Property 5 (insensitivity to $-\infty$). Since $\max_\Omega(\boldsymbol{x}) = \max_{\boldsymbol{q}\in\triangle^D} \langle \boldsymbol{q}, \boldsymbol{x}\rangle - \Omega(\boldsymbol{q})$, if $x_j = -\infty$, then $q_j = \nabla \max_\Omega(\boldsymbol{x})_j = 0$ is the only feasible solution for the j^{th} coordinate.

A.2. Proof of Proposition 1 (optimality of DP recursion)

Let $v_i(\boldsymbol{\theta})$ be the highest-score path up to node $i \in [N]$. Let $\mathcal{Y}_i$ be the set of paths $\boldsymbol{y} = (y_1, \dots, y_L)$ starting from node 1 and reaching node i, that is $y_1 = 1$ and $y_L = i$. Note that L may depend on $\boldsymbol{y}$ but we do not make this dependency explicit. Because nodes are sorted in topological order, we can compute $v_i(\boldsymbol{\theta})$ by

$$v_i(\boldsymbol{\theta}) = \max_{\boldsymbol{y}\in\mathcal{Y}_i} \sum_{t=2}^{L} \theta_{y_t,y_{t-1}} = \max_{\boldsymbol{y}\in\mathcal{Y}_i} \sum_{t=2}^{L-1} \theta_{y_t,y_{t-1}} + \theta_{y_L,y_{L-1}} = \max_{\boldsymbol{y}\in\mathcal{Y}_i} \sum_{t=2}^{L-1} \theta_{y_t,y_{t-1}} + \theta_{i,y_{L-1}}.$$

Recall that $\mathcal{P}_i$ is the set of parent nodes of node i. From the *associativity* of the max operator,

$$v_i(\boldsymbol{\theta}) = \max_{j\in\mathcal{P}_i} \max_{\substack{\boldsymbol{y}\in\mathcal{Y}_i \\ y_{L-1}=j}} \left(\sum_{t=2}^{L-1} \theta_{y_t,y_{t-1}} + \theta_{i,y_{L-1}} \right) = \max_{j\in\mathcal{P}_i} \max_{\substack{\boldsymbol{y}\in\mathcal{Y}_i \\ y_{L-1}=j}} \left(\sum_{t=2}^{L-1} \theta_{y_t,y_{t-1}} + \theta_{i,j} \right).$$

From the *distributivity* of $+$ over $\max$, we obtain

$$v_i(\boldsymbol{\theta}) = \max_{j\in\mathcal{P}_i} \left(\max_{\substack{\boldsymbol{y}\in\mathcal{Y}_i \\ y_{L-1}=j}} \sum_{t=2}^{L-1} \theta_{y_t,y_{t-1}} \right) + \theta_{i,j} = \max_{j\in\mathcal{P}_i} v_j(\boldsymbol{\theta}) + \theta_{i,j},$$

where we used the fact that the inner max operations are independent of $y_L = i$. This concludes the proof of the optimality of (3).

A.3. Proof of Proposition 2 (properties of $\text{DP}_\Omega(\boldsymbol{\theta})$)

We prove in this section the three main claims of Proposition 2. For the first two claims, we rewrite (3) and (6) using the following notations:

$$\begin{aligned}
&v_i^0(\boldsymbol{\theta}) \triangleq \max(\boldsymbol{u}_i^0(\boldsymbol{\theta})) \quad \text{and} \quad v_i^\Omega(\boldsymbol{\theta}) \triangleq \max(\boldsymbol{u}_i^\Omega(\boldsymbol{\theta})), \quad \text{where} \\
&\boldsymbol{u}_i^0(\boldsymbol{\theta}) \triangleq (\theta_{i,1} + v_1^0(\boldsymbol{\theta}), \dots, \theta_{i,i-1} + v_{i-1}^0(\boldsymbol{\theta}), -\infty, -\infty, \dots, -\infty) \in \mathbb{R}^N \quad \text{and} \\
&\boldsymbol{u}_i^\Omega(\boldsymbol{\theta}) \triangleq (\theta_{i,1} + v_1^\Omega(\boldsymbol{\theta}), \dots, \theta_{i,i-1} + v_{i-1}^\Omega(\boldsymbol{\theta}), \underbrace{-\infty}_{i}, -\infty, \dots, -\infty) \in \mathbb{R}^N.
\end{aligned}$$

These definitions are indeed valid as per Lemma 1, property 5.

Proof of $\text{DP}_\Omega(\boldsymbol{\theta})$ convexity. Since $v_1^\Omega(\boldsymbol{\theta}) = 0$, it is trivially convex. Assume that $v_2^\Omega(\boldsymbol{\theta}), \dots, v_{i-1}^\Omega(\boldsymbol{\theta})$ are convex. Then, $v_i^\Omega(\boldsymbol{\theta})$ is the composition of $\max_\Omega$ and $\boldsymbol{u}_i^\Omega$, a convex function and a function which outputs a vector whose each coordinate is convex in $\boldsymbol{\theta}$. By induction, since $\max_\Omega$ is non-decreasing per coordinate (cf. Lemma 1 property 4), $v_i^\Omega(\boldsymbol{\theta})$ is convex (*e.g.,* **?**, §3.2.4). Therefore $v_i^\Omega(\boldsymbol{\theta})$ is convex for all $i \in [N]$ and $\text{DP}_\Omega(\boldsymbol{\theta}) = v_N^\Omega(\boldsymbol{\theta})$ is convex.

Proof of $\text{DP}_\Omega(\boldsymbol{\theta})$ bound. We clearly have $v_1^\Omega(\boldsymbol{\theta}) \geq v_1^0(\boldsymbol{\theta})$. Assume that $v_j^\Omega(\boldsymbol{\theta}) \geq v_j^0(\boldsymbol{\theta}) - (j-1)U_{\Omega,N}$ for all $j \in \{2, \dots, i-1\}$. That is, $\boldsymbol{u}_i^\Omega(\boldsymbol{\theta}) \geq \boldsymbol{u}_i^0(\boldsymbol{\theta}) - (i-2)U_{\Omega,N}\mathbf{1}$, where $\mathbf{1} \in \mathbb{R}^N$ is the unit vector. Then, by induction, we have

$$\max_\Omega(\boldsymbol{u}_i^\Omega(\boldsymbol{\theta})) \geq \max_\Omega(\boldsymbol{u}_i^0(\boldsymbol{\theta})) - (i-2)U_{\Omega,N} \geq \max(\boldsymbol{u}_i^0(\boldsymbol{\theta})) - (i-1)U_{\Omega,N},$$

where we used Lemma 1, properties 1, 2 and 4. Therefore $v_i^\Omega(\boldsymbol{\theta}) \geq v_i^0(\boldsymbol{\theta}) - (i-1)U_{\Omega,N}$ for all $i \in [N]$ and hence, $\text{DP}_\Omega(\boldsymbol{\theta}) \geq \text{LP}(\boldsymbol{\theta}) - (N-1)U_{\Omega,N}$. Using a similar reasoning we

obtain $v_i^0(\boldsymbol{\theta}) - (i-1)L_{\Omega,N} \geq v_i^\Omega(\boldsymbol{\theta})$ and therefore $\mathrm{LP}(\boldsymbol{\theta}) - (N-1)L_{\Omega,N} \geq \mathrm{DP}_\Omega(\boldsymbol{\theta})$. To summarize, we obtain

$$\mathrm{LP}(\boldsymbol{\theta}) - (N-1)L_{\Omega,N} \geq \mathrm{DP}_\Omega(\boldsymbol{\theta}) \geq \mathrm{LP}(\boldsymbol{\theta}) - (N-1)U_{\Omega,N},$$

which concludes the proof. Note that using property 1 of Lemma 1, this immediately implies a bound involving $\mathrm{LP}_\Omega(\boldsymbol{\theta})$ instead of $\mathrm{LP}(\boldsymbol{\theta})$.

Proof that $\Omega = -\gamma H \Rightarrow \mathbf{DP}_\Omega(\boldsymbol{\theta}) = \mathbf{LP}_\Omega(\boldsymbol{\theta})$. We first show that $\max_\Omega$ is associative.

Lemma 2. *Associativity of $\max_\Omega$ when $\Omega = -\gamma H$*

We have $\max_\Omega(\max_\Omega(\boldsymbol{x}), c) = \max_\Omega(\boldsymbol{x}, c) \quad \forall \boldsymbol{x} \in \mathbb{R}^D, c \in \mathbb{R}$.

Proof. We simply use the closed form of $\max_\Omega$ when $\Omega = -\gamma H$ (cf. §B.1):

$$\begin{aligned}
\max_\Omega(\max_\Omega(\boldsymbol{x}), c) &= \gamma \log(\exp(\max_\Omega(\boldsymbol{x})/\gamma) + \exp(c/\gamma)) \\
&= \gamma \log\left(\exp\left(\log \sum_{i=1}^D \exp(x_i/\gamma)\right) + \exp(c/\gamma)\right) \\
&= \gamma \log\left(\sum_{i=1}^D \exp(x_i/\gamma) + \exp(c/\gamma)\right) \\
&= \max_\Omega(\boldsymbol{x}, c),
\end{aligned}$$

and the lemma follows. □

Using our shorthand notation, Lemma 2 can be used to write

$$\max_{\Omega \atop (y_1,\ldots,y_i,\ldots,y_L)} f(\boldsymbol{y}) = \max_{\Omega \atop v} \max_{\Omega \atop (y_1,\ldots,v,\ldots,y_L)} f(\boldsymbol{y}).$$

This is precisely the associative property that we used in the proof of Proposition 1. The second property that we used, the distributivity of $+$ over $\max$, holds for any $\max_\Omega$, as per Lemma 1 property 2. Thus, the same proof as Proposition 1 is also valid when we substitute $\max$ with $\max_\Omega$, when $\Omega = -\gamma H$, which yields $\mathrm{LP}_\Omega(\boldsymbol{\theta}) = \mathrm{DP}_\Omega(\boldsymbol{\theta})$.

Proof that $\Omega = -\gamma H \Leftarrow \mathbf{DP}_\Omega(\boldsymbol{\theta}) = \mathbf{LP}_\Omega(\boldsymbol{\theta})$. Mirroring the previous proof, we first characterize the regularizations Ω for which $\max_\Omega$ is associative.

Lemma 3. *Let $\Omega \colon \triangle^D \to \mathbb{R}$ be a regularization function, i.e., $\operatorname{dom}\Omega = \triangle^D$. Assume that there exist ω convex lower-semi-continuous defined on $[0,1]$ such that $\Omega(\boldsymbol{q}) = \sum_{i=1}^d \omega(q_i)$. If*

$$\max_\Omega(\max_\Omega(\boldsymbol{x}), c) = \max_\Omega(\boldsymbol{x}, c) \quad \forall \boldsymbol{x} \in \mathbb{R}^D, c \in \mathbb{R},$$

then $\Omega(\boldsymbol{q}) = -\gamma \sum_{i=1}^d q_i \log(q_i)$ for some $\gamma \geq 0$.

Proof. We start by writing the associativity property for three elements. For all $x_1, x_2, x_3 \in \mathbb{R}$,

$$\begin{aligned}
\max_\Omega\big((x_1, x_2, x_3)\big) &= \max_\Omega\big(\max_\Omega(x_1, x_2), x_3\big) \\
&= \max_{\substack{q+q_3=1 \\ q,q_3\geq 0}} q \max_{\substack{\tilde{q}_1+\tilde{q}_2=1 \\ \tilde{q}_i\geq 0}} \big(\tilde{q}_1 x_1 + \tilde{q}_2 x_2 - \omega(\tilde{q}_1) - \omega(\tilde{q}_2)\big) + q_3 x_3 - \omega(q_3) - \omega(q) \\
&= \max_{\substack{q_1+q_2+q_3=1 \\ q_i\geq 0}} q_1 x_1 + q_2 x_2 + q_3 x_3 - \Phi(q_1, q_2, q_3), \quad \text{where} \\
\Phi(q_1, q_2, q_3) &\triangleq (q_1+q_2)\Big(\omega\big(\frac{q_1}{q_1+q_2}\big) + \omega\big(\frac{q_2}{q_1+q_2}\big)\Big) + \omega(q_1+q_2) + \omega(q_3).
\end{aligned}$$

We have performed a variable change $q_{1,2} = q\,\tilde{q}_{1,2}$ at the second line, and noticed $q = q_1 + q_2$. Therefore

$$\max_\Omega\big((x_1, x_2, x_3)\big) = \Phi^\star(x_1, x_2, x_3),$$

where $\Phi^\star$ is the convex conjugate of Φ restricted to $]0, 1]^3$. By definition, we also have $\max_\Omega\big((x_1, x_2, x_3)\big) = \Omega^\star(x_1, x_2, x_3)$, so that $\Omega^\star = \Phi^\star$ on $\mathbb{R}^3$. As Ω is convex and lower semi-continous, we can apply Moreau-Yoshida theorem and obtain $\Omega^{\star\star} = \Omega = \Phi^{\star\star} \leq \Phi$.

Suppose that there exists $\boldsymbol{q} = (q_1, q_2, q_3) \in \triangle^3$ such that $\Phi(q_1, q_2, q_3) < \Omega(q_1, q_2, q_3)$. Given the forms of Φ and Ω, $\Phi(q_1, q_2, 0) < \Omega(q_1, q_2, 0)$. We let $\boldsymbol{x} = (x_1, x_2, -\infty) \in \mathbb{R}^3$ such that

$$\begin{aligned}\max_\Omega(x_1, x_2, -\infty) = \max_\Omega(x_1, x_2) = x_1 q_1 + x_2 q_2 - \omega(q_1) - \omega(q_2) = \langle \boldsymbol{x}, \boldsymbol{q}\rangle - \Omega(\boldsymbol{q}) \\ < \langle \boldsymbol{x}, \boldsymbol{q}\rangle - \Phi(\boldsymbol{q}) \leq \max_{\boldsymbol{q}\in\triangle^3} \langle \boldsymbol{x}, \boldsymbol{q}\rangle - \Phi(\boldsymbol{q}) = \max_\Omega\big(\max_\Omega(x_1, x_2), -\infty\big),\end{aligned}$$

leading to a contradiction. Therefore $\Omega \geq \Phi$ over $\triangle^3$, and finally $\Omega = \Phi$. We have used the fact that the operator $\nabla\max_\Omega : \mathbb{R}^2 \to \triangle^2$ is surjective, as $\triangle^2$ is a one-dimensional segment, $\nabla\max_\Omega$ is continuous and reaches the extreme values $\nabla\max_\Omega(0, -\infty) = (1, 0)$ and $\nabla\max_\Omega(-\infty, 0) = (0, 1)$ — which allows to use the intermediate value theorem.

To conclude, for all $q_1, q_2 \in]0, 1]$ such that $q_1 + q_2 \leq 1$, we have

$$\begin{aligned}\omega(q_1) + \omega(q_2) &= (q_1 + q_2)\Big(\omega\big(\frac{q_1}{q_1 + q_2}\big) + \omega\big(\frac{q_2}{q_1 + q_2}\big)\Big) + \omega(q_1 + q_2) \\ \omega(xy) + \omega((1 - x)y) - \omega(y) &= y(\omega(x) + \omega(1 - x)) \quad \forall\, 0 < y \leq 1,\ 0 < x < 1, \qquad (8)\end{aligned}$$

where we have set $y = q_1 + q_2$ and $x = \frac{q_1}{q_1 + q_2}$. The functional equation (8) was first studied in the field of information theory. As first shown by **?**, Theorem 0, and further extended (**?**), all measurable solutions have the form

$$\omega(x) = -\gamma x \log(x),$$

where $\gamma \geq 0$ is a constant. The lemma follows. □

Assuming that Ω is not equal to $-\gamma H$ for any $\gamma \geq 0$, the previous lemma tells us that the associativity property is not met for a triplet $(x_1, x_2, x_3) \in \mathbb{R}^3$. In Figure 5, we construct a graph G such that

$$\mathrm{DP}_\Omega(\boldsymbol{\theta}) = \max_\Omega(\max_\Omega(x_1, x_2), x_3) \neq \mathrm{LP}_\Omega(\boldsymbol{\theta}) = \max_\Omega(x_1, x_2, x_3)$$

The proposition follows.

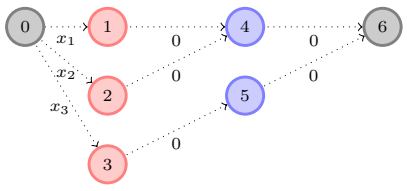

Figure 5. In general, $v_6(\boldsymbol{\theta}) = \mathrm{DP}_\Omega(\boldsymbol{\theta}) \neq \mathrm{LP}_\Omega(\boldsymbol{\theta})$.

A.4. Computation of $\nabla\mathrm{LP}_\Omega(\boldsymbol{\theta})$ and interpretation as an expectation

We show that $\nabla\mathrm{LP}_\Omega(\boldsymbol{\theta}) \in \mathrm{conv}(\mathcal{Y})$, and characterize a path distribution of which $\nabla\mathrm{LP}_\Omega(\boldsymbol{\theta})$ is the expectation.

Convex hull of $\mathcal{Y}$. We rewrite $\mathrm{LP}_\Omega(\boldsymbol{\theta}) = \max_\Omega(\boldsymbol{u}(\boldsymbol{\theta}))$, where $\boldsymbol{u}(\boldsymbol{\theta}) \triangleq (\langle \boldsymbol{Y}, \boldsymbol{\theta} \rangle)_{\boldsymbol{Y} \in \mathcal{Y}}$. Using the chain rule, we have

$$\nabla \mathrm{LP}_\Omega(\boldsymbol{\theta}) = \boldsymbol{J}_{\boldsymbol{u}}(\boldsymbol{\theta})^\top \nabla \max_\Omega(\boldsymbol{u}(\boldsymbol{\theta})), \tag{9}$$

where $\boldsymbol{J}_{\boldsymbol{u}}$ is the Jacobian of $\boldsymbol{u}$ w.r.t. $\boldsymbol{\theta}$, a matrix of size $|\mathcal{Y}| \times (N \times N)$. The horizontal slices of $\boldsymbol{J}_{\boldsymbol{u}}$ are exactly all the paths $\boldsymbol{Y}$ of $\mathcal{Y}$. Using $\nabla \max_\Omega(\boldsymbol{u}(\boldsymbol{\theta})) \in \triangle^{|\mathcal{Y}|}$, we conclude that $\nabla \mathrm{LP}_\Omega(\boldsymbol{\theta}) \in \mathrm{conv}(\mathcal{Y})$.

Induced distribution. From (9), we see that $\nabla \mathrm{LP}_\Omega(\boldsymbol{\theta}) = \sum_{\boldsymbol{Y} \in \mathcal{Y}} p_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y})\ \boldsymbol{Y}$, where we defined the distribution

$$p_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y}) \triangleq \Big(\nabla \max_\Omega(\boldsymbol{u}(\boldsymbol{\theta}))\Big)_{\boldsymbol{Y}}.$$

Unfortunately, since $\boldsymbol{u}(\boldsymbol{\theta}) \in \mathbb{R}^{|\mathcal{Y}|}$, computing $p_{\boldsymbol{\theta},\Omega}(\boldsymbol{Y})$, let alone the expectation $\mathbb{E}_{\boldsymbol{\theta},\Omega}[\boldsymbol{Y}]$ under that distribution, is intractable for general Ω.

A.5. Proof of Proposition 3 (computation of $\nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$)

Gradient computation. We first derive the recursion over $\boldsymbol{E} \triangleq \nabla \mathrm{DP}_\Omega(\boldsymbol{\theta})$ using sensitivity analysis, a.k.a backpropagation calculus. For any $(i,j) \in \mathcal{E}$, since $\theta_{i,j}$ influences only v_i, a straighforward application of the chain rule gives

$$e_{i,j} = \frac{\partial v_N}{\partial \theta_{i,j}} = \frac{\partial v_N}{\partial v_i} \frac{\partial v_i}{\partial \theta_{i,j}}. \tag{10}$$

Recall that $\boldsymbol{v} = (v_1, \dots, v_N)$ and $\boldsymbol{q}_i \triangleq \nabla \max_\Omega(\boldsymbol{\theta}_i + \boldsymbol{v})$. With this vector defined, we can now easily derive the two terms on the r.h.s of (10). Differentiating (6) w.r.t. $\theta_{i,j}$ straighforwardly gives the second term $\frac{\partial v_i}{\partial \theta_{i,j}} = q_{i,j}$.

The first term must be computed recursively. Recall that $\mathcal{C}_j$ denotes the children of node j. Since a node j influences only its children $i \in \mathcal{C}_j$, using the chain rule, we get

$$\frac{\partial v_N}{\partial v_j} = \sum_{i \in \mathcal{C}_j} \frac{\partial v_N}{\partial v_i} \frac{\partial v_i}{\partial v_j} \triangleq \bar{e}_j. \tag{11}$$

Differentiating (6) w.r.t. v_j again gives $\frac{\partial v_i}{\partial v_j} = q_{i,j}$. By definition, we also have $\frac{\partial v_N}{\partial v_i} = \bar{e}_i$ and $e_{i,j} = \bar{e}_i q_{i,j}$. Hence,

$$\bar{e}_j = \sum_{i \in \mathcal{C}_j} \bar{e}_i q_{i,j} = \sum_{i \in \mathcal{C}_j} e_{i,j}.$$

Combining the above, for any $j \in [N-1]$, we obtain the following two-step recursion

$$\forall\, i \in \mathcal{C}_j,\ e_{i,j} = \bar{e}_i q_{i,j} \quad \text{and} \quad \bar{e}_j = \sum_{i \in \mathcal{C}_j} e_{i,j}.$$

The values $(e_{i,j})_{(i,j) \in \mathcal{E}}$ can thus be computed in reverse topological order over the nodes of G, initializing $\bar{e}_N = \frac{\partial v_N}{\partial v_N} = 1$. The pseudo-code is summarized in Algorithm 1.

Associated random walk. It remains to show that $\boldsymbol{E}$ is also the expectation of $\boldsymbol{Y} \in \mathcal{Y}$ support of the following random walk, defined informally in the main text. Formally, we define the random sequence $(w_t)_t$ as

$$w_0 = N, \quad \forall\, t > 0, \forall\, i \in [N], \forall\, j \in \mathcal{P}_i, \quad \mathbb{P}[w_t = j | w_{t-1} = i] = q_{i,j}.$$

We set $y_{i,j} \triangleq \mathbf{1}\{\exists\, t > 0 \text{ s.t. } w_{t-1} = i, w_t = j\}$ where $\mathbf{1}$ is the characteristic function of an event, thereby defining a random variable $\boldsymbol{Y} \in \mathcal{Y}$, with distribution $\mathcal{D}$. We leave implicit the dependency of $\mathbb{P}$ in $\boldsymbol{\theta}$ and Ω. As the depth of w_t (number of edges to connect to the root

Algorithm 1 Compute $\mathrm{DP}_\Omega(\boldsymbol{\theta})$ and $\nabla\mathrm{DP}_\Omega(\boldsymbol{\theta})$

Input: Edge weights $\boldsymbol{\theta} \in \mathbb{R}^{N\times N}$
$v_1 \leftarrow 0, \quad \bar{e}_N \leftarrow 1, \qquad \boldsymbol{Q}, \boldsymbol{E} \leftarrow \mathbf{0} \in \mathbb{R}^{N\times N}$
for $i \in [2, \dots, N]$ **do** ▷ Topological order
$v_i \leftarrow \max_\Omega \theta_{i,j} + v_j$
$(\boldsymbol{q}_{i,j})_{j\in\mathcal{P}_i} \leftarrow \nabla\max_\Omega \theta_{i,j} + v_j$
for $j \in [N-1, \dots, 1]$ **do** ▷ Reverse topological order
$\forall\, i \in \mathcal{C}_j,\ e_{i,j} \leftarrow q_{i,j}\bar{e}_i, \qquad \bar{e}_j \leftarrow \sum_{i\in\mathcal{C}_j} e_{i,j}$
Return: $\mathrm{DP}_\Omega(\boldsymbol{\theta}) = v_N$, $\nabla\mathrm{DP}_\Omega(\boldsymbol{\theta}) = \boldsymbol{E} \in \mathbb{R}^{N\times N}$
Intermediate computation for Algorithm 2
$\bar{\boldsymbol{e}} \triangleq [\bar{e}]_{i=1}^N \in \mathbb{R}^N$, $\boldsymbol{Q} \in \mathbb{R}^{N\times N}$

Algorithm 2 Compute $\langle\nabla\mathrm{DP}_\Omega(\boldsymbol{\theta}), \boldsymbol{Z}\rangle$ and $\nabla^2\mathrm{DP}_\Omega(\boldsymbol{\theta})\boldsymbol{Z}$

Input: Edge weights and perturbation $\boldsymbol{\theta}, \boldsymbol{Z} \in \mathbb{R}^{N\times N}$
Call Algorithm 1 with input $\boldsymbol{\theta}$ to get $\bar{\boldsymbol{e}}$ and $\boldsymbol{Q}$
$\dot{v}_1 \leftarrow 0; \qquad \dot{e_N} \leftarrow 0, \qquad \dot{\boldsymbol{Q}}, \dot{\boldsymbol{E}} \leftarrow \mathbf{0} \in \mathbb{R}^{N\times N}$
for $i \in [2, \dots, N]$ **do** ▷ Topological order
$\dot{v}_i \leftarrow \sum_{j\in\mathcal{P}_i} q_{i,j}(z_{i,j} + \dot{v}_j)$ (A1)
$(\dot{\boldsymbol{q}}_{i,j})_{j\in\mathcal{P}_i} \leftarrow \boldsymbol{J}_\Omega\big((\boldsymbol{q}_{i,j})_{j\in\mathcal{P}_i}\big)(z_{i,j} + \dot{v}_j)_{j\in\mathcal{P}_i}$ (A2)
for $j \in [N-1, \dots, 1]$ **do** ▷ Reverse topological order
$\forall\, i \in \mathcal{C}_j,\ \dot{e}_{i,j} \leftarrow \dot{q}_{i,j}\bar{e}_i + q_{i,j}\dot{\bar{e}}_i$ (A3)
$\dot{\bar{e}}_j \leftarrow \sum_{i\in\mathcal{C}_j} \dot{e}_{i,j}$
Return: $\langle\nabla\mathrm{DP}_\Omega(\boldsymbol{\theta}), \boldsymbol{Z}\rangle = \dot{v}_N$
$\nabla^2\mathrm{DP}_\Omega(\boldsymbol{\theta})\boldsymbol{Z} = \dot{\boldsymbol{E}} \in \mathbb{R}^{N\times N}$

node) is stricly decreasing with t, $(w_t)_t$ reaches node 1 in finite time with probability one and is constant after this event. We introduce the random variables $(\bar{y}_j)_j$, defined for all $j \in [N]$ as

$$\bar{y}_j \triangleq \mathbf{1}\{\exists\, t \geq 0, w_t = j\} = \sum_{i\in\mathcal{C}_j} y_{i,j} \text{ if } j \neq N, 0 \text{ otherwise.}$$

By definition, using the fact that $\mathbb{P}[w_t = j | w_{t-1} = i]$ is independent of t (Markov property), for all $i \in \mathcal{C}_j$ and for all $j \in [N-1]$, we have

$$\mathbb{P}[y_{i,j} = 1] = \mathbb{E}[y_{i,j}] = \mathbb{P}[\exists\, t > 0, w_{t-1} = i]\mathbb{P}[w_t = j | w_{t-1} = i] = \mathbb{E}[\bar{y}_i]q_{i,j}.$$

Linearity of the expectation then provides

$$\mathbb{E}[\bar{y}_j] = \sum_{i\in\mathcal{C}_j} \mathbb{E}[y_{i,j}],$$

with initialization $\mathbb{E}[\bar{y}_N] = 1$. We recover the same two-step recursion as the one defining $\boldsymbol{E}$ and $\bar{\boldsymbol{e}}$, with the same initialization. Hence the probabilistic interpretation of the gradient, where the expectation is taken with respect to the distribution $\mathcal{D}$ of $\boldsymbol{Y}$:

$$\boldsymbol{E} = \mathbb{E}_{\boldsymbol{\theta},\Omega}[\boldsymbol{Y}] \quad \text{and} \quad \bar{\boldsymbol{e}} = \mathbb{E}_{\boldsymbol{\theta},\Omega}[\bar{\boldsymbol{y}}].$$

A.6. Computation of the directional derivative $\langle\nabla\mathbf{DP}_\Omega(\boldsymbol{\theta}), \boldsymbol{Z}\rangle$

The derivations of the following two sections allows to write Algorithm 2. Let $\dot{v}_i \triangleq \langle\nabla v_i(\boldsymbol{\theta}), \boldsymbol{Z}\rangle$, where $v_i(\boldsymbol{\theta})$ is defined in (6). Since v_i only directly depends on $v_j + \theta_{i,j}$ for $j \in \mathcal{P}_i$, a straighforward differentiation of $\langle\nabla v_i(\boldsymbol{\theta}), \boldsymbol{Z}\rangle$ gives

$$\dot{v}_i = \sum_{j\in\mathcal{P}_i} \frac{\partial v_i}{\partial v_j}\ (\dot{v}_j + z_{i,j})\,.$$

Recall that $\frac{\partial v_i}{\partial v_j} = q_{i,j}$ and has already been obtained when computing $\nabla\mathrm{DP}_\Omega(\boldsymbol{\theta})$. Hence equation (A1), reproduced here:

$$\forall\, i \in [2, \dots, N]: \qquad \dot{v}_i = \sum_{j\in\mathcal{P}_i} q_{i,j}(\dot{v}_j + z_{i,j}). \tag{12}$$

This recursion can be computed in topological order, starting from $\dot{v}_1 = 0$ to finish at $\dot{v}_N = \langle\nabla\mathrm{DP}_\Omega(\boldsymbol{\theta}), \boldsymbol{Z}\rangle$.

A.7. Computation of the Hessian-vector product $\nabla^2\mathrm{DP}_\Omega(\boldsymbol{\theta})\boldsymbol{Z}$

For convenience, let us define $\nabla^2\mathrm{DP}_\Omega(\boldsymbol{\theta})\boldsymbol{Z} \triangleq \dot{\boldsymbol{E}}$. For $(i,j) \notin \mathcal{E}$, we evidently have $\dot{e}_{i,j} = 0$. For $(i,j) \in \mathcal{E}$, since $\theta_{i,j}$ influences only v_i and $\dot{v}_i$, we obtain

$$\dot{e}_{i,j} = \frac{\partial \dot{v}_N}{\partial \theta_{i,j}} = \frac{\partial \dot{v}_N}{\partial v_i}\frac{\partial v_i}{\partial \theta_{i,j}} + \frac{\partial \dot{v}_N}{\partial \dot{v}_i}\frac{\partial \dot{v}_i}{\partial \theta_{i,j}}.$$

We will now show how to derive each of the right-hand side terms in turn. We already know that $\frac{\partial v_i}{\partial \theta_{i,j}} = q_{i,j}$. We also have $\frac{\partial \dot{v}_N}{\partial \dot{v}_i} = u_i$. Indeed, observe that $\dot{v}_j$ only directly influences $\dot{v}_i$ for $i \in \mathcal{C}_i$. Therefore, we have

$$\frac{\partial \dot{v}_N}{\partial \dot{v}_j} = \sum_{i \in \mathcal{C}_j} \frac{\partial \dot{v}_N}{\partial \dot{v}_i} q_{i,j} \quad \forall j \in [N-1] \tag{13}$$

and $\frac{\partial \dot{v}_N}{\partial \dot{v}_1} = 1$. Comparing (11) and (13), we see that $(\frac{\partial \dot{v}_N}{\partial \dot{v}_i})_i$ follows the same recursion as $(\frac{\partial v_N}{\partial v_i})_i$. Since $\frac{\partial \dot{v}_N}{\partial \dot{v}_n} = \frac{\partial v_N}{\partial v_n}$, both sequences are equal:

$$\frac{\partial \dot{v}_N}{\partial \dot{v}_i} = \frac{\partial v_N}{\partial v_i} = e_i.$$

Next, we derive $\frac{\partial \dot{v}_i}{\partial \theta_{i,j}}$. Since, for $j \in \mathcal{P}_i$, $\dot{v}_j + z_{i,j}$ does not depend on $\theta_{i,j}$, differentiating (12) w.r.t. $\theta_{i,j}$, we obtain

$$\begin{aligned}\frac{\partial \dot{v}_i}{\partial \theta_{i,j}} &= \sum_{k \in \mathcal{P}_i} \frac{\partial q_{i,j}}{\partial \theta_{i,j}}(\dot{v}_k + z_{i,k}) \\ &= \sum_{k \in \mathcal{P}_i} \frac{\partial^2 v_i}{\partial \theta_{i,j}\partial \theta_{i,k}}(\dot{v}_k + z_{i,k}) \triangleq \dot{q}_{i,j}.\end{aligned}$$

This can be conveniently rewritten in a vectorial form as

$$\dot{\boldsymbol{q}}_i = \nabla^2 \max_\Omega(\boldsymbol{\theta}_i + \boldsymbol{v})\,(\boldsymbol{z}_i + \dot{\boldsymbol{v}}) = \boldsymbol{J}_\Omega(\boldsymbol{q}_i)\,(\boldsymbol{z}_i + \dot{\boldsymbol{v}}),$$

where we have defined $\dot{\boldsymbol{v}} \triangleq (\dot{v}_1, \ldots, \dot{v}_N)$ and where we have used the function $\boldsymbol{J}_\Omega$ defined in §B.1, that conveniently computes the Hessian of $\max_\Omega$ from its gradient. The Hessian has this form for both negentropy and ℓ_2^2 regularizations. In a practical implementation, we only need to compute the coordinates (i,j) of $\dot{\boldsymbol{Q}}$, for $j \in \mathcal{P}_i$. Namely, as specified in (A2),

$$(\dot{\boldsymbol{q}}_{i,j})_{j \in \mathcal{P}_i} \leftarrow \boldsymbol{J}_\Omega\big((\boldsymbol{q}_{i,j})_{j \in \mathcal{P}_i}\big)(z_{i,j} + \dot{v}_j)_{j \in \mathcal{P}_i}.$$

Finally, we derive $\frac{\partial \dot{v}_N}{\partial v_i}$. Since v_j influences only v_i and $\dot{v}_i$ for $i \in \mathcal{C}_j$, the chain rule gives

$$\frac{\partial \dot{v}_N}{\partial v_i} = \sum_{j \in \mathcal{C}_i} \frac{\partial \dot{v}_N}{\partial v_j}\frac{\partial v_j}{\partial v_i} + \frac{\partial \dot{v}_N}{\partial \dot{v}_j}\frac{\partial \dot{v}_j}{\partial v_i} = \sum_{j \in \mathcal{C}_j} \dot{e}_{i,j} \triangleq \dot{\dot{e}}_i.$$

Combining the above, for any $j \in [N-1]$, we obtain the following two-step recursion (A3), reproduced here:

$$\forall\, i \in \mathcal{C}_j, \quad \dot{e}_{i,j} = \dot{q}_{i,j} e_i + q_{i,j}\dot{\dot{e}}_i \quad \text{and} \quad \dot{\dot{e}}_j = \sum_{i \in \mathcal{C}_j} \dot{e}_{i,j}.$$

Similarly to the computation of $\nabla\mathrm{DP}_\Omega(\boldsymbol{\theta})$, our algorithm computes this recursion in reverse topological order over the graph G, yielding $\nabla^2\mathrm{DP}_\Omega(\boldsymbol{\theta})\boldsymbol{Z} = \dot{\boldsymbol{E}}$.

B. Examples of algorithm instantiations

We provide the explicit forms of $\max_\Omega$ and its derivative for the negentropy and ℓ_2^2 regularizations. Then, we provide details and pseudo-code for the two instances of differentiable dynamic programming presented in §3.

B.1. Examples of $\max_\Omega$

Negative entropy. When $\Omega(\boldsymbol{q}) = \gamma \sum_{i=1}^D q_i \log q_i$, where $\gamma > 0$ (smaller is less regularized), we obtain

$$\begin{aligned}
\max\nolimits_\Omega(\boldsymbol{x}) &= \gamma \log \left(\sum_{i=1}^D \exp(x_i/\gamma) \right) \\
\nabla \max\nolimits_\Omega(\boldsymbol{x}) &= \exp(\boldsymbol{x}/\gamma) \Big/ \sum_{i=1}^D \exp(x_i/\gamma) \\
\nabla^2 \max\nolimits_\Omega(\boldsymbol{x}) &= \boldsymbol{J}_\Omega(\nabla \max\nolimits_\Omega(\boldsymbol{x})),
\end{aligned}$$

where $\boldsymbol{J}_\Omega(\boldsymbol{q}) \triangleq (\mathrm{Diag}(\boldsymbol{q}) - \boldsymbol{q}\boldsymbol{q}^\top)/\gamma$. Note that $\nabla \max_\Omega(\boldsymbol{x})$ recovers the usual "softmax" with temperature $\gamma = 1$. For a proof of the expression of $\max_\Omega$, see, *e.g.,* (**?**, Example 3.25).

Squared ℓ_2 norm. When $\Omega(\boldsymbol{x}) = \frac{\gamma}{2}\|\boldsymbol{x}\|_2^2$ with $\gamma > 0$, we obtain the following expressions

$$\begin{aligned}
\max\nolimits_\Omega(\boldsymbol{x}) &= \langle \boldsymbol{q}^\star, \boldsymbol{x} \rangle - \frac{\gamma}{2}\|\boldsymbol{q}^\star\|_2^2 \\
\nabla \max\nolimits_\Omega(\boldsymbol{x}) &= \operatorname*{argmin}_{\boldsymbol{q} \in \triangle^D} \|\boldsymbol{q} - \boldsymbol{x}/\gamma\|_2^2 = \boldsymbol{q}^\star \\
\nabla^2 \max\nolimits_\Omega(\boldsymbol{x}) &= \boldsymbol{J}_\Omega(\nabla \max\nolimits_\Omega(\boldsymbol{x})),
\end{aligned}$$

where $\boldsymbol{J}_\Omega(\boldsymbol{q}) \triangleq (\mathrm{Diag}(\boldsymbol{s}) - \boldsymbol{s}\boldsymbol{s}^\top/\|\boldsymbol{s}\|_1)/\gamma$ and $\boldsymbol{s} \in \{0,1\}^D$ is a vector that indicates the support of $\boldsymbol{q}$. Note that $\nabla \max_\Omega(\boldsymbol{x})$ is precisely the Euclidean projection onto the simplex of $\boldsymbol{x}/\gamma$ and can be computed exactly in worst-case $\mathcal{O}(D \log D)$ time using the algorithm of (**?**) or in expected $\mathcal{O}(D)$ time using the randomized pivot algorithm of (**?**). It can be efficiently performed on Nvidia GPUs since recently. An important benefit of the squared ℓ_2 norm, compared to the negative entropy, is that $\nabla \max_\Omega(\boldsymbol{x})$ tends to be sparse. This is useful, among other things, to define sparse attention mechanisms (**??**).

B.2. Sequence prediction with the smoothed Viterbi algorithm

Computational graph. As illustrated in §3, the DAG contains a start node, S nodes for each time step and end node. Therefore $|\mathcal{V}| = N = TS + 2$. Only nodes from consecutive time steps are connected to each other. Taking into account the start and end nodes, the total number of edges is therefore $|\mathcal{E}| = (T-1)S^2 + 2S$.

Representation. We follow the notation of §3, *i.e.* we represent $\boldsymbol{Y}$ and $\boldsymbol{\theta}$ as $T \times S \times S$ tensors (we can safely ignore the edges connected to the end node since their value is 0). We represent $\boldsymbol{Y}$ as a binary tensor such that $y_{t,i,j} = 1$ if $\boldsymbol{Y}$ is in states i and j in time steps t and $t-1$, and $y_{t,i,j} = 0$ otherwise. Likewise, we represent the potentials $\boldsymbol{\theta}$ as a real tensor such that $\theta_{t,i,j}$ contains the potential of transitioning from state j to state i on time t.

Algorithms. Applying recursion (6) to this specific DAG, we obtain a smoothed version of the Viterbi algorithm. Let $v_{t,i}$ be the score of being in state i up to time t. We can rewrite the smoothed Bellman recursion as

$$v_{t,i}(\boldsymbol{\theta}) \triangleq \max_{j \in [S]}\nolimits_\Omega \, v_{t-1,j}(\boldsymbol{\theta}) + \theta_{t,i,j} = \max\nolimits_\Omega(\boldsymbol{v}_{t-1}(\boldsymbol{\theta}) + \boldsymbol{\theta}_{t,i}).$$

Algorithm 3 Compute $\text{Vit}_\Omega(\boldsymbol{\theta})$ and $\nabla\text{Vit}_\Omega(\boldsymbol{\theta})$

Input: Potential scores $\boldsymbol{\theta} \in \mathbb{R}^{T\times S\times S}$
$\triangleright$ Forward pass
$\boldsymbol{v}_0 = \mathbf{0}_S$
for $t \in [1,\dots,T], i \in [S]$ **do**
  $v_{t,i} = \max_\Omega(\boldsymbol{\theta}_{t,i} + \boldsymbol{v}_{t-1})$
  $\boldsymbol{q}_{t,i} = \nabla\max_\Omega(\boldsymbol{\theta}_{t,i} + \boldsymbol{v}_{t-1})$
$v_{T+1,1} = \max_\Omega(\boldsymbol{v}_T); \quad \boldsymbol{q}_{T+1,1} = \nabla\max_\Omega(\boldsymbol{v}_T)$
$\triangleright$ Backward pass
$\boldsymbol{u}_{T+1} = (1,0,\dots,0) \in \mathbb{R}^S$
for $t \in [T,\dots,0], j \in [S]$ **do**
  $\boldsymbol{e}_{t,\cdot,j} = \boldsymbol{q}_{t+1,\cdot,j} \circ \boldsymbol{u}_{t+1}; \quad u_{t,j} = \langle \boldsymbol{e}_{t,\cdot,j}, \mathbf{1}_S\rangle$
Return: $\text{Vit}_\Omega(\boldsymbol{\theta}) = v_{T+1,1}$
$\nabla\text{Vit}_\Omega(\boldsymbol{\theta}) = (e_{t-1,i,j})_{t=1,i,j=1}^{T,S,S}$
Intermediary computations for Alg. 4:
$\boldsymbol{Q} \triangleq (q)_{t=1,i,j=1}^{T+1,S,S}, \boldsymbol{U} \triangleq (u)_{t=1,j=1}^{T+1,S}$

Algorithm 4 Compute $\langle\nabla\text{Vit}_\Omega(\boldsymbol{\theta}), \boldsymbol{Z}\rangle$ and $\nabla^2\text{Vit}_\Omega(\boldsymbol{\theta})\boldsymbol{Z}$

Input: $\boldsymbol{Z} \in \mathbb{R}^{T\times S\times S}, \boldsymbol{\theta} \in \mathbb{R}^{T\times S\times S}$
Call Alg. 3 with input $\boldsymbol{\theta}$ to get $\boldsymbol{U}, \boldsymbol{Q}$
$\triangleright$ Forward pass
$\dot{v}_0 = \mathbf{0}_S$
for $t \in [1,\dots,T], i \in [S]$ **do**
  $\dot{v}_{t,i} = \langle \boldsymbol{q}_{t,i}, \boldsymbol{z}_{t,i} + \dot{\boldsymbol{v}}_{t-1}\rangle$
  $\dot{\boldsymbol{q}}_{t,i} = \boldsymbol{J}_\Omega(\boldsymbol{q}_{t,i})\,(\boldsymbol{z}_t + \dot{\boldsymbol{v}}_{t-1})$
$\dot{v}_{T+1,1} = \langle \boldsymbol{q}_{T+1,1}, \dot{\boldsymbol{v}}_T\rangle; \quad \dot{\boldsymbol{q}}_{T+1,1} = \boldsymbol{J}_\Omega(\dot{\boldsymbol{q}}_{T+1,1})\,\dot{\boldsymbol{v}}_T$
$\triangleright$ Backward pass
$\dot{\boldsymbol{u}}_{T+1} = \mathbf{0}_S; \quad \dot{\boldsymbol{Q}}_{T+1} = \mathbf{0}_{S\times S}$
for $t \in [T,\dots,0], j \in [S]$ **do**
  $\dot{\boldsymbol{e}}_{t,\cdot,j} = \boldsymbol{q}_{t+1,\cdot,j} \circ \dot{\boldsymbol{u}}_{t+1} + \dot{\boldsymbol{q}}_{t+1,\cdot,j} \circ \boldsymbol{u}_{t+1}$
  $\dot{u}_{t,j} = \langle \dot{\boldsymbol{e}}_{t,\cdot,j}, \mathbf{1}_S\rangle$
Return: $\langle\text{Vit}_\Omega(\boldsymbol{\theta}), \boldsymbol{Z}\rangle = \dot{v}_{T+1}$
$\nabla^2\text{Vit}_\Omega(\boldsymbol{\theta})\boldsymbol{Z} = (\dot{e}_{t-1,i,j})_{t=1,i,j=1}^{T,S,S}$

The value $\text{Vit}_\Omega(\boldsymbol{\theta}) \triangleq \max_\Omega(\boldsymbol{v}_T(\boldsymbol{\theta}))$ can be computed in topological order, starting from $\boldsymbol{v}_0(\boldsymbol{\theta})$. The total computational cost is $\mathcal{O}(TS^2)$. Using the computations of §2.3 and §2.4 to this specific DAG, we can compute $\nabla\text{Vit}_\Omega(\boldsymbol{\theta})$, $\langle\nabla\text{Vit}_\Omega(\boldsymbol{\theta}), \boldsymbol{Z}\rangle$ and $\nabla^2\text{Vit}_\Omega(\boldsymbol{\theta})\boldsymbol{Z}$ with the same complexity. The procedures are summarized in Algorithm 3 and Algorithm 4, respectively. From Proposition 2 property 1, $\text{Vit}_\Omega(\boldsymbol{\theta})$ is a convex function for any Ω.

B.3. Monotonic aligment prediction with the smoothed DTW

Computational graph. As illustrated in §3, the DAG contains a start node and $N_A N_B$ nodes. Therefore, $|\mathcal{V}| = N = N_A N_B + 1$. Due to the monotonic constraint, each node may only be connected with at most 3 other nodes. The cardinality of $\mathcal{Y}$ is the delannoy(N_A-1, N_B-1) number (**??**). That number grows exponentially with N_A and N_B.

Representation. We follow the notation of §3, *i.e.* we represent $\boldsymbol{Y}$ and $\boldsymbol{\theta}$ as $N_A \times N_B$ matrices. We represent $\boldsymbol{Y}$ as a binary matrix such that $y_{i,j} = 1$ if $\boldsymbol{a}_i$ is aligned with $\boldsymbol{b}_j$, and $y_{i,j} = 0$ otherwise. Likewise, we represent $\boldsymbol{\theta}$ as a real matrix such that $\theta_{i,j}$ is a measure of "discrepancy" between $\boldsymbol{a}_i$ and $\boldsymbol{b}_j$.

Algorithms. Following the DTW literature (**?**), we seek an alignment with *minimal* cost. For that reason, we introduce the smoothed min operator, its gradient and its Hessian as follows

$$\begin{aligned}
\min_\Omega(\boldsymbol{x}) &\triangleq -\max_\Omega(-\boldsymbol{x}) \\
\nabla\min_\Omega(\boldsymbol{x}) &= \nabla\max_\Omega(-\boldsymbol{x}) \\
\nabla^2\min_\Omega(\boldsymbol{x}) &= -\nabla^2\max_\Omega(-\boldsymbol{x}) \\
&= -\boldsymbol{J}_\Omega(\nabla\max_\Omega(-\boldsymbol{x})) \\
&= -\boldsymbol{J}_\Omega(\nabla\min_\Omega(\boldsymbol{x})).
\end{aligned}$$

Applying (6) to the DTW DAG gives rise to a smoothed version of the algorithm. Let $v_{i,j}(\boldsymbol{\theta})$ be the alignment cost up to cell (i,j). Then the smoothed DTW recursion is

$$v_{i,j}(\boldsymbol{\theta}) = \theta_{i,j} + \min_\Omega(v_{i,j-1}(\boldsymbol{\theta}), v_{i-1,j-1}(\boldsymbol{\theta}), v_{i-1,j}(\boldsymbol{\theta}))$$

The value $\text{DTW}_\Omega(\boldsymbol{\theta}) \triangleq \boldsymbol{v}_{N_A,N_B}(\boldsymbol{\theta})$ can be computed in $\mathcal{O}(N_A N_B)$ time. Applying the derivations of §2.3 and §2.4 to this specific DAG, we can compute $\nabla\text{DTW}_\Omega(\boldsymbol{\theta})$, $\langle\nabla\text{DTW}_\Omega(\boldsymbol{\theta}), \boldsymbol{Z}\rangle$ and $\nabla^2\text{DTW}_\Omega(\boldsymbol{\theta})\boldsymbol{Z}$ with the same complexity. The procedures, with appropriate handling of the edge cases, are summarized in Algorithm 5 and 6, respectively.

Algorithm 5 Compute $\mathrm{DTW}_\Omega(\boldsymbol{\theta})$ and $\nabla\mathrm{DTW}_\Omega(\boldsymbol{\theta})$

Input: Distance matrix $\boldsymbol{\theta} \in \mathbb{R}^{N_A \times N_B}$
▷ Forward pass
$v_{0,0} = 0$; $v_{i,0} = v_{0,j} = \infty$, $i \in [N_A], j \in [N_B]$
for $i \in [1, \dots, N_A], j \in [1, \dots, N_B]$ **do**
  $v_{i,j} = d_{i,j} + \min_\Omega(v_{i,j-1}, v_{i-1,j-1}, v_{i-1,j})$
  $\boldsymbol{q}_{i,j} = \nabla\min_\Omega(v_{i,j-1}, v_{i-1,j-1}, v_{i-1,j}) \in \mathbb{R}^3$
▷ Backward pass
$\boldsymbol{q}_{i,N_B+1} = \boldsymbol{q}_{N_A+1,j} = \mathbf{0}_3$, $i \in [N_A], j \in [N_B]$
$e_{i,N_B+1} = e_{N_A+1,j} = 0$, $i \in [N_A], j \in [N_B]$
$\boldsymbol{q}_{N_A+1,N_B+1} = (0, 1, 0)$; $e_{N_A+1,N_B+1} = 1$
for $j \in [N_B, \dots, 1], i \in [N_A, \dots, 1]$ **do**
  $e_{i,j} = q_{i,j+1,1}\, e_{i,j+1} + q_{i+1,j+1,2}\, e_{i+1,j+1} + q_{i+1,j,3}\, e_{i+1,j}$

Return: $\mathrm{DTW}_\Omega(\boldsymbol{\theta}) = v_{N_A,N_B}$
$\nabla\mathrm{DTW}_\Omega(\boldsymbol{\theta}) = (e)_{i,j=1}^{N_A,N_B}$
Intermediate computations for Algo. 6:
$\boldsymbol{Q} \triangleq (q)_{i,j,k=1}^{N_A+1,N_B+1,3}$; $\boldsymbol{E} \triangleq (e)_{i,j=1}^{N_A+1,N_B+1}$

Algorithm 6 Compute $\langle\nabla\mathrm{DTW}_\Omega(\boldsymbol{\theta}), \boldsymbol{Z}\rangle$, $\nabla^2\mathrm{DTW}_\Omega(\boldsymbol{\theta})\, \boldsymbol{Z}$

Input: $\boldsymbol{\theta} \in \mathbb{R}^{N_A \times N_B}$, $\boldsymbol{Z} \in \mathbb{R}^{N_A \times N_B}$
Call Algo. 5 with input $\boldsymbol{\theta}$ to retrieve $\boldsymbol{Q}$ and $\boldsymbol{E}$
▷ Forward pass
$\dot{v}_{i,0} = \dot{v}_{0,j} = 0$, $i \in [0, \dots, N_A], j \in [N_B]$
for $i \in [1, \dots, N_B], j \in [1, \dots, N_A]$ **do**
  $\dot{v}_{i,j} = z_{i,j} + q_{i,j,1}\, \dot{v}_{i,j-1} + q_{i,j,2}\, \dot{v}_{i-1,j-1} + q_{i,j,3}\, \dot{v}_{i-1,j}$
  $\dot{\boldsymbol{q}}_{i,j} = -\boldsymbol{J}_\Omega(\boldsymbol{q}_{i,j})\, (\dot{v}_{i,j-1}, \dot{v}_{i-1,j-1}, \dot{v}_{i-1,j}) \in \mathbb{R}^3$
▷ Backward pass
$\dot{\boldsymbol{q}}_{i,N_B+1} = \dot{\boldsymbol{q}}_{N_A+1,j} = \mathbf{0}_3$, $i \in [0, \dots, N_A], j \in [N_B]$
$\dot{e}_{i,N_B+1} = \dot{e}_{N_A+1,j} = 0$, $i \in [0, \dots, N_A], j \in [N_B]$
for $j \in [N_B, \dots, 1], i \in [N_A, \dots, 1]$ **do**
  $\dot{e}_{i,j} = \dot{q}_{i,j+1,1}\, e_{i,j+1} + q_{i,j+1,1}\, \dot{e}_{i,j+1} + \dot{q}_{i+1,j+1,2}\, e_{i+1,j+1} + q_{i+1,j+1,2}\, \dot{e}_{i+1,j+1} + \dot{q}_{i+1,j,3}\, e_{i+1,j} + q_{i+1,j,3}\, \dot{e}_{i+1,j}$

Return: $\langle\nabla\mathrm{DTW}_\Omega(\boldsymbol{\theta}), \boldsymbol{Z}\rangle = \dot{v}_{N_A,N_B}$
$\nabla^2\mathrm{DTW}_\Omega(\boldsymbol{\theta})\, \boldsymbol{Z} = (\dot{e})_{i,j=1}^{N_A,N_B}$

Note that when Ω is the negative entropy, $\mathrm{DTW}_\Omega(\boldsymbol{\theta})$ is known as soft-DTW (**?**). While the DP computation of $\mathrm{DTW}_\Omega(\boldsymbol{\theta})$ and of its gradient were already known, the generalization to any strongly convex Ω and the computation of $\nabla^2\mathrm{DTW}_\Omega(\boldsymbol{\theta})\boldsymbol{Z}$ are new. From Proposition 2 property 1, $\mathrm{DTW}_\Omega(\boldsymbol{\theta})$ is a *concave* function of the discrepancy matrix $\boldsymbol{\theta}$ for any Ω. With respect to time-series, DTW_Ω is neither convex nor concave.

C. Experimental details and further results

We finally provide details on the architecture used in experiments, with additionnals figures.

C.1. Named entity recognition (section §4.2)

Our model extracts word embedding from a 300-dimensional lookup table concatenated with a 50-dimensional character embedding. This character embedding corresponds to the concatenation of the last hidden unit of a bi-directional character LSTM, as in **?**. Character embedding size is set to 50. A word LSTM then produces sentence-aware features for each word. This LSTM is bi-directional with 100-dimensional hidden units per direction. The final features $\boldsymbol{X}$ used to build the potential tensor $\boldsymbol{\theta}$ are thus 200-dimensional. Note that, in contrast with **?**:

- The look-up table is initialized with 300-dimensional embeddings from *FastText* (**?**), trained on Wikipedia corpus.
- We do not pad letters prior to feeding the character LSTM as it is not principled.
- We do not train the unknown word embedding as we found it had no effect.

We convert tags to the IOBES (Inside-Outside-Begin-End-Stop) scheme to build a richer Vit_Ω model than if we used the simpler IOB (Inside-Outside-Begin) scheme, that has a lower number of tags. We performed a small grid-search to select the step-size and batch-size used for optimization: $s \in \{0.005, 0.01, 0.02\}$, $b \in \{8, 32, 128\}$. For each language and each loss, we select the highest-scoring model on the validation set, and report the test score.

The model is strongly subject to overfitting using the convex surrogate loss and the log likelihood. We have to use a small batch size ($b = 8$) and vanilla SGD with large step size ($s = 0.01$) to avoid this overfitting issue. For all losses, accelerated stochastic optimizers have all lower generalization performance than SGD, as also noticed in (**?**) when using the

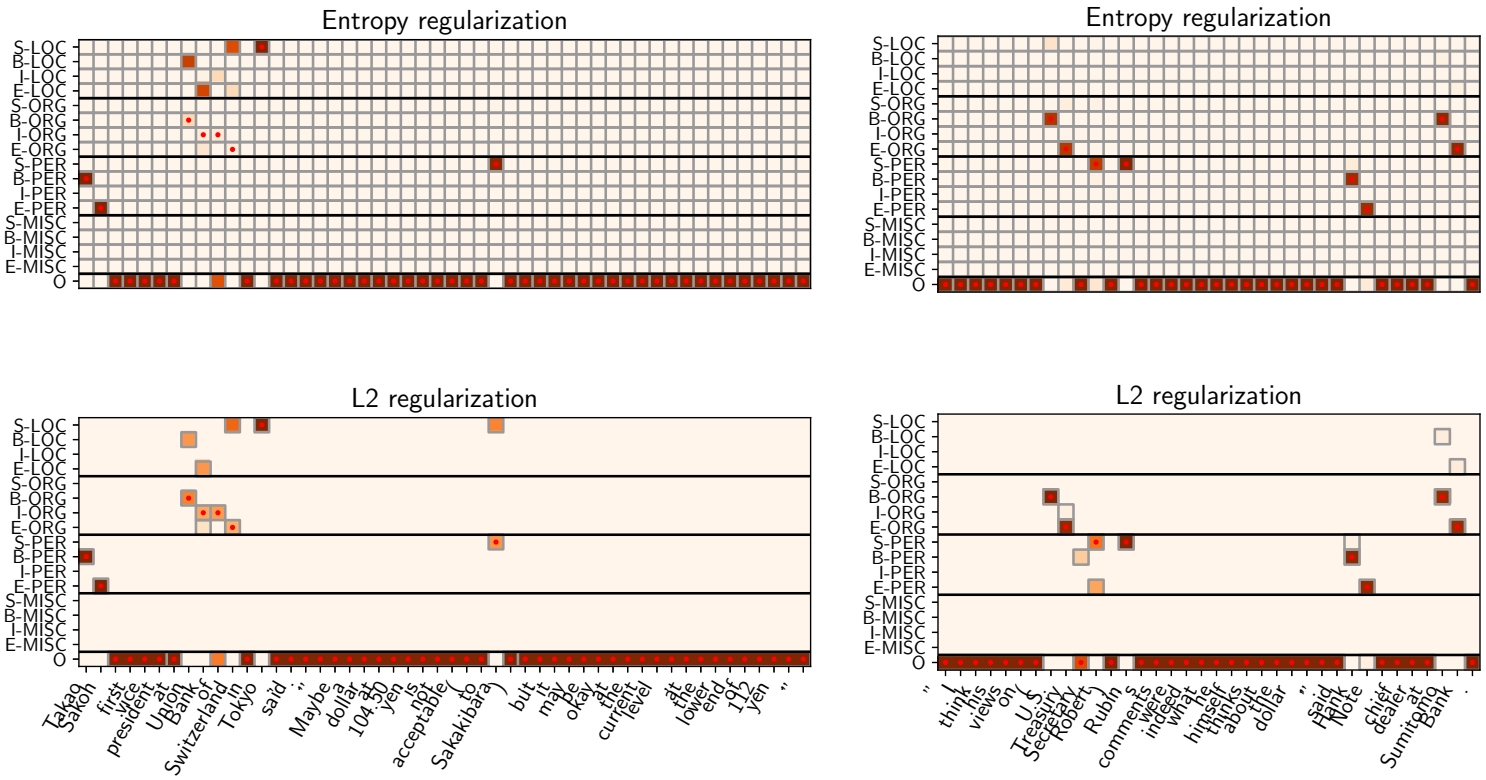

Figure 6. Test predictions from the entropy and ℓ_2^2 regularized named entity recognition (NER) models. Red dots indicate ground truth. When using ℓ_2^2 regularization, model predictions are sparse (grey borders indicates non-zero cells). They are thus easier to introspect for ambiguities, as we can list a finite number of possible outputs.

classical negative log-likelihood as a loss.

Visualization. The models using ℓ_2^2 regularization perform nearly on par with the ones using negentropy, as demonstrated in Table 1. On the other hand, ℓ_2^2 regularization leads to tag probability vectors that are sparse and hence easier to parse. They allow to detect ambiguities more easily. We display a few tagged English sequences in Figure 6. The model using ℓ_2^2 regularization correctly identifies an ambiguous entity (*Union Bank of Switzerland*) and can be used to propose two tag sequences: *(B-ORG, I-ORG, I-ORG, E-ORG)* or *(B-ORG, E-ORG, O, S-LOC)*. Probabilities of every tag sequence can be computed using the matrix $\boldsymbol{Q}$, as described in §2.3 — this remains tractable as long as the matrix $\boldsymbol{Q}$ is *sparse enough*, so that the number of non-zero probabilities sequence remains low. On the other hand, the model using negentropy regularization never assign a zero probability to any tag sequence — it is therefore not tractable to provide the user with a small set of interesting sequences.

C.2. Supervised audio-to-score transcription (section §4.3)

Audio sequences, sampled at 22.05 kHz, are split into frames of 512 samples. We extract the following features from these sequences: energy, spectral centroid, spectral bandwidth, and the 5 first MFCC features. All features are centered around the median and normalized.

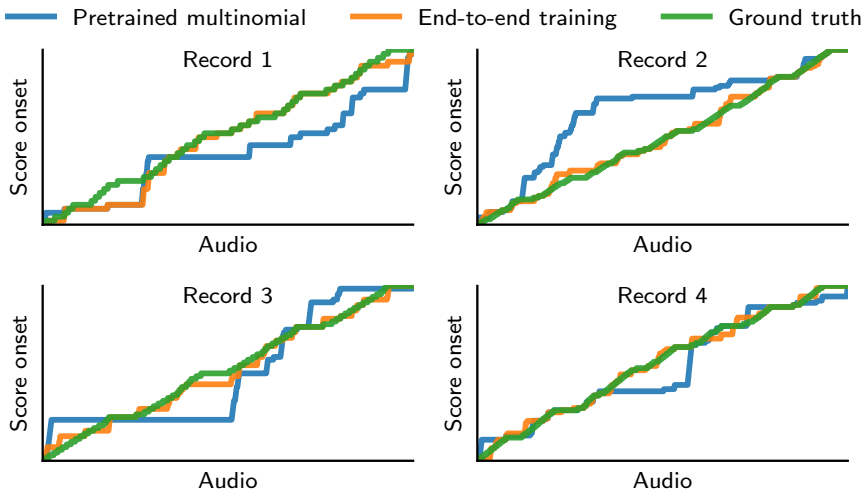

Figure 7. Alignment maps between score onsets and audio frames on test data from the Bach10 dataset. Our end-to-end trained model qualitatively performs better than the baseline model.

The ∇DTW_Ω layer is written in *Cython*[1], and hence run on CPU. This technical choice was suggested by the fact that we have to write explicit loops to specify the topological and reverse topological pass over the DTW computation graph (see Algorithm 5). However, it is possible to use only contiguous vector operations and thus take advantage of GPU computations — this is left for future work. We use *SciPy*'s[2] LBFGS-B solver to perform end-to-end training and multinomial regression. We use a ℓ_2^2 regularization on the weight $\boldsymbol{W}_{,:}$ we selected it using a grid search over $\{10^{-5}, 10^{-4}, \dots, 1\}$ and selected 10^{-3}.

Further vizualisation. In Figure 7, we display the alignment maps we obtained using our algorithm and using the baseline multinomial model followed by a hard-DTW alignment computation. These alignment maps correspond to the predicted onsets of keys. Our model (in orange) performs visibly better in predicting onsets.

C.3. Structured and sparse attention (section §5)

We use *OpenNMT-py* library[3] to fit our structured attention model. Model architecture and optimization details are as follow:

- We use a bidirectional LSTM encoder and decoder, with 500 units in each direction and a depth of 2 layers .
- The decoder is fed with the input representation as in **?**.
- SGD training with $s = 1$ learning rate, decaying from epoch 8 to epoch 15 with rate 0.65, batch size of size 256.
- Training sentence of lengths superior to 50 are ignored, and translated sentence are forced to a length inferior to 100.
- The temperature parameter is set to $\gamma = 2$ for entropy, and $\gamma = 10$ for ℓ_2^2. Performance is not affected much by this parameter, provided that it is not set too low in the ℓ_2^2 case — with a too small γ, Vit_Ω reduces to unregularized MAP estimation and ∇Vit_Ω has zero derivatives.

We use a 1-million sentence subject of WMT14 English-to-French corpus, available at

[1] http://cython.org/
[2] http://scipy.org/
[3] http://opennmt.net/

Table 3. Detokenized BLEU score on newstest2014 data using regularized and unregularized attention.

Attention model	WMT14 1M fr→en	WMT14 en→fr
Softmax	**27.96**	**28.08**
Entropy regularization	**27.96**	27.98
ℓ_2^2 reg.	27.21	27.28

http://nmt-benchmark.net/. We use Moses tokenizer and do not perform any post-processing, before computing BLEU score on detokenized sentences (*multi_bleu.perl* script).

Implementation. We implemeted a batch version of the $\nabla \mathrm{Vit}_\Omega$ layer on GPU, using the *PyTorch* tensor API. Model with negentropy-regularized attention mechanism runs $1/2$ as fast as the softmax attention mechanism (approximately 7500 tokens/s vs 15000 tokens/s on a single Nvidia Titan X Pascal). With ℓ_2^2 regularization, it is only $1/3$ as fast: approximately 5000 tokens/s. Although this remains reasonable, it could certainly be optimized by rewriting kernels using lower-level languages (*e.g.,* using *ATen* API from *PyTorch*.)

Further results. Table 3 provides BLEU scores for both translation directions on the 1 million sentence subset of WMT14 we used. We observe that the introduction of structure and sparsity does not hinder the general performance of the model. We provide several examples of attention maps in Figure 8, that illustrate the sparsity patterns ℓ_2^2 regularization uncovers.

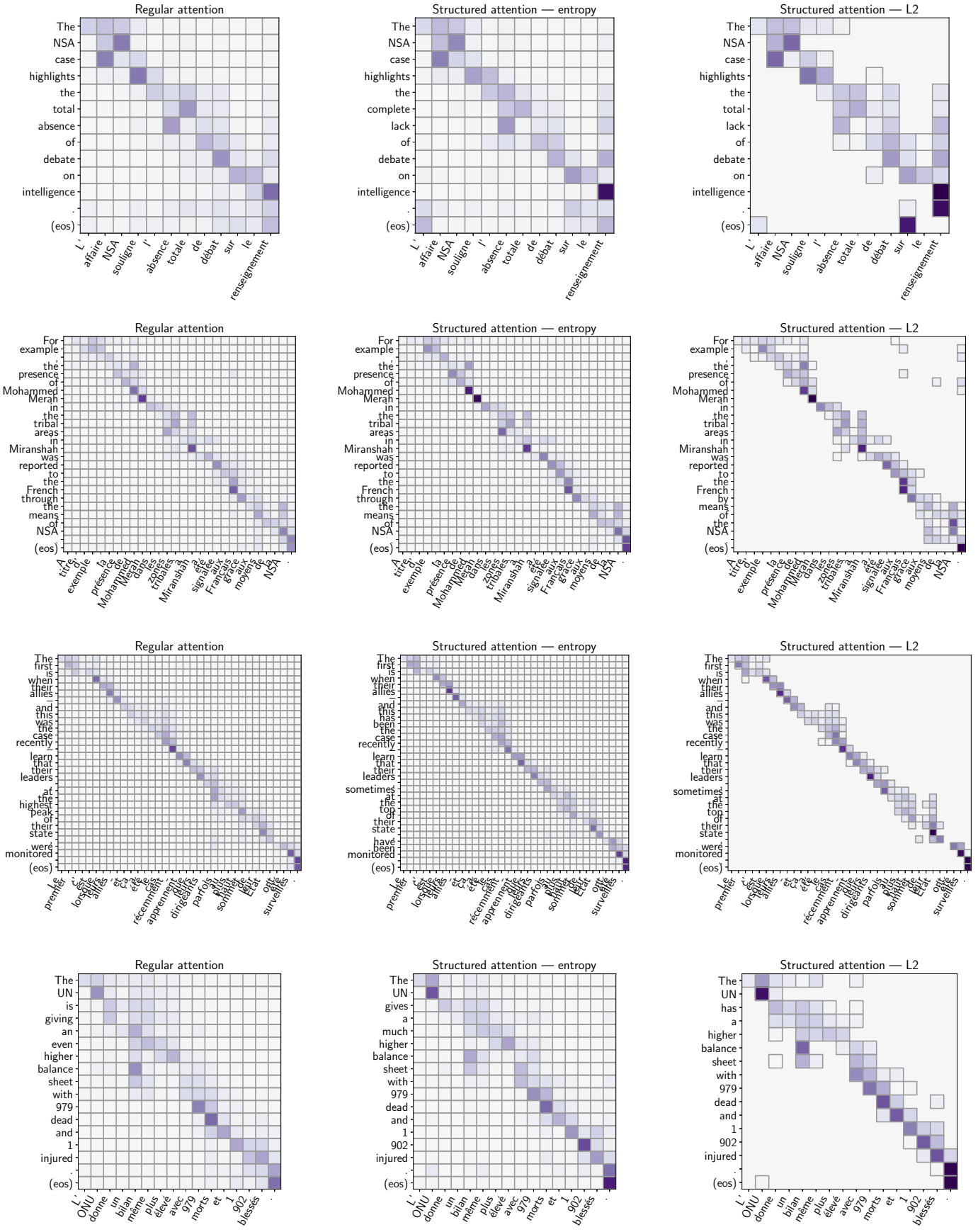

Figure 8. Attention on test samples from Newstest2014. Borders indicate non-zero cells. Translations (y-axis) are often qualitatively equivalent, while attentions maps are sparse in the ℓ_2^2 case.