Higher-order Derivatives of Weighted Finite-state Machines

Ran Zmigrod **Tim Vieira** **Ryan Cotterell**
University of Cambridge Johns Hopkins University ETH Zürich
rz279@cam.ac.uk tim.f.vieira@gmail.com
ryan.cotterell@inf.ethz.ch

Abstract

Weighted finite-state machines are a fundamental building block of NLP systems. They have withstood the test of time—from their early use in noisy channel models in the 1990s up to modern-day neurally parameterized conditional random fields. This work examines the computation of higher-order derivatives with respect to the normalization constant for weighted finite-state machines. We provide a general algorithm for evaluating derivatives of all orders, which has not been previously described in the literature. In the case of second-order derivatives, our scheme runs in the *optimal* $\mathcal{O}(A^2N^4)$ time where A is the alphabet size and N is the number of states. Our algorithm is significantly faster than prior algorithms. Additionally, our approach leads to a significantly faster algorithm for computing second-order expectations, such as covariance matrices and gradients of first-order expectations.

1 Introduction

Weighted finite-state machines (WFSMs) have a storied role in NLP. They are a useful formalism for speech recognition (Mohri et al., 2002), machine transliteration (Knight and Graehl, 1998), morphology (Geyken and Hanneforth, 2005; Lindén et al., 2009) and phonology (Cotterell et al., 2015) *inter alia*. Indeed, WFSMs have been "neuralized" (Rastogi et al., 2016; Hannun et al., 2020; Schwartz et al., 2018) and are still of practical use to the NLP modeler. Moreover, many popular sequence models, e.g., conditional random fields for part-of-speech tagging (Lafferty et al., 2001), are naturally viewed as special cases of WFSMs. For this reason, we consider the study of algorithms for the WFSMs of interest *in se* for the NLP community.

This paper considers inference algorithms for WSFMs. When WFSMs are acyclic, there exist simple linear-time dynamic programs, e.g., the forward algorithm (Rabiner, 1989), for inference. However, in general, WFSMs may contain cycles and such approaches are not applicable. Our work considers this general case and provides a method for efficient computation of m^{th}-order derivatives over a cyclic WFSM. To the best of our knowledge, no algorithm for higher-order derivatives has been presented in the literature beyond a general-purpose method from automatic differentiation. In contrast to many presentations of WFSMs (Mohri, 1997), our work provides a purely linear-algebraic take on them. And, indeed, it is this connection that allows us to develop our general algorithm.

We provide a thorough analysis of the soundness, runtime, and space complexity of our algorithm. In the special case of second-order derivatives, our algorithm runs *optimally* in $\mathcal{O}(A^2N^4)$ time and space where A is the size of the alphabet, and N is the number of states.[1] In contrast, the second-order expectation semiring of Li and Eisner (2009) provides an $\mathcal{O}(A^2N^7)$ solution and automatic differentiation (Griewank, 1989) yields a slightly faster $\mathcal{O}(AN^5{+}A^2N^4)$ solution. Additionally, we provide a speed-up for the general family of second-order expectations. Indeed, we believe our algorithm is the fastest known for computing common quantities, e.g., a covariance matrix.[2]

2 Weighted Finite-State Machines

In this section we briefly provide important notation for WFSMs and a classic result that efficiently finds the normalization constant for the probability distribution of a WFSM.

[1] Our implementation is available at https://github.com/rycolab/wfsm.

[2] Due to space constraints, we keep the discussion of our paper theoretical, though applications of expectations that we can compute are discussed in Li and Eisner (2009), Sánchez and Romero (2020), and Zmigrod et al. (2021).

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Short Papers), pages 240–248

Definition 1. *A **weighted finite-state machine** $\mathcal{M}$ is a tuple $\langle \boldsymbol{\alpha}, \{\mathbf{W}^{(a)}\}_{a\in\overline{\mathcal{A}}}, \boldsymbol{\omega}\rangle$ where $\mathcal{A}$ is an alphabet of size A, $\overline{\mathcal{A}} \overset{\text{def}}{=} \mathcal{A} \cup \{\varepsilon\}$, each $a \in \overline{\mathcal{A}}$ has a symbol-specific transition matrix $\mathbf{W}^{(a)} \in \mathbb{R}_{\geq 0}{}^{N\times N}$ where N is the number of states, and $\boldsymbol{\alpha}, \boldsymbol{\omega} \in \mathbb{R}_{\geq 0}{}^{N}$ are column vectors of start and end weights, respectively. We define the matrix $\mathbf{W} \overset{\text{def}}{=} \sum_{a\in\overline{\mathcal{A}}} \mathbf{W}^{(a)}$.*

Definition 2. *A **trajectory** $\tau_{i\rightsquigarrow\ell}$ is an ordered sequence of transitions from state i to state ℓ. Visually, we can represent a trajectory by*

$$\tau_{i\rightsquigarrow\ell} \overset{\text{def}}{=} i \xrightarrow{a} j \cdots k \xrightarrow{a'} \ell$$

*The **weight** of a trajectory is*

$$w(\tau_{i\rightsquigarrow\ell}) \overset{\text{def}}{=} \alpha_i \left(\prod_{(j \xrightarrow{a} k)\in\tau_{i\rightsquigarrow\ell}} \mathrm{W}_{jk}^{(a)} \right) \omega_\ell \tag{1}$$

We denote the (possibly infinite) set of trajectories from i to ℓ by $\mathcal{T}_{i\ell}$ and the set of all trajectories by $\mathcal{T} \overset{\text{def}}{=} \bigcup_{i,\ell\in[N]} \mathcal{T}_{i\ell}$.[3] *Consequently, when we say $\tau_{i\rightsquigarrow\ell} \in \mathcal{T}$, we make i and ℓ implicit arguments to which $\mathcal{T}_{i\ell}$ we are accessing.*

We define the **probability** of a trajectory $\tau_{i\rightsquigarrow\ell} \in \mathcal{T}$,

$$p(\tau_{i\rightsquigarrow\ell}) \overset{\text{def}}{=} \frac{w(\tau_{i\rightsquigarrow\ell})}{\mathrm{Z}} \tag{2}$$

where

$$\mathrm{Z} \overset{\text{def}}{=} \boldsymbol{\alpha}^\top \sum_{k=0}^{\infty} \mathbf{W}^k \, \boldsymbol{\omega} \tag{3}$$

Of course, p is only well-defined when $0 < \mathrm{Z} < \infty$.[4] Intuitively, $\boldsymbol{\alpha}^\top \mathbf{W}^k \boldsymbol{\omega}$ is the total weight of all trajectories of length k. Thus, Z is the total weight of all possible trajectories as it sums over the total weight for each possible trajectory length.

Theorem 1 (Corollary 4.2, Lehmann (1977))**.**

$$\mathbf{W}^\star \overset{\text{def}}{=} \sum_{k=0}^{\infty} \mathbf{W}^k = (\mathbf{I} - \mathbf{W})^{-1} \tag{4}$$

Thus, we can solve the infinite summation that defines $\mathbf{W}^\star$ by matrix inversion in $\mathcal{O}(N^3)$ time.[5]

[3] $|\mathcal{T}|$ is infinite if and only if $\mathcal{M}$ is cyclic.

[3] Another formulation for Z is $\sum_{\tau_{i\rightsquigarrow\ell}\in\mathcal{T}} w(\tau_{i\rightsquigarrow\ell})$.

[4] This requirement is equivalent to $\mathbf{W}$ having a spectral radius < 1.

[5] This solution technique may be extended to closed semirings (Kleene, 1956; Lehmann, 1977).

Corollary 1.

$$\mathrm{Z} = \boldsymbol{\alpha}^\top \mathbf{W}^\star \boldsymbol{\omega} \tag{5}$$

Proof. Follows from (4) in Theorem 1. ■

By Corollary 1, we can find Z in $\mathcal{O}(N^3 + AN^2)$.[6]

Strings versus Trajectories. Importantly, WFSMs can be regarded as weighted finite-state acceptors (WFSAs) which accept strings as their input. Each trajectory $\tau_{i\rightsquigarrow\ell} \in \mathcal{T}$ has a **yield** $\gamma(\tau_{i\rightsquigarrow\ell})$ which is the concatenation of the alphabet symbols of the trajectory. The yield of a trajectory ignores any ε symbols, a discussion regarding the semantics of ε is given in Hopcroft et al. (2001). As we focus on distributions over trajectories, we do not need special considerations for ε transitions. We do not consider distributions over yields in this work as such a distribution requires constructing a latent-variable model

$$p(\sigma) = \frac{1}{\mathrm{Z}} \sum_{\substack{\tau_{i\rightsquigarrow\ell}\in\mathcal{T},\\ \gamma(\tau_{i\rightsquigarrow\ell})=\sigma}} w(\tau_{i\rightsquigarrow\ell}) \tag{6}$$

where $\sigma \in \mathcal{A}^*$ and $\gamma(\tau_{i\rightsquigarrow\ell})$ is the yield of the trajectory. While marginal likelihood can be found efficiently,[7] many quantities, such as the entropy of the distribution over yields, are intractable to compute (Cortes et al., 2008).

3 Computing the Hessian (and Beyond)

In this section, we explore algorithms for efficiently computing the Hessian matrix $\nabla^2\mathrm{Z}$. We briefly describe two inefficient algorithms, which are derived by forward-mode and reverse-mode automatic differentiation. Next, we propose an efficient algorithm which is based on a key differential identity.

3.1 An $\mathcal{O}(A^2N^7)$ Algorithm with Forward-Mode Automatic Differentiation

One proposal for computing the Hessian comes from Li and Eisner (2009) who introduce a method based on semirings for computing a general family of quantities known as second-order expectations

[6] Throughout this paper, we assume a dense weight matrix and that matrix inversion is $\mathcal{O}(N^3)$ time. We note, however, that when the weight matrix is sparse and structured, faster matrix-inversion algorithms exist that exploit the strongly connected components decomposition of the graph (Mohri et al., 2000). We are agnostic to the specific inversion algorithm, but for simplicity we assume the aforementioned running time.

[7] This is done by intersecting the WFSA with another WFSA that only accepts σ.

(defined formally in §4). When applied to the computation of the Hessian their method reduces precisely to forward-mode automatic differentiation (AD; Griewank and Walther, 2008, Chap 3.1). This approach requires that we "lift" the computation of Z to operate over a richer numeric representation known as *dual numbers* (Clifford, 1871; Pearlmutter and Siskind, 2007). Unfortunately, the second-order dual numbers that we require to compute the Hessian introduce an overhead of $\mathcal{O}(A^2N^4)$ per numeric operation of the $\mathcal{O}(N^3)$ algorithm that computes Z, which results in $\mathcal{O}(A^2N^7)$ time.

3.2 An $\mathcal{O}(AN^5+A^2N^4)$ Algorithm with Reverse-Mode Automatic Differentiation

Another method for materializing the Hessian ∇^2Z is through reverse-mode automatic differentiation (AD). Recall that we can compute Z in $\mathcal{O}(N^3 + AN^2)$, and can consequently find ∇Z in $\mathcal{O}(N^3+AN^2)$ using one pass of reverse-mode AD (Griewank and Walther, 2008, Chapter 3.3). We can repeat differentiation to materialize ∇^2Z. Specifically, we run reverse-mode AD once for each element i of ∇Z. Taking the gradient of $(\nabla\mathrm{Z})_i$ gives a row of the Hessian matrix, $\nabla[(\nabla\mathrm{Z})_i] = [\nabla^2\mathrm{Z}]_{(i,:)}$. Since each of these passes takes time $\mathcal{O}(N^3+AN^2)$ (i.e., the same as the cost of Z), and ∇Z has size AN^2, the overall time is $\mathcal{O}(AN^5+A^2N^4)$.

3.3 Our Optimal $\mathcal{O}(A^2N^4)$ Algorithm

In this section, we will provide an $\mathcal{O}(A^2N^4)$-time and space algorithm for computing the Hessian. Since the Hessian has size $\mathcal{O}(A^2N^4)$, no algorithm can run faster than this bound; thus, our algorithm's time and space complexities are *optimal*. Our algorithm hinges on the following lemma, which shows that the each of partial derivatives of $\mathbf{W}^\star$ can be cheaply computed given $\mathbf{W}^\star$.

Lemma 1. *For $i, j, k, \ell \in [N]$ and $a \in \overline{\mathcal{A}}$*

$$\frac{\partial \mathrm{W}^\star_{i\ell}}{\partial \mathrm{W}^{(a)}_{jk}} = \mathrm{W}^\star_{ij}\dot{\mathrm{W}}^{(a)}_{jk}\mathrm{W}^\star_{k\ell} \tag{7}$$

where $\dot{\mathrm{W}}^{(a)}_{jk}$ is shorthand for $\partial \mathrm{W}^{(a)}_{jk}$.

Proof.

$$\begin{aligned}\frac{\partial \mathrm{W}^\star_{i\ell}}{\partial \mathrm{W}^{(a)}_{jk}} &= \frac{\partial}{\partial \mathrm{W}^{(a)}_{jk}}\left[(\mathbf{I}-\mathbf{W})^{-1}_{i\ell}\right] \\ &= -\mathrm{W}^\star_{ij}\frac{\partial}{\partial \mathrm{W}^{(a)}_{jk}}\left[(\mathbf{I}-\mathbf{W})\right]\mathrm{W}^\star_{k\ell} \\ &= \mathrm{W}^\star_{ij}\dot{\mathrm{W}}^{(a)}_{jk}\mathrm{W}^\star_{k\ell}\end{aligned}$$

The second step uses Equation 40 of the Matrix Cookbook (Petersen and Pedersen, 2008). ∎

We now extend Lemma 1 to express higher-order derivatives in terms of $\mathbf{W}^\star$. Note that as in Lemma 1, we will use $\dot{\mathrm{W}}^{(a)}_{ij}$ as a shorthand for the partial derivative $\partial \mathrm{W}^{(a)}_{ij}$.

Theorem 2. *For $m \geq 1$ and m-tuple of transitions $\vec{\tau} = \langle i_1 \xrightarrow{a_1} j_1, \dots, i_m \xrightarrow{a_m} j_m\rangle$*

$$\frac{\partial^m \mathrm{Z}}{\partial \mathrm{W}^{(a_1)}_{i_1 j_1}\cdots\partial \mathrm{W}^{(a_m)}_{i_m j_m}} = \sum_{\left\langle i'_1 \xrightarrow{a'_1} j'_1, \cdots, i'_m \xrightarrow{a'_m} j'_m\right\rangle \in \mathcal{S}_{\vec{\tau}}} \mathrm{s}_{i'_1}\dot{\mathrm{W}}^{(a'_1)}_{i'_1 j'_1}\mathrm{W}^\star_{j'_1 i'_2}\dot{\mathrm{W}}^{(a'_2)}_{i'_2 j'_2}\cdots\mathrm{W}^\star_{j'_{m-1} i'_m}\dot{\mathrm{W}}^{(a'_m)}_{i'_m j'_m}\mathrm{e}_{j'_m} \tag{8}$$

where $\mathbf{s} = \boldsymbol{\alpha}^\top\mathbf{W}^\star$, $\mathbf{e} = \mathbf{W}^\star\boldsymbol{\omega}$ and $\mathcal{S}_{\vec{\tau}}$ is the multi-set of permutations of $\vec{\tau}$.[8]

Proof. See App. A.1 ∎

Corollary 2. *For $i, j, k, l \in [N]$ and $a, b \in \overline{\mathcal{A}}$*

$$\frac{\partial^2 \mathrm{Z}}{\partial \mathrm{W}^{(a)}_{ij}\partial \mathrm{W}^{(b)}_{kl}} = \mathrm{s}_i\dot{\mathrm{W}}^{(a)}_{ij}\mathrm{W}^\star_{jk}\dot{\mathrm{W}}^{(b)}_{kl}\mathrm{e}_l + \mathrm{s}_k\dot{\mathrm{W}}^{(b)}_{kl}\mathrm{W}^\star_{li}\dot{\mathrm{W}}^{(a)}_{ij}\mathrm{e}_j \tag{9}$$

Proof. Application of Theorem 2 for the m=2 case. ∎

Theorem 2 shows that, if we have already computed $\mathbf{W}^\star$, each element of the m^{th} derivative can be found in $\mathcal{O}(m\,m!)$ time: We must sum over $\mathcal{O}(m!)$ permutations, where each summand is the product of $\mathcal{O}(m)$ items. Importantly, for the Hessian ($m = 2$), we can find each element in $\mathcal{O}(1)$ using Corollary 2. Algorithm $\mathtt{D}_m$ in Fig. 1 provides pseudocode for materializing the tensor containing the m^{th} derivatives of Z.

Theorem 3. *For $m \geq 1$, algorithm $\mathtt{D}_m$ computes ∇^mZ in $\mathcal{O}(N^3 + m\,m!\,A^mN^{2m})$ time and $\mathcal{O}(A^mN^{2m})$ space.*

Proof. Correctness of algorithm $\mathtt{D}_m$ follows from Theorem 2. The runtime and space bounds follow by needing to compute and store each combination of transitions. Each line of the algorithm is annotated with its running time. ∎

Corollary 3. *The Hessian ∇^2Z can be materialized in $\mathcal{O}(A^2N^4)$ time and $\mathcal{O}(A^2N^4)$ space. Note that these bounds are optimal.*

[8] As $\vec{\tau}$ may have duplicates, $\mathcal{S}_{\vec{\tau}}$ can also have duplicates and so must be a multi-set.

def $\mathtt{D}_m(\mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\omega})$:
    ▷ *Compute the tensor of m^{th}-order derivative of a WFSM; requires $\mathcal{O}(N^3 + m\, m!\, A^m N^{2m})$ time, $\mathcal{O}(A^m N^{2m})$ space.*
    $\mathbf{W}^\star \leftarrow (\mathbf{I} - \mathbf{W})^{-1}$ ▷ $\mathcal{O}(N^3)$
    $\mathbf{s} \leftarrow \boldsymbol{\alpha}^\top \mathbf{W}^\star$; $\mathbf{e} \leftarrow \mathbf{W}^\star \boldsymbol{\omega}$ ▷ $\mathcal{O}(N^2)$
    $\mathbf{D} \leftarrow \mathbf{0}$
    for $\vec{\tau} \in ([N] \times [N] \times \overline{\mathcal{A}})^m$: ▷ $\mathcal{O}(m m! A^m N^{2m})$
        for $\langle i_1 \xrightarrow{a_1} j_1, \dots, i_m \xrightarrow{a_m} j_m \rangle \in \mathcal{S}_{\vec{\tau}}$:
            $\mathrm{D}_{\vec{\tau}}$ += $\mathrm{s}_{i_1} \dot{\mathrm{W}}^{(a_1)}_{i_1 j_1} \mathrm{W}^\star_{j_1 i_2} \dot{\mathrm{W}}^{(a_2)}_{i_2 j_2} \mathrm{W}^\star_{j_2 i_3} \cdots \mathrm{W}^\star_{j_{m-1} i_m} \dot{\mathrm{W}}^{(a_m)}_{i_m j_m} \mathrm{e}_{j_m}$
    return $\mathbf{D}$
def $\mathtt{E}_2(\mathbf{W}, \boldsymbol{\alpha}, \boldsymbol{\omega}, r, t)$:
    ▷ *Compute the second-order expectation of a WFSM; requires $\mathcal{O}(N^3 + N^2(\overline{R}\,\overline{T} + AR'T'))$ time, $\mathcal{O}(N^2 + RT + N(R+T))$ space where $\overline{R} \overset{\text{def}}{=} \min(NR', R)$ and $\overline{T} \overset{\text{def}}{=} \min(NT', T)$.*
    Compute $\mathbf{W}^\star$, $\mathbf{s}$, and $\mathbf{e}$ as in $\mathtt{D}_m$ ▷ $\mathcal{O}(N^3)$
    $\mathrm{Z} \leftarrow \boldsymbol{\alpha}^\top \mathbf{W}^\star \boldsymbol{\omega}$
    $\widehat{r^s} \leftarrow \mathbf{0}$; $\widehat{r^e} \leftarrow \mathbf{0}$; $\widehat{t^s} \leftarrow \mathbf{0}$; $\widehat{t^e} \leftarrow \mathbf{0}$
    for $i, j \in [N], a \in \overline{\mathcal{A}}$: ▷ $\mathcal{O}(AN^2)$
        $\widehat{r^s_i}$ += $\mathrm{s}_j \dot{\mathrm{W}}^{(a)}_{ji} \mathrm{W}^{(a)}_{ji} r^{(a)}_{ji}$ ▷ $\mathcal{O}(R')$
        $\widehat{r^e_i}$ += $\dot{\mathrm{W}}^{(a)}_{ij} \mathrm{e}_j \dot{\mathrm{W}}^{(a)}_{ji} \mathrm{W}^{(a)}_{ij} r^{(a)}_{ij}$ ▷ $\mathcal{O}(R')$
        $\widehat{t^s_i}$ += $\mathrm{s}_j \dot{\mathrm{W}}^{(i)}_{aj} \mathrm{W}^{(a)}_{ji} t^{(a)}_{ji}$ ▷ $\mathcal{O}(T')$
        $\widehat{t^e_i}$ += $\dot{\mathrm{W}}^{(a)}_{ij} \mathrm{e}_j \mathrm{W}^{(a)}_{ij} t^{(a)}_{ij}$ ▷ $\mathcal{O}(T')$
    return $\frac{1}{\mathrm{Z}} \Big[\sum_{i,j=0}^N \widehat{r^s_i} \mathrm{W}^\star_{ij} \widehat{t^e_j}^\top + \big[\widehat{t^s_i} \mathrm{W}^\star_{ij} \widehat{r^e_j}^\top \big]^\top + \sum_{a \in \overline{\mathcal{A}}} \mathrm{s}_i \dot{\mathrm{W}}^{(a)}_{ij} \mathrm{e}_j \mathrm{W}^{(a)}_{ij} r^{(a)}_{ij} t^{(a)\top}_{ij} \Big]$ ▷ $\mathcal{O}(N^2(\overline{R}\,\overline{T} + AR'T'))$

Figure 1: Algorithms

Proof. Application of Theorem 3 for the m=2 case. ■

4 Second-Order Expectations

In this section, we leverage the algorithms of the previous section to efficiently compute a family expectations, known as a second-order expectations. To begin, we define an **additively decomposable** function $r\colon \mathcal{T} \mapsto \mathbb{R}^R$ as any function expressed as

$$r(\tau_{i \rightsquigarrow \ell}) = \sum_{(j \xrightarrow{a} k) \in \tau_{i \rightsquigarrow \ell}} r^{(a)}_{jk} \tag{10}$$

where each $r^{(a)}_{jk}$ is an R-dimensional vector. Since many r of interest are sparse, we analyze our algorithms in terms of R and its maximum density $R' \overset{\text{def}}{=} \max_{j \xrightarrow{a} k} \|r^{(a)}_{jk}\|_0$. Previous work has considered expectations of such functions (Eisner, 2001) and the *product* of two such functions (Li and Eisner, 2009), better known as second-order expectations. Formally, given two additively decomposable functions $r\colon \mathcal{T} \mapsto \mathbb{R}^R$ and $t\colon \mathcal{T} \mapsto \mathbb{R}^T$, a **second-order expectation** is

$$\mathbb{E}_{\tau_{i \rightsquigarrow \ell}} \left[r(\tau_{i \rightsquigarrow \ell}) t(\tau_{i \rightsquigarrow \ell})^\top \right] \overset{\text{def}}{=} \sum_{\tau_{i \rightsquigarrow \ell} \in \mathcal{T}} p(\tau_{i \rightsquigarrow \ell}) r(\tau_{i \rightsquigarrow \ell}) t(\tau_{i \rightsquigarrow \ell})^\top \tag{11}$$

Examples of second-order expectations include the Fisher information matrix and the gradients of first-order expectations (e.g., expected cost, entropy, and the Kullback–Leibler divergence).

Our algorithm is based on two fundamental concepts. Firstly, expectations for probability distributions as described in (1), can be decomposed as expectations over transitions (Zmigrod et al., 2021). Secondly, the marginal probabilities of transitions are connected to derivatives of Z.[9]

Lemma 2. *For $m \geq 1$ and m-tuple of transitions $\vec{\tau} = \langle i_1 \xrightarrow{a_1} j_1, \dots, i_m \xrightarrow{a_m} j_m \rangle$*

$$p(\vec{\tau}) = \frac{1}{\mathrm{Z}} \sum_{n=1}^{m} \frac{\partial^n \mathrm{Z}}{\partial \mathrm{W}^{(a_1)}_{i_1 j_1} \dots \partial \mathrm{W}^{(a_n)}_{i_n j_n}} \prod_{k=1}^{n} \mathrm{W}^{(a_k)}_{i_k j_k} \tag{12}$$

Proof. See App. A.2. ■

We formalize our algorithm as $\mathtt{E}_2$ in Fig. 1. Note that we achieve an additional speed-up by exploiting associativity (see App. A.3).

Theorem 4. *Algorithm $\mathtt{E}_2$ computes the second-order expectation of additively decomposable functions $r\colon \mathcal{T} \mapsto \mathbb{R}^R$ and $t\colon \mathcal{T} \mapsto \mathbb{R}^T$ in:*

$$\mathcal{O}(N^3 + N^2(\overline{R}\,\overline{T} + AR'T')) \text{ time}$$
$$\mathcal{O}(N^2 + RT + N(R+T)) \text{ space}$$

where $\overline{R} = \min(NR', R)$ and $\overline{T} = \min(NT', T)$.

Proof. Correctness of algorithm $\mathtt{E}_2$ is given in App. A.3. The runtime bounds are annotated on each line of the algorithm. We note that each $\widehat{r}$ and $\widehat{t}$ is $\overline{R}$ and $\overline{T}$ sparse. $\mathcal{O}(N^2)$ space is required to store $\mathbf{W}^\star$, $\mathcal{O}(RT)$ is required to store the expectation, and $\mathcal{O}(N(R+T))$ space is required to store the various $\widehat{r}$ and $\widehat{t}$ quantities. ■

Previous approaches for computing second-order expectations are significantly slower than $\mathtt{E}_2$. Specifically, using Li and Eisner (2009)'s second-order expectation semiring requires augmenting the

[9]This is commonly used in the case of single transition marginals, which can be found by $\nabla \log \mathrm{Z}$

arc weights to be $R \times T$ matrices and so runs in $\mathcal{O}(N^3RT + AN^2RT)$. Alternatively, we can use AD, as in §3.2, to materialize the Hessian and compute the pairwise transition marginals. This would result in a total runtime of $\mathcal{O}(AN^5 + A^2N^4R'T')$.

5 Conclusion

We have presented efficient methods that exploit properties of the derivative of a matrix inverse to find m-order derivatives for WFSMs. Additionally, we provided an explicit, novel, algorithm for materializing the Hessian in its *optimal* complexity, $\mathcal{O}(A^2N^4)$. We also showed how this could be utilized to efficiently compute second-order expectations of distributions under WFSMs, such as covariance matrices and the gradient of entropy. We hope that our paper encourages future research to use the Hessian and second-order expectations of WFSM systems, which have previously been disadvantaged by inefficient algorithms.

Acknowledgments

We would like to thank the reviewers for engagine with our work and providing valuable feedback. The first author is supported by the University of Cambridge School of Technology Vice-Chancellor's Scholarship as well as by the University of Cambridge Department of Computer Science and Technology's EPSRC.

Ethical Concerns

We do not foresee how the more efficient algorithms presented this work exacerbate any existing ethical concerns with NLP systems.

References

W. K. Clifford. 1871. Preliminary sketch of biquaternions. *Proceedings of the London Mathematical Society*, 1.

Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. 2008. On the computation of the relative entropy of probabilistic automata. *International Journal of Foundations of Computer Science*, 19.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015. Modeling word forms using latent underlying morphs and phonology. *Transactions of the Association for Computational Linguistics*, 3.

Jason Eisner. 2001. Expectation semirings: Flexible EM for learning finite-state transducers. In *Proceedings of the European Summer School in Logic, Language and Information Workshop on Finite-state Methods in Natural Language Processing*.

Alexander Geyken and Thomas Hanneforth. 2005. TAGH: A complete morphology for German based on weighted finite state automata. In *Finite-State Methods and Natural Language Processing, 5th International Workshop*.

Andreas Griewank. 1989. On automatic differentiation. *Mathematical Programming: Recent Developments and Applications*, 6.

Andreas Griewank and Andrea Walther. 2008. *Evaluating Derivatives–Principles and Techniques of Algorithmic Differentiation*, 2nd edition. Society for Industrial and Applied Mathematics.

Awni Hannun, Vineel Pratap, Jacob Kahn, and Wei-Ning Hsu. 2020. Differentiable weighted finite-state transducers. *CoRR*, abs/2010.01003.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2001. *Introduction to automata theory, languages, and computation, 2nd Edition*. Addison-Wesley series in computer science. Addison-Wesley-Longman.

Stephen C. Kleene. 1956. Representation of events in nerve nets and finite automata. *Automata Studies*.

Kevin Knight and Jonathan Graehl. 1998. Machine transliteration. *Computational Linguistics*, 24.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Daniel J. Lehmann. 1977. Algebraic structures for transitive closure. *Theoretical Computer Science*, 4.

Zhifei Li and Jason Eisner. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Krister Lindén, Miikka Silfverberg, and Tommi A. Pirinen. 2009. HFST tools for morphology - an efficient open-source package for construction of morphological analyzers. In *Proceedings of the State of the Art in Computational Morphology - Workshop on Systems and Frameworks for Computational Morphology*.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16.

Mehryar Mohri, Fernando C. N. Pereira, and Michael Riley. 2000. The design principles of a weighted finite-state transducer library. *Theoretical Computer Science*, 231.

Barak A. Pearlmutter and Jeffrey Mark Siskind. 2007. Lazy multivariate higher-order forward-mode AD. In *Proceedings of the 34th Association for Computer Machinery Special Interest Group on Programming Languages and Special Interest Group on Algorithms and Computation Theory Symposium on Principles of Programming Languages*.

K. B. Petersen and M. S. Pedersen. 2008. The matrix cookbook. Version 20081110.

Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the Institute of Electrical and Electronics Engineers*, 77.

Pushpendre Rastogi, Ryan Cotterell, and Jason Eisner. 2016. Weighting finite-state transductions with neural context. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Joan-Andreu Sánchez and Verónica Romero. 2020. Computation of moments for probabilistic finite-state automata. *Information Sciences*, 516.

Roy Schwartz, Sam Thomson, and Noah A. Smith. 2018. Bridging CNNs, RNNs, and weighted finite-state machines. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, volume 1.

Ran Zmigrod, Tim Vieira, and Ryan Cotterell. 2021. Efficient computation of expectations under spanning tree distributions. *Transactions of the Association for Computational Linguistics*.

A Proofs

A.1

Theorem 2. *For $m \geq 1$ and m-tuple of transitions $\vec{\tau} = \langle i_1 \xrightarrow{a_1} j_1, \ldots, i_m \xrightarrow{a_m} j_m \rangle$*

$$\frac{\partial^m \mathrm{Z}}{\partial \mathrm{W}^{(a_1)}_{i_1 j_1} \ldots \partial \mathrm{W}^{(a_m)}_{i_m j_m}} = \sum_{\left\langle i'_1 \xrightarrow{a'_1} j'_1, \ldots, i'_m \xrightarrow{a'_m} j'_m \right\rangle \in \mathcal{S}_{\vec{\tau}}} \mathrm{s}_{i'_1} \dot{\mathrm{W}}^{(a'_1)}_{i'_1 j'_1} \mathrm{W}^{\star}_{j'_1 i'_2} \dot{\mathrm{W}}^{(a'_2)}_{i'_2 j'_2} \cdots \mathrm{W}^{\star}_{j'_{m-1} i'_m} \dot{\mathrm{W}}^{(a'_m)}_{i'_m j'_m} \mathrm{e}_{j'_m}$$

where $\mathbf{s} = \boldsymbol{\alpha}^\top \mathbf{W}^\star$, $\mathbf{e} = \mathbf{W}^\star \boldsymbol{\omega}$ and $\mathcal{S}_{\vec{\tau}}$ is the multi-set of permutations of $\vec{\tau}$.

Proof. We prove this by induction on m.
Base Case: $m = 1$ and $\vec{\tau} = \langle i \xrightarrow{a} j \rangle$:

$$\frac{\partial \mathrm{Z}}{\partial \mathrm{W}^{(a)}_{ij}} = \frac{\partial}{\partial \mathrm{W}^{(a)}_{ij}} \left[\sum_{k,l=0}^{N} \alpha_k \mathrm{W}^\star_{kl} \omega_l \right] = \sum_{k,l=0}^{N} \alpha_k \mathrm{W}^\star_{ki} \dot{\mathrm{W}}^{(a)}_{ij} \mathrm{W}^\star_{jl} \omega_l = \mathrm{s}_i \dot{\mathrm{W}}^{(a)}_{ij} \mathrm{e}_j$$

Inductive Step: Assume that the expression holds for m. Let $\vec{\tau} = \langle i_1 \xrightarrow{a_1} j_1, \ldots, i_m \xrightarrow{a_m} j_m \rangle$ and consider the tuple $\vec{\tau}\,'$, the concatenation of $(i \xrightarrow{a} j)$ and $\vec{\tau}$.

$$\frac{\partial^{m+1} \mathrm{Z}}{\mathrm{W}^{(a)}_{ij} \partial \mathrm{W}^{(a_1)}_{i_1 j_1} \ldots \partial \mathrm{W}^{(a_m)}_{i_m j_m}} = \frac{\partial}{\partial \mathrm{W}^{(a)}_{ij}} \sum_{\left\langle i'_1 \xrightarrow{a'_1} j'_1, \ldots, i'_m \xrightarrow{a'_m} j'_m \right\rangle \in \mathcal{S}_{\vec{\tau}}} \mathrm{s}_{i'_1} \dot{\mathrm{W}}^{(a'_1)}_{i'_1 j'_1} \mathrm{W}^{\star}_{j'_1 i'_2} \cdots \dot{\mathrm{W}}^{(a'_m)}_{i'_m j'_m} \mathrm{e}_{j'_m}$$

Consider the derivative of each summand with respect to $\mathrm{W}^{(a)}_{ij}$. By the product rule, we have

$$\begin{aligned}
&\frac{\partial}{\partial \mathrm{W}^{(a)}_{ij}} \left[\mathrm{s}_{i'_1} \dot{\mathrm{W}}^{(a'_1)}_{i'_1 j'_1} \mathrm{W}^{\star}_{j'_1 i'_2} \cdots \dot{\mathrm{W}}^{(a'_m)}_{i'_m j'_m} \mathrm{e}_{j'_m} \right] \\
&= \mathrm{s}_i \dot{\mathrm{W}}^{(a)}_{ij} \mathrm{W}^{\star}_{j i'_1} \dot{\mathrm{W}}^{(a'_1)}_{i'_1 j'_1} \mathrm{W}^{\star}_{j'_1 i'_2} \cdots \dot{\mathrm{W}}^{(a'_m)}_{i'_m j'_m} \mathrm{e}_{j'_m} + \\
&\quad \cdots + \mathrm{s}_{i'_1} \cdots \mathrm{W}^{\star}_{j_k i} \dot{\mathrm{W}}^{(a)}_{ij} \mathrm{W}^{\star}_{j i_{k+1}} \cdots \mathrm{e}_{j'_m} + \\
&\quad \cdots + \mathrm{s}_{i'_1} \dot{\mathrm{W}}^{(a'_1)}_{i'_1 j'_1} \mathrm{W}^{\star}_{j'_1 i'_2} \cdots \dot{\mathrm{W}}^{(a'_m)}_{i'_m j'_m} \mathrm{W}^{\star}_{j'_m i} \dot{\mathrm{W}}^{(a)}_{ij} \mathrm{e}_j
\end{aligned}$$

The above expression is equal to inserting $i \xrightarrow{a} j$ in every spot of the induction hypothesis's permutation, thereby creating a permutation over $\vec{\tau}\,'$. Reassembling with the expression for the derivative,

$$\frac{\partial^{m+1} \mathrm{Z}}{\partial \mathrm{W}^{(a)}_{ij} \partial \mathrm{W}^{(a_1)}_{i_1 j_1} \ldots \partial \mathrm{W}^{(a_m)}_{i_m j_m}} = \sum_{\left\langle i'_1 \xrightarrow{a'_1} j'_1, \ldots, i'_{m+1} \xrightarrow{a'_{m+1}} j'_{m+1} \right\rangle) \in \mathcal{S}_{\vec{\tau}\,'}} \mathrm{s}_{i'_1} \dot{\mathrm{W}}^{(a'_1)}_{i'_1 j'_1} \mathrm{W}^{\star}_{j'_1 i'_2} \dot{\mathrm{W}}^{(a'_2)}_{i'_2 j'_2} \cdots \dot{\mathrm{W}}^{(a'_{m+1})}_{i'_{m+1} j'_{m+1}} \mathrm{e}_{j'_{m+1}}$$

∎

A.2

Lemma 2. *For $m \geq 1$ and m-tuple of transitions $\vec{\tau} = \langle i_1 \xrightarrow{a_1} j_1, \ldots, i_m \xrightarrow{a_m} j_m \rangle$*

$$p(\vec{\tau}) = \frac{1}{\mathrm{Z}} \sum_{n=1}^{m} \frac{\partial^n \mathrm{Z}}{\partial \mathrm{W}^{(a_1)}_{i_1 j_1} \ldots \partial \mathrm{W}^{(a_n)}_{i_n j_n}} \prod_{k=1}^{n} \mathrm{W}^{(a_k)}_{i_k j_k} \tag{10}$$

Proof. Let $\mathcal{T}_{\vec{\tau}}$ be the set of trajectories such that $\tau_{i\rightsquigarrow\ell} \in \mathcal{T}_{\vec{\tau}} \iff \vec{\tau} \subseteq \tau_{i\rightsquigarrow\ell}$. Then,

$$p(\vec{\tau}) = \frac{1}{\mathrm{Z}} \sum_{\tau_{i\rightsquigarrow\ell} \in \mathcal{T}_{\vec{\tau}}} w(\tau_{i\rightsquigarrow\ell})$$

We prove the lemma by induction on m.
Base Case: Then, $m = 1$ and $\vec{\tau} = \langle i_1 \xrightarrow{a_1} j_1 \rangle$. We have that

$$\frac{1}{\mathrm{Z}} \frac{\partial \mathrm{Z}}{\partial \mathrm{W}^{(a_1)}_{i_1 j_1}} \mathrm{W}^{(a_1)}_{i_1 j_1} = \frac{1}{\mathrm{Z}} \frac{\partial}{\partial \mathrm{W}^{(a_1)}_{i_1 j_1}} \left[\sum_{\tau_{i\rightsquigarrow\ell} \in \mathcal{T}} w(\tau_{i\rightsquigarrow\ell}) \right] \mathrm{W}^{(a_1)}_{i_1 j_1} \overset{\text{(a)}}{=} \frac{1}{\mathrm{Z}} \left(\sum_{\tau_{i\rightsquigarrow\ell} \in \mathcal{T}_{\vec{\tau}}} w(\tau_{i\rightsquigarrow\ell}) \right) = p(i_1 \xrightarrow{a_1} j_1)$$

Step (a) holds because taking the derivative of Z with respect to $\mathrm{W}^{(a_1)}_{i_1 j_1}$ yields the sum of the weights all trajectories which include $i_1 \xrightarrow{a_1} j_1$ where we exclude $\mathrm{W}^{(a_1)}_{i_1 j_1}$ from the computation of the weight. Then, we can push the outer $\mathrm{W}^{(a_1)}_{i_1 j_1}$ into the equation to obtain the sum of the weights of all trajectories containing $i_1 \xrightarrow{a_1} j_1$.
Inductive Step: Suppose that (10) holds for any m-tuple. Let $\vec{\tau} = \langle i_1 \xrightarrow{a_1} j_1, \ldots, i_{m+1} \xrightarrow{a_{m+1}} j_{m+1} \rangle$. Without loss of generality, fix $i_1 \xrightarrow{a_1} j_1$ and let $\vec{\tau}'$ be $\vec{\tau}$ without $i_1 \xrightarrow{a_1} j_1$.

$$\begin{aligned}
&\frac{1}{\mathrm{Z}} \sum_{n=1}^{m+1} \frac{\partial^n \mathrm{Z}}{\partial \mathrm{W}^{(a_1)}_{i_1 j_1} \ldots \partial \mathrm{W}^{(a_n)}_{i_n j_n}} \prod_{k=1}^{n} \mathrm{W}^{(a_k)}_{i_k j_k} \\
&\overset{\text{(b)}}{=} \mathrm{W}^{(a_1)}_{i_1 j_1} \frac{\partial}{\partial \mathrm{W}^{(a_1)}_{i_1 j_1}} \underbrace{\left[\frac{1}{\mathrm{Z}} \sum_{n=2}^{m+1} \frac{\partial^{(n-1)} \mathrm{Z}}{\partial \mathrm{W}^{(a_2)}_{i_2 j_2} \ldots \partial \mathrm{W}^{(a_n)}_{i_n j_n}} \prod_{k=2}^{n} \mathrm{W}^{(a_k)}_{i_k j_k} \right]}_{\text{Inductive hypothesis}} \\
&\overset{\text{(c)}}{=} \mathrm{W}^{(a_1)}_{i_1 j_1} \frac{\partial}{\partial \mathrm{W}^{(a_1)}_{i_1 j_1}} \overbrace{\left[\frac{1}{\mathrm{Z}} \sum_{\tau_{i\rightsquigarrow\ell} \in \mathcal{T}_{\vec{\tau}'}} w(\tau_{i\rightsquigarrow\ell}) \right]} \\
&\overset{\text{(d)}}{=} \frac{1}{\mathrm{Z}} \frac{\partial}{\partial \mathrm{W}^{(a_1)}_{i_1 j_1}} \left[\sum_{\tau_{i\rightsquigarrow\ell} \in \mathcal{T}_{\vec{\tau}'}} w(\tau_{i\rightsquigarrow\ell}) \right] \mathrm{W}^{(a_1)}_{i_1 j_1} \\
&\overset{\text{(e)}}{=} p(\vec{\tau})
\end{aligned}$$

Step (b) pushes $\frac{1}{\mathrm{Z}}$ and $\prod_{k=2}^{n} \mathrm{W}^{(a_k)}_{i_k j_k}$ as constants into the derivative and step (c) uses our induction hypothesis on $\vec{\tau}'$. Then, step (d) takes $\frac{1}{\mathrm{Z}}$ out of the derivative as we pushed it in as a constant. Finally, step (e) follows by the same reasoning as step (a) in the base case above. ■

A.3

Theorem 4. *Algorithm* E_2 *computes the second-order expectation of additively decomposable functions* $r\colon \mathcal{T} \mapsto \mathbb{R}^R$ *and* $t\colon \mathcal{T} \mapsto \mathbb{R}^T$ *in:*

$$\mathcal{O}(N^3 + N^2(\overline{R}\,\overline{T} + AR'T')) \textit{ time}$$
$$\mathcal{O}(N^2 + RT + N(R+T)) \textit{ space}$$

where $\overline{R} = \min(NR', R)$ *and* $\overline{T} = \min(NT', T)$.
Proof. We provide a proof of correctness (the time and space bounds are discussed in the main paper). Zmigrod et al. (2021) show that we can find second-order expectations over by finding the expectations over pairs of transitions. That is,

$$\mathbb{E}_{\tau_{i\rightsquigarrow\ell}} \left[r(\tau_{i\rightsquigarrow\ell}) t(\tau_{i\rightsquigarrow\ell})^{\top} \right] = \sum_{i,j,k,l=0}^{N} \sum_{a,b \in \overline{\mathcal{A}}} p\left(i \xrightarrow{a} j, k \xrightarrow{b} l \right) r^{(a)}_{ij} t^{(b)\top}_{kl}$$

We can use Lemma 2 for the $m = 2$ case, to find that the expectation is given by

$$\begin{aligned}
&\mathbb{E}_{\tau_{i\rightsquigarrow\ell}}\Big[r(\tau_{i\rightsquigarrow\ell})t(\tau_{i\rightsquigarrow\ell})^\top\Big] \\
&= \frac{1}{\mathrm{Z}}\Bigg[\sum_{i,j=0}^{N}\sum_{a\in\overline{\mathcal{A}}}\frac{\partial \mathrm{Z}}{\partial \mathrm{W}_{ij}^{(a)}}\mathrm{W}_{ij}^{(a)}r_{ij}^{(a)}t_{ij}^{(a)\top} + \sum_{i,j,k,l=0}^{N}\sum_{a,b\in\overline{\mathcal{A}}}\frac{\partial^2 \mathrm{Z}}{\partial \mathrm{W}_{ij}^{(a)}\partial \mathrm{W}_{kl}^{(b)}}\mathrm{W}_{ij}^{(a)}\mathrm{W}_{kl}^{(b)}r_{ij}^{(a)}t_{kl}^{(b)\top}\Bigg]
\end{aligned}$$

The first summand can be rewritten as

$$\sum_{i,j=0}^{N}\sum_{a\in\overline{\mathcal{A}}}\frac{\partial \mathrm{Z}}{\partial \mathrm{W}_{ij}^{(a)}}\mathrm{W}_{ij}^{(a)}r_{ij}^{(a)}t_{ij}^{(a)\top} = \sum_{i,j=0}^{N}\sum_{a\in\overline{\mathcal{A}}}\mathrm{s}_i\dot{\mathrm{W}}_{ij}^{(a)}\mathrm{e}_j\mathrm{W}_{ij}^{(a)}r_{ij}^{(a)}t_{ij}^{(a)\top}$$

The second summand can be rewritten as

$$\begin{aligned}
&\sum_{i,j,k,l=0}^{N}\sum_{a,b\in\overline{\mathcal{A}}}\frac{\partial^2 \mathrm{Z}}{\partial \mathrm{W}_{ij}^{(a)}\partial \mathrm{W}_{kl}^{(b)}}\mathrm{W}_{ij}^{(a)}\mathrm{W}_{kl}^{(b)}r_{ij}^{(a)}t_{kl}^{(b)\top} \\
&\quad=\sum_{i,j,k,l=0}^{N}\sum_{a,b\in\overline{\mathcal{A}}}\mathrm{s}_i\dot{\mathrm{W}}_{ij}^{(a)}\mathrm{W}_{jk}^{\star}\dot{\mathrm{W}}_{kl}^{(b)}\mathrm{e}_l\mathrm{W}_{ij}^{(a)}\mathrm{W}_{kl}^{(b)}r_{ij}^{(a)}t_{kl}^{(b)\top} + \mathrm{s}_k\dot{\mathrm{W}}_{kl}^{(b)}\mathrm{W}_{li}^{\star}\dot{\mathrm{W}}_{ij}^{(a)}\mathrm{e}_j\mathrm{W}_{ij}^{(a)}\mathrm{W}_{kl}^{(b)}r_{ij}^{(a)}t_{kl}^{(b)\top}
\end{aligned}$$

Consider the first summand of the above expression

$$\begin{aligned}
&\sum_{i,j,k,l=0}^{N}\sum_{a,b\in\overline{\mathcal{A}}}\mathrm{s}_i\dot{\mathrm{W}}_{ij}^{(a)}\mathrm{W}_{jk}^{\star}\dot{\mathrm{W}}_{kl}^{(b)}\mathrm{e}_l\mathrm{W}_{ij}^{(a)}\mathrm{W}_{kl}^{(b)}r_{ij}^{(a)}t_{kl}^{(b)\top} \\
&=\sum_{j,k=0}^{N}\underbrace{\left[\sum_{i=0}^{N}\sum_{a\in\overline{\mathcal{A}}}\mathrm{s}_i\dot{\mathrm{W}}_{ij}^{(a)}\mathrm{W}_{ij}^{(a)}r_{ij}^{(a)}\right]}_{\overset{\text{def}}{=}\widehat{r}_j^s}\mathrm{W}_{jk}^{\star}\underbrace{\left[\sum_{l=0}^{N}\sum_{b\in\overline{\mathcal{A}}}\dot{\mathrm{W}}_{kl}^{(b)}\mathrm{e}_l\mathrm{W}_{kl}^{(b)}t_{kl}^{(b)}\right]^\top}_{\overset{\text{def}}{=}\widehat{t}_k^{e\top}} \\
&=\sum_{j,k=0}^{N}\widehat{r}_j^s\mathrm{W}_{jk}^{\star}\widehat{t}_k^{e\top}
\end{aligned}$$

Similarly, the second summand can be written as

$$\sum_{j,k=0}^{N}\widehat{r}_k^e\mathrm{W}_{jk}^{\star}\widehat{t}_j^{s\top}$$

Finally, recomposing all the pieces together,

$$\mathbb{E}_{\tau_{i\rightsquigarrow\ell}}\Big[r(\tau_{i\rightsquigarrow\ell})t(\tau_{i\rightsquigarrow\ell})^\top\Big] = \frac{1}{\mathrm{Z}}\Bigg[\sum_{i,j=0}^{N}\widehat{r}_i^s\mathrm{W}_{ij}^{\star}\widehat{t}_j^{e\top} + \widehat{r}_j^e\mathrm{W}_{ij}^{\star}\widehat{t}_i^{s\top} + \sum_{a\in\overline{\mathcal{A}}}\mathrm{s}_i\dot{\mathrm{W}}_{ij}^{(a)}\mathrm{e}_j\mathrm{W}_{ij}^{(a)}r_{ij}^{(a)}t_{ij}^{(a)\top}\Bigg]$$

$\blacksquare$