state. Therefore, $\mathcal{V}_t$ accepts any sequence of transitions names. Moreover, since the weights of all transitions are 1, the weight of any accepting path is simply the weight of its final state. The construction of $\mathcal{V}_t$ ensures that the weight of every sequence of transition names is 1, except for $\pi_t = e_1 e_2 \dots e_k$. Thus, the property of $\mathcal{V}_t$ is achieved:

$$\mathcal{V}_t(\pi) = \begin{cases} \exp\left(\frac{\eta\, g_{t,\pi}}{\mathcal{W}_t(\pi)}\right) & \pi = \pi_t \\ 1 & \text{otherwise} \end{cases}$$

To proof of the result for $\mathcal{W}_{t+1}$ is by induction on t. Consider the base case of $t = 0$. $\mathcal{W}_1$ is initialized to the automaton $\mathcal{A}$ with all weights being one. Thus, the weights of all paths are equal to 1 before weight pushing (i.e. $\mathcal{W}_1(\pi) \propto 1$). The inductive step is as follows:

$$\begin{aligned} \mathcal{W}_{t+1}(\pi) &\propto \mathcal{W}_t(\pi) \cdot \mathcal{V}_t(\pi) && \text{(definition of composition)} \\ &= \exp(\eta \sum_{s=1}^{t-1} \widetilde{g}_{s,\pi}) \cdot \exp(\eta\, \widetilde{g}_{t,\pi}) && \text{(induction hypothesis)} \\ &= \exp(\eta \sum_{s=1}^{t} \widetilde{g}_{s,\pi}), \end{aligned}$$

which completes the proof. ■

Lemma 11 *The time complexity of* EXP3-AG *at round t is in $O(|\mathcal{W}_t| + |\mathcal{V}_t|)$. Moreover, in the worst case, the growth of $|\mathcal{W}_t|$ over time is at most linear in K where K is the length of the longest path in $\mathcal{A}$.*

Proof Figure 10 gives the pseudocode of EXP3-AG. The time complexity of the weight-pushing step is in $O(|\mathcal{W}_t|)$, where $|\mathcal{W}_t|$ is the sum of the number of states and transitions in $\mathcal{W}_t$. Lines 4 and 6 in Algorithm 10 take $O(|\mathcal{V}_t|)$ time. Finally, regarding line 7, the worst-case space and time complexity of the intersection of two deterministic WFA is linear in the size of the automaton the algorithm returns. However, the size of the intersection automaton $\mathcal{W}_t \circ \mathcal{V}_t$ is significantly smaller than the general worst case (i.e. $O(|\mathcal{W}_t||\mathcal{V}_t|)$) due to the state "else" with all in-coming ρ-transitions (see Figure 9). Since $\mathcal{W}_t$ is deterministic, in the construction of $\mathcal{W}_t \circ \mathcal{V}_t$, each state of $\mathcal{V}_t$ except from the "else" state is paired up only with one state of $\mathcal{W}_t$. For example, if the state is the one reached by $e_1 e_2 e_3$, then it is paired up with the single state of $\mathcal{W}_t$ reached when reading $e_1 e_2 e_3$ from the initial state. Thus $|\mathcal{W}_t \circ \mathcal{V}_t| \leq |\mathcal{W}_t| + |\mathcal{V}_t|$, and therefore, the intersection operation in line 7 takes $O(|\mathcal{W}_t| + |\mathcal{V}_t|)$ time, which also dominates the time complexity of EXP3-AG algorithm.

Additionally, observe that the size $|\mathcal{V}_t|$ is in $O(K)$ where K is the length of the longest path in $\mathcal{A}$. Since $|\mathcal{W}_{t+1}| = |\mathcal{W}_t \circ \mathcal{V}_t| \leq |\mathcal{W}_t| + |\mathcal{V}_t|$, in the worst case, the growth of $|\mathcal{W}_t|$ over time is at most linear in K. ■

Proposition 1 *Let $\mathcal{A}$ be an expert automaton and let $T_\mathcal{A}$ be a deterministic transducer representing the rewrite rules (2). Then, for each accepting path π in $\mathcal{A}$ there exists a unique corresponding accepting path π' in $\mathcal{A}' = \Pi(\mathcal{A} \circ T_\mathcal{A})$.*

Proof To establish the correspondence, we introduce $T_\mathcal{A}$ as a mapping from the accepting paths in $\mathcal{A}$ to the accepting paths in $\mathcal{A}'$. Since $T_\mathcal{A}$ is deterministic, for each accepting path π in $\mathcal{A}$ (i.e. $\mathcal{A}(\pi) = 1$), $T_\mathcal{A}$ assigns a unique output π', that is $T_\mathcal{A}(\pi, \pi') = 1$. We show that π' is an accepting path in $\mathcal{A}'$. Observe that

$$(\mathcal{A} \circ T_\mathcal{A})(\pi, \pi') = \mathcal{A}(\pi) \cdot T_\mathcal{A}(\pi, \pi') = 1 \times 1 = 1,$$

which implies that $\mathcal{A}'(\pi') = \Pi(\mathcal{A} \circ T_\mathcal{A})(\pi') = 1$. Thus for each accepting path π in $\mathcal{A}$ there is a unique accepting path π' in $\mathcal{A}'$. ■

Theorem 2 *Given the trial t, for each transition $e' \in E'$ in $\mathcal{A}'$ define the gain $g_{e',t} := [\Theta(y_t)]_i$ if $out_t(e') = \theta_i$ for some i and $g_{e',t} := 0$ if no such i exists. Then, the gain of each path π in $\mathcal{A}$ at trial t can be expressed as an additive gain of π' in $\mathcal{A}'$:*

$$\forall t \in [1, T],\ \forall \pi \in \mathcal{P}: \quad \mathcal{U}(out_t(\pi), y_t) = \sum_{e' \in \pi'} g_{e',t}\ .$$

Proof By definition, the ith component of $\Theta(\text{out}_t(\pi))$ is the number of occurrences of θ_i in $\text{out}_t(\pi)$. Also, by construction of the context-dependent automaton based on rewrite rules, π' contains all path segments of length $|\theta_i|$ of π in $\mathcal{A}$ as transition labels in $\mathcal{A}'$. Thus every occurrence of θ_i in $\text{out}_t(\pi)$ will appear as a transition label in $\text{out}_t(\pi')$. Therefore the number of occurrences of θ_i in $\text{out}_t(\pi)$ is

$$[\Theta(\text{out}_t(\pi))]_i = \sum_{e' \in \pi'} \mathbf{1}\{\text{out}_t(e') = \theta_i\}, \tag{5}$$

where $\mathbf{1}\{\cdot\}$ is the indicator function. Thus, we have that

$$\begin{aligned}
\mathcal{U}(\text{out}_t(\pi), y_t) &= \Theta(y_t) \cdot \Theta(\text{out}_t(\pi)) && \text{(definition of } \mathcal{U}\text{)}\\
&= \sum_i [\Theta(y_t)]_i \, [\Theta(\text{out}_t(\pi))]_i \\
&= \sum_i [\Theta(y_t)]_i \sum_{e' \in \pi'} \mathbf{1}\{\text{out}_t(e') = \theta_i\} && \text{(Equation (5))}\\
&= \sum_{e' \in \pi'} \underbrace{\sum_i [\Theta(y_t)]_i \, \mathbf{1}\{\text{out}_t(e') = \theta_i\}}_{= g_{e',t}},
\end{aligned}$$

which concludes the proof. ■

Theorem 14 *Given the trial t and discount rate $\gamma \in [0,1]$, for each transition $e' \in E'$ in $\mathcal{A}'$ define the gain $g_{e',t} := \gamma^k \left[\Theta(y_t)\right]_i$ if $out_t(e') = (\theta_i)_k$ for some i and k and $g_{e',t} := 0$ if no such i and k exist. Then, the gain of each path π in $\mathcal{A}$ at trial t can be expressed as an additive gain of π' in $\mathcal{A}'$:*

$$\forall t \in [1,T],\ \forall \pi \in \mathcal{P}\colon \quad \mathcal{U}(out_t(\pi), y_t) = \sum_{e' \in \pi'} g_{e',t}\ .$$

Proof By definition, the ith component of $\Theta(\text{out}_t(\pi))$ is the discounted count of gappy occurrences of θ_i in $\text{out}_t(\pi)$. Also, by construction of the context-dependent automaton based on rewrite rules, π' contains all gappy path segments of length $|\theta_i|$ of π in $\mathcal{A}$ as transition labels in $\mathcal{A}'$. Thus every gappy occurrence of θ_i with k gaps in $\text{out}_t(\pi)$ will appear as a transition label $(\theta_i)_k$ in $\text{out}_t(\pi')$. Therefore the discounted counts of gappy occurrences of θ_i in $\text{out}_t(\pi)$ is

$$\left[\Theta(\text{out}_t(\pi))\right]_i = \sum_{e' \in \pi'} \sum_k \gamma^k\, \mathbf{1}\{\text{out}_t(e') = (\theta_i)_k\}. \tag{6}$$

Therefore, the following holds:

$$\begin{aligned}
\mathcal{U}(\text{out}_t(\pi), y_t) &= \Theta(y_t) \cdot \Theta(\text{out}_t(\pi)) && \text{(definition of } \mathcal{U}\text{)}\\
&= \sum_i \left[\Theta(y_t)\right]_i \left[\Theta(\text{out}_t(\pi))\right]_i \\
&= \sum_i \left[\Theta(y_t)\right]_i \sum_{e' \in \pi'} \sum_k \gamma^k\, \mathbf{1}\{\text{out}_t(e') = (\theta_i)_k\} && \text{(Equation (6))}\\
&= \sum_{e' \in \pi'} \underbrace{\sum_i \sum_k \gamma^k\ \left[\Theta(y_t)\right]_i \mathbf{1}\{\text{out}_t(e') = (\theta_i)_k\}}_{=g_{e',t}},
\end{aligned}$$

and the proof is complete. ■

Appendix E. Gains $\mathcal{U}$ vs Losses $-\log(\mathcal{U})$

Let $\mathcal{U}$ be a non-negative gain function. Also let $\pi^* \in \mathcal{P}$ be the best comparator over the T rounds. The regret associated with $\mathcal{U}$ and $-\log \mathcal{U}$, which are respectively denoted by R_G and R_L, are defined as below:

$$\begin{aligned}
R_G &:= \sum_{t=1}^{T} \mathcal{U}(\text{out}_t(\pi^*), y_t) - \mathcal{U}(\text{out}_t(\pi_t), y_t),\\
R_L &:= \sum_{t=1}^{T} -\log(\mathcal{U}(\text{out}_t(\pi_t), y_t)) - (-\log(\mathcal{U}(\text{out}_t(\pi^*), y_t))).
\end{aligned}$$

Observe that if $\mathcal{U}(\text{out}_t(\pi_t), y_t) = 0$ for any t, then R_L is unbounded. Otherwise, let us assume that there exists a positive constant $\alpha > 0$ such that $\mathcal{U}(\text{out}_t(\pi_t), y_t) \geq \alpha$ for all $t \in$

$[T]$. Note, for count-based gains, we have $\alpha \geq 1$, since all components of the representation $\Theta(\cdot)$ are non-negative integers. Thus, the next proposition shows that for count-based gains we have $R_L \leq R_G$.

Proposition 15 *Let $\mathcal{U}$ be a non-negative gain function. Assume that there exists $\alpha > 0$ such that $\mathcal{U}(out_t(\pi_t), y_t) \geq \alpha$ for all $t \in [T]$. Then, the following inequality holds: $R_L \leq \frac{1}{\alpha} R_G$.*

Proof The following chain of inequalities hold:

$$\begin{aligned}
R_L &= \sum_{t=1}^{T} -\log(\mathcal{U}(\mathrm{out}_t(\pi_t), y_t)) - (-\log(\mathcal{U}(\mathrm{out}_t(\pi^*), y_t))) \\
&= \sum_{t=1}^{T} \log\left[\frac{\mathcal{U}(\mathrm{out}_t(\pi^*), y_t)}{\mathcal{U}(\mathrm{out}_t(\pi_t), y_t)}\right] \\
&= \sum_{t=1}^{T} \log\left[1 + \frac{\mathcal{U}(\mathrm{out}_t(\pi^*), y_t) - \mathcal{U}(\mathrm{out}_t(\pi_t), y_t)}{\mathcal{U}(\mathrm{out}_t(\pi_t), y_t)}\right] \\
&\leq \sum_{t=1}^{T} \frac{\mathcal{U}(\mathrm{out}_t(\pi^*), y_t) - \mathcal{U}(\mathrm{out}_t(\pi_t), y_t)}{\mathcal{U}(\mathrm{out}_t(\pi_t), y_t)} && \text{(since } \log(1+x) \leq x\text{)} \\
&\leq \frac{1}{\alpha} \sum_{t=1}^{T} \mathcal{U}(\mathrm{out}_t(\pi^*), y_t) - \mathcal{U}(\mathrm{out}_t(\pi_t), y_t) && \text{(since } \mathcal{U}(\mathrm{out}_t(\pi_t), y_t) \geq \alpha\text{)} \\
&\leq \frac{1}{\alpha} R_G,
\end{aligned}$$

which completes the proof. ■

Appendix F. Non-Additivity of the Count-Based Gains

Here we show that the count-based gains are not additive in general. Consider the special case of 4-gram-based gains in Figure 1. Suppose the count-based gain defined in Equation (1) can be expressed additively along the transitions (proof by contradiction). Let the target sequence y be as below:

$$y = \textbf{He would like to eat cake}$$

Then the only 4-gram in Figure 1 with positive gain is "**He-would-like-to**". Thus, if a path contains this 4-gram, it will have a gain of 1. Otherwise, its gain will be 0. Suppose that the transitions are labeled as depicted in Figure 11 and each transition $e \in E$ carries an additive gain of $g(e)$. Consider the following four paths:

$$\begin{aligned}
\pi_1 &= e_1 e_2 e_3 e_4 e_5 e_6 \\
\pi_2 &= e_1' e_2 e_3 e_4 e_5 e_6 \\
\pi_3 &= e_1 e_2 e_3' e_4 e_5 e_6 \\
\pi_4 &= e_1' e_2 e_3' e_4 e_5 e_6
\end{aligned}$$

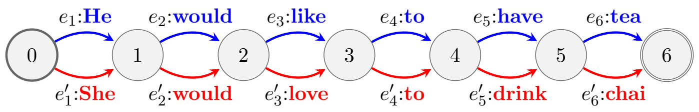

Figure 11: Additive gains carried by the transitions.

Due to the additivity of gains, we can obtain:

$$\begin{aligned}\mathcal{U}(\text{out}_t(\pi_1), y) + \mathcal{U}(\text{out}_t(\pi_4), y) &= g(e_1) + g(e'_1) + g(e_3) + g(e'_3) \\ &\quad + 2g(e_2) + 2g(e_4) + 2g(e_5) + 2g(e_6) \\ &= \mathcal{U}(\text{out}_t(\pi_2), y) + \mathcal{U}(\text{out}_t(\pi_3), y)\end{aligned}$$

This, however, contradicts the definition of the count-based gains in Equation (1):

$$\underbrace{\mathcal{U}(\text{out}_t(\pi_1), y)}_{=1} + \underbrace{\mathcal{U}(\text{out}_t(\pi_4), y)}_{=0} \neq \underbrace{\mathcal{U}(\text{out}_t(\pi_2), y)}_{=0} + \underbrace{\mathcal{U}(\text{out}_t(\pi_3), y)}_{=0}$$