Hashing as Tie-Aware Learning to Rank
Supplementary Material

A. Proof of Proposition 1

Proof. Our proof essentially restates the results in [3] using our notation. In [3], a *tie-vector* $T = (t_0, \dots, t_{d+1})$ is defined, where $t_0 = 0$ and the next elements indicate the ending indices of the equivalence classes in the ranking, *e.g.* t_1 is the ending index of $R^{(0)}$, and so on. Using our notation, we can see that $R^{(d)} = (R_{1+t_d}, \dots, R_{t_{d+1}})$, and $t_d = N_{d-1} = \sum_{j=0}^{d-1} n_j$.

We first consider $\mathrm{AP_T}$. In Section 2.4 of [3], the tie-aware AP at cutoff k is defined as

$$\mathrm{AP_T}@k(R) = \frac{\sum_{j=1}^{k} \frac{n_i^+}{n_i} \left(N_{i-1}^+ + (j - t_i - 1)\frac{n_i^+ - 1}{n_i - 1} + 1 \right) \frac{1}{j}}{\sum_{j=1}^{|S|} \mathcal{A}_q(j)}, \tag{1}$$

where i is the index of the tie that item j is in. To derive $\mathrm{AP_T}$ in our formulation, we take k to be the maximum possible cutoff $|S|$, and substitute by definition $N^+ = \sum_{j=1}^{|S|} \mathcal{A}_q(j), t_i = N_{i-1}$:

$$\mathrm{AP_T}(R) = \frac{1}{N^+} \sum_{j=1}^{|S|} \frac{\frac{n_i^+}{n_i} \left(N_{i-1}^+ + (j - N_{i-1} - 1)\frac{n_i^+ - 1}{n_i - 1} + 1 \right)}{j}. \tag{2}$$

It is clear that this sum decomposes additively over j. Therefore, we can explicitly compute the contribution from items in a single tie $R^{(d)}$,

$$\mathrm{AP_T}(R^{(d)}) = \frac{1}{N^+} \sum_{j=N_{d-1}+1}^{N_d} \frac{\frac{n_d^+}{n_d} \left(N_{d-1}^+ + (j - N_{d-1} - 1)\frac{n_d^+ - 1}{n_d - 1} + 1 \right)}{j}, \tag{3}$$

and this gives (6) in the paper.

Next, tie-aware DCG is given in Section 2.6 of [3] as

$$\mathrm{DCG_T}@k(R) = \sum_d \left[\left(\frac{1}{n_d} \sum_{j=t_d+1}^{t_{d+1}} G(\mathcal{A}_q(j)) \right) \sum_{j=t_d+1}^{\min(t_{d+1},k)} D(j) \right]. \tag{4}$$

Again, we consider a single tie $R^{(d)}$, take $k = |S|$, and make the substitution $t_d = N_{d-1}$:

$$\mathrm{DCG_T}(R^{(d)}) = \frac{1}{n_d} \sum_{j \in R^{(d)}} G(\mathcal{A}_q(j)) \sum_{j=N_{d-1}+1}^{N_d} D(j) \tag{5}$$

$$= \frac{1}{n_d} \sum_{v \in \mathcal{V}} \overbrace{\sum_{j \in R^{(d)}} \mathbf{1}[v = \mathcal{A}_q(j)]}^{n_{d,v}} G(v) \sum_{j=N_{d-1}+1}^{N_d} D(j) \tag{6}$$

$$= \frac{1}{n_d} \sum_{v \in \mathcal{V}} G(v) n_{d,v} \sum_{j=N_{d-1}+1}^{N_d} D(j). \tag{7}$$

This completes the derivation for (7) in the paper.

□

B. Proof of Proposition 2

Proof. First, we denote the summand in the definition of $\mathrm{AP_T}$ (3) as $\beta_d(t)$, and rewrite it as

$$\beta_d(t) = \frac{N^+_{d-1} + (t - N_{d-1} - 1)\frac{n^+_d - 1}{n_d - 1} + 1}{t} = \frac{n^+_d - 1}{n_d - 1} + \frac{N^+_{d-1} + 1 - \frac{n^+_d - 1}{n_d - 1}(N_{d-1} + 1)}{t}. \tag{8}$$

It is of the form $A + B/t$ where A, B are constant in t. We proceed with the summation over t in (3):

$$\mathrm{AP_T}(R^{(d)}) = \frac{n^+_d}{n_d N^+} \sum_{t=N_{d-1}+1}^{N_d} \beta_d(t) \tag{9}$$

$$= \frac{n^+_d}{n_d N^+} \left[\frac{n^+_d - 1}{n_d - 1} n_d + \left(N^+_{d-1} + 1 - \frac{n^+_d - 1}{n_d - 1}(N_{d-1} + 1) \right) \sum_{t=N_{d-1}+1}^{N_d} \frac{1}{t} \right]. \tag{10}$$

The main obstacle in the continuous relaxation is the finite sum in (10), which has in its limits N_{d-1} and N_d, variables to be relaxed. However, it is a partial sum of the harmonic series, which can be well approximated by differences of the natural logarithm:

$$\sum_{t=N_{d-1}+1}^{N_d} \frac{1}{t} \approx \int_{N_{d-1}}^{N_d} \frac{dt}{t} = \ln(N_d) - \ln(N_{d-1}). \tag{11}$$

In fact, (11) corresponds to the *midpoint rule* in approximating definite integrals by finite sums, but is applied in the reverse direction. The relaxation of $\mathrm{AP_T}$ is then derived as follows:

$$\mathrm{AP_T}(R^{(d)}) \approx \frac{n^+_d}{n_d N^+} \left[\frac{n^+_d - 1}{n_d - 1} n_d + \left(N^+_{d-1} + 1 - \frac{n^+_d - 1}{n_d - 1}(N_{d-1} + 1) \right) \ln \frac{N_d}{N_{d-1}} \right] \tag{12}$$

$$\Rightarrow \ \mathrm{AP_r}(R^{(d)}) = \frac{c^+_d(c^+_d - 1)}{(c_d - 1)N^+} + \frac{c^+_d}{c_d N^+} \left[C^+_{d-1} + 1 - \frac{c^+_d - 1}{c_d - 1}(C_{d-1} + 1) \right] \ln \frac{C_d}{C_{d-1}}. \tag{13}$$

Note that $N^+ = \sum_d n^+_d$ is a constant for a fixed query and fixed database, thus it is not affected by the relaxation.

Next, we consider $\mathrm{DCG_T}$, where the sum of logarithmic[1] discount values similarly involves variables to be relaxed in its limits. Thus, we employ the same approximation strategy using continuous integrals.

$$\sum_{t=N_{d-1}+1}^{N_d} D(t) = \sum_{t=N_{d-1}+1}^{N_d} \frac{1}{\log_2(t+1)} \approx \int_{N_{d-1}}^{N_d} \frac{dt}{\log_2(t+1)} = \ln 2 \int_{N_{d-1}+1}^{N_d+1} \frac{dt}{\ln t}. \tag{14}$$

Combining with the definition of $\mathrm{DCG_T}$, we get its continuous relaxation:

$$\mathrm{DCG_r}(R^{(d)}) = \ln 2 \sum_{v \in \mathcal{V}} \frac{G(v) c_{d,v}}{c_d} \int_{C_{d-1}+1}^{C_d+1} \frac{dt}{\ln t} \tag{15}$$

$$= \ln 2 \sum_{v \in \mathcal{V}} \frac{G(v) c_{d,v}}{c_d} \left[Li(C_d + 1) - Li(C_{d-1} + 1) \right] \tag{16}$$

where Li is the logarithmic integral function: $Li(x) = \int_0^x \frac{dx}{\ln x}$.

□

C. Approximation Error Analysis

We now analyze the approximation error when doing the continuous relaxations. We take $\mathrm{AP_T}$ as example, and note that the analysis for $\mathrm{DCG_T}$ is similar.

[1] Other types of discounts are also used in the literature, including linear discount: $D(t) \propto \frac{1}{t}$. It is easy to see that our technique also applies.

The continuous relaxation for $\mathrm{AP}_\mathrm{T}(R^{(d)})$ is given in (11), which replaces a finite sum with a defnite integral, where the finite sum has $N_d - N_{d-1} = n_d$ summands. First, we consider the case where there are no ties, or $n_d \in \{0, 1\}$, *i.e.* the d-th histogram bin is either empty or contains a single item. In this case, we can directly evaluate the lefthand side sum in (11) to be either 0 or $\frac{1}{N_d}$, without using the integral approximation. Therefore, when there are no ties, there is no approximation error.

Next we consider $n_d \geq 2$. Let the N-th harmonic number be $H(N) = \sum_{t=1}^{N} \frac{1}{t}$, then the lefthand side of (11) is exactly $H(N_d) - H(N_{d-1})$. It is well known that the harmonic number can be closely approximated as

$$H(N) = \gamma + \ln(N) + \frac{1}{2N} + O\left(\frac{1}{12N^2}\right), \tag{17}$$

where $\gamma \approx 0.5772$ is Euler's constant. A direct application of this approximation gives the following:

$$H(N_d) = \gamma + \ln(N_d) + \frac{1}{2N_d} + O\left(\frac{1}{12N_d^2}\right) \tag{18}$$

$$H(N_{d-1}) = \gamma + \ln(N_{d-1}) + \frac{1}{2N_{d-1}} + O\left(\frac{1}{12N_{d-1}^2}\right) \tag{19}$$

$$\Rightarrow H(N_d) - H(N_{d-1}) = \ln(N_d) - \ln(N_{d-1}) + O\left(\frac{1}{2N_{d-1}} - \frac{1}{2N_d}\right). \tag{20}$$

Comparing (20) with (11), we see that the approximation error is

$$O\left(\frac{1}{2N_{d-1}} - \frac{1}{2N_d}\right) = O\left(\frac{n_d}{2N_{d-1}N_d}\right) = O\left(\frac{n_d}{2N_{d-1}(N_{d-1} + n_d)}\right) = O\left(\frac{n_d}{2N_{d-1}^2}\right). \tag{21}$$

The error is proportional to n_d, the number of items in the d-th bin in the Hamming distance histogram. However, even if n_d is large, the error is in general still small, since it has N_{d-1}^2 in the denominator. Note that (11) can be further tightened by including the $\frac{1}{2N}$ term, or even higher order terms in the approximation of Harmonic numbers, but the approximation using the first two terms (Euler's constant and natural log) is already quite tight, and is in fact used widely.

D. Tightening the $\tanh$ Relaxation

As is common among relaxation-based hashing methods, we relax the binary constraints by replacing the discontinuous sgn function with $\tanh$. With this simple relaxation, the performance gains are mainly due to optimizing the proposed objectives. Nevertheless, it is conceivable that the continuation method, *e.g.* as employed by HashNet [2], can tighten the relaxation and lead to better results. To provide a concrete example, below we report improved results for TALR-AP on CIFAR-10 (setting 1), when we increase the scaling α in $\tanh$ over time, instead of using a fixed value.

AP_T	12 bits	24 bits	32 bits	48 bits
$\alpha = 40$	0.732	0.789	0.800	0.826
$\alpha \to \infty$	0.751	0.804	0.813	0.830

E. Efficient Minibatch Backpropagation

As mentioned in Sec. 4.2 in the paper, our models are trained using minibatch SGD. To maximally utilize supervision, we use the following strategy: each example in the minibatch is used to query the rest of the batch (which acts as the database), and the resulting objective values are averaged. Here, we detail the derivation of the backpropagation rules according to this formulation. We take inspirations from [1] and [4], both of which similarly average listwise objectives designed for binary affinities. Our technique can be seen as a direct generalization of [1] in that we support multi-level affinities.

Consider a minibatch of size M, $\{x_1, \ldots, x_M\}$. We use a unified shorthand $\mathcal{O}_i$ to denote the (relaxed) objective value when x_i is the query, which can either be AP_r or DCG_r in our formulation. The overall minibatch objective is then $\mathcal{O} = \frac{1}{M} \sum_{i=1}^{M} \mathcal{O}_i$. For the entire minibatch, we group the relaxed hash mapping output into a $b \times M$ matrix,

$$\hat{\mathbf{\Phi}} = \left[\hat{\Phi}(x_1)\ \hat{\Phi}(x_2)\ \cdots\ \hat{\Phi}(x_M)\right] \in \mathbb{R}^{b \times M}. \tag{22}$$

We consider the multi-level affinity setup where affinity values are from a finite set $\mathcal{V}$, which includes binary affinities as a special case, *i.e.* when $\mathcal{V} = \{0, 1\}$. The Jacobian of the minibatch objective with respect to $\hat{\mathbf{\Phi}}$ can be written as

$$\frac{\partial \mathcal{O}}{\partial \hat{\mathbf{\Phi}}} = \frac{1}{M} \sum_{i=1}^{M} \frac{\partial \mathcal{O}_i}{\partial \hat{\mathbf{\Phi}}} = \frac{1}{M} \sum_{i=1}^{M} \sum_{d=0}^{b} \sum_{v \in \mathcal{V}} \frac{\partial \mathcal{O}_i}{\partial c_{d,v}^{(i)}} \frac{\partial c_{d,v}^{(i)}}{\partial \hat{\mathbf{\Phi}}}, \tag{23}$$

where as defined in Sec. 4.1, $c_{d,v}^{(i)}$ is the continuous relaxation of $n_{d,v}^{(i)}$, the d-th bin in the distance histogram conditioned on affinity level v. The superscript $^{(i)}$ indicates that the current query is x_i.

Evaluating this Jacobian involves two steps. First, we need to compute the partial derivative $\partial \mathcal{O}_i / \partial c_{d,v}^{(i)}, \forall d, \forall v$. Note that this is exactly the differentiation of $\mathrm{AP_r}$ and $\mathrm{DCG_r}$, and as we pointed out in Sec. 4.1, both can be evaluated in closed form. We use variables α to denote these partial derivatives, omitting the details of derivation:

$$\alpha_{d,v}^{(i)} = \frac{\partial \mathcal{O}_i}{\partial c_{d,v}^{(i)}}. \tag{24}$$

Next, we need to evaluate the Jacobian $\partial c_{d,v}^{(i)} / \partial \hat{\mathbf{\Phi}}$, which is essentially differentiating the soft histogram binning process. Let us consider each column of this Jacobian. First, for $\forall j \neq i$, using chain rule,

$$\frac{\partial c_{d,v}^{(i)}}{\partial \hat{\Phi}(x_j)} = \frac{\partial c_{d,v}^{(i)}}{\partial \hat{d}_\Phi(x_i, x_j)} \frac{\partial \hat{d}_\Phi(x_i, x_j)}{\partial \hat{\Phi}(x_j)} = \mathbf{1}[\mathcal{A}_i(j) = v] \frac{\partial \delta(\hat{d}_\Phi(x_i, x_j), d)}{\partial \hat{d}_\Phi(x_i, x_j)} \frac{\partial \hat{d}_\Phi(x_i, x_j)}{\partial \hat{\Phi}(x_j)} \tag{25}$$

$$= \mathbf{1}[\mathcal{A}_i(j) = v] \delta_d'(\hat{d}_\Phi(x_i, x_j)) \frac{-\hat{\Phi}(x_i)}{2} \tag{26}$$

$$\triangleq \beta_{d,v}(i, j) \frac{-\hat{\Phi}(x_i)}{2}, \tag{27}$$

where δ_d' is the derivative of the single-argument function $\delta(\cdot, d)$, and we define the shorthand

$$\beta_{d,v}(i, j) = \mathbf{1}[\mathcal{A}_i(j) = v] \delta_d'(\hat{d}_\Phi(x_i, x_j)). \tag{28}$$

Note that β is symmetric, *i.e.* $\beta_{d,v}(i, j) = \beta_{d,v}(j, i)$, which follows from the fact that both the affinity $\mathcal{A}$ and the distance function $\hat{d}_\Phi$ are symmetric.

For $j = i$, we have a similar result:

$$\frac{\partial c_{d,v}^{(i)}}{\partial \hat{\Phi}(x_i)} = \sum_{k \neq i} \frac{\partial c_{d,v}^{(i)}}{\partial \hat{d}_\Phi(x_i, x_k)} \frac{\partial \hat{d}_\Phi(x_i, x_k)}{\partial \hat{\Phi}(x_i)} = \sum_{k \neq i} \beta_{d,v}(i, k) \frac{-\hat{\Phi}(x_k)}{2}. \tag{29}$$

To unify these two cases, we require that $\beta_{d,v}(i, i) \equiv 0, \forall i$. We now have a unified expression for the j-th column in the Jacobian $\partial c_{d,v}^{(i)} / \partial \hat{\mathbf{\Phi}}$:

$$\frac{\partial c_{d,v}^{(i)}}{\partial \hat{\Phi}(x_j)} = -\frac{1}{2} \beta_{d,v}(i, j) \hat{\Phi}(x_i) - \frac{\mathbf{1}[j = i]}{2} \sum_{k=1}^{M} \beta_{d,v}(i, k) \hat{\Phi}(x_k). \tag{30}$$

We now obtain a compact matrix form for $\partial c_{d,v}^{(i)} / \partial \hat{\mathbf{\Phi}}$. Let $\boldsymbol{\beta}_{d,v}^{(i)} = (\beta_{d,v}(i, 1), \ldots, \beta_{d,v}(i, M)) \in \mathbb{R}^M$. Also, let $\boldsymbol{e}_i$ be the i-th standard basis vector in $\mathbb{R}^M$, *i.e.* the i-th element is 1 and all others are 0. We have that

$$\frac{\partial c_{d,v}^{(i)}}{\partial \hat{\mathbf{\Phi}}} = -\frac{1}{2} \hat{\Phi}(x_i) (\boldsymbol{\beta}_{d,v}^{(i)})^\top - \left[\sum_{k=1}^{M} \frac{1}{2} \beta_{d,v}(i, k) \hat{\Phi}(x_k) \right] \boldsymbol{e}_i^\top = -\frac{1}{2} \left[\hat{\Phi}(x_i) (\boldsymbol{\beta}_{d,v}^{(i)})^\top + \hat{\mathbf{\Phi}} \boldsymbol{\beta}_{d,v}^{(i)} \boldsymbol{e}_i^\top \right]. \tag{31}$$

Finally, we complete (23) using the result above. The main trick is to change the ordering of sums: we bring the sum over $i = 1, \ldots, M$ inside,

$$\frac{\partial \mathcal{O}}{\partial \hat{\mathbf{\Phi}}} = \frac{1}{M} \sum_{i=1}^{M} \sum_{d=0}^{b} \sum_{v \in \mathcal{V}} \frac{\partial \mathcal{O}_i}{\partial c_{d,v}^{(i)}} \frac{\partial c_{d,v}^{(i)}}{\partial \hat{\mathbf{\Phi}}} \tag{32}$$

$$= \frac{1}{M} \sum_{d=0}^{b} \sum_{v \in \mathcal{V}} \sum_{i=1}^{M} -\frac{1}{2} \alpha_{d,v}^{(i)} \left[\hat{\Phi}(x_i) (\boldsymbol{\beta}_{d,v}^{(i)})^\top + \hat{\mathbf{\Phi}} \boldsymbol{\beta}_{d,v}^{(i)} \boldsymbol{e}_i^\top \right] \tag{33}$$

$$= -\frac{1}{2M} \sum_{l=0}^{b} \sum_{v \in \mathcal{V}} \left[\sum_{i=1}^{M} \alpha_{d,v}^{(i)} \hat{\Phi}(x_i) (\boldsymbol{\beta}_{d,v}^{(i)})^\top + \hat{\mathbf{\Phi}} \sum_{i=1}^{M} \alpha_{d,v}^{(i)} \boldsymbol{\beta}_{d,v}^{(i)} \boldsymbol{e}_i^\top \right]. \tag{34}$$

To further simplify this result, we define two $M \times M$ matrices:

$$A_{d,v} = \operatorname{diag}(\alpha_{d,v}^{(1)}, \ldots, \alpha_{d,v}^{(M)}) \in \mathbb{R}^{M \times M}, \tag{35}$$

$$B_{d,v} = \left[\boldsymbol{\beta}_{d,v}^{(1)} \cdots \boldsymbol{\beta}_{d,v}^{(M)} \right] = \begin{bmatrix} \beta_{d,v}(1,1) & \beta_{d,v}(2,1) & \cdots & \beta_{d,v}(M,1) \\ \beta_{d,v}(1,2) & \beta_{d,v}(2,2) & \cdots & \beta_{d,v}(M,2) \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{d,v}(1,M) & \beta_{d,v}(2,M) & \cdots & \beta_{d,v}(M,M) \end{bmatrix} \in \mathbb{R}^{M \times M}. \tag{36}$$

Then, we arrive at the following simplification of (34) and (23),

$$\frac{\partial \mathcal{O}}{\partial \hat{\mathbf{\Phi}}} = -\frac{1}{2M} \sum_{d=0}^{b} \sum_{v \in \mathcal{V}} \left[\hat{\mathbf{\Phi}} A_{d,v} (B_{d,v})^\top + \hat{\mathbf{\Phi}} B_{d,v} A_{d,v} \right] = -\frac{\hat{\mathbf{\Phi}}}{2M} \sum_{d=0}^{b} \sum_{v \in \mathcal{V}} \left(A_{d,v} B_{d,v} + B_{d,v} A_{d,v} \right). \tag{37}$$

Note that we have used the fact that $B_{d,v}$ is a symmetric matrix (36), which is because β is symmetric, as mentioned earlier. This operation can be implemented efficiently using only matrix multiplications and additions. Also, since $A_{d,v}$ is a diagonal matrix, multiplying it with $B_{d,v}$ essentially scales the rows or columns of $B_{d,v}$, which is an $O(M^2)$ operation as opposed to $O(M^3)$ as in general matrix multiplication. The entire time complexity is therefore $O(b|\mathcal{V}|M^2)$.

At this point, we have completed the differentiation of the minibatch objective $\mathcal{O}$ with respect to the relaxed hash mapping output, $\hat{\mathbf{\Phi}}$. Further backpropagation is straightforward, since $\hat{\mathbf{\Phi}}$ is obtained by applying a pointwise $\tanh$ function on the raw activations from the previous layer.

References

[1] Fatih Cakir, Kun He, Sarah Adel Bargal, and Stan Sclaroff. Hashing with mutual information. *arXiv preprint arXiv:1803.00974*, 2018.

[2] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. HashNet: Deep learning to hash by continuation. In *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017.

[3] Frank McSherry and Marc Najork. Computing information retrieval performance measures efficiently in the presence of tied scores. In *Proc. European Conference on Information Retrieval*, 2008.

[4] Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. Few-shot learning through an information retrieval lens. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2252–2262, 2017.