Autoregressive Ranking: Bridging the Gap Between Dual and Cross Encoders

Benjamin Rozonoyer[1*] Chong You[2] Michael Boratko[2] Himanshu Jain[2]
Nilesh Gupta[3*] Srinadh Bhojanapalli[2] Andrew McCallum[2] Felix Yu[2]

[1]University of Massachusetts Amherst [2]Google DeepMind [3]The University of Texas at Austin

Abstract

The success of Large Language Models (LLMs) has motivated a shift toward generative approaches to retrieval and ranking, aiming to supersede classical Dual Encoders (DEs) and Cross Encoders (CEs). A prominent paradigm is pointwise Autoregressive Ranking (ARR), where an LLM generates document identifiers (docIDs) token-by-token to enable ranking via beam search. ARR offers the promise of superior expressivity compared to DEs while avoiding the prohibitive computational cost of CEs. However, a formal theoretical foundation for this expressive power has been missing. Moreover, the standard next-token prediction loss is rank-agnostic and inappropriate for finetuning an LLM for ranking tasks. In this paper, we first prove that, under an infinite-capacity assumption, the expressive capacity of ARR is strictly superior to that of DEs. While a DE requires an embedding dimension that grows linearly with corpus size to achieve arbitrary rankings, ARR can solve it with a constant hidden dimension. We then propose a generalized rank-aware training loss for LLM finetuning. By using item-level reweighting and prefix-tree marginalization, we distribute probability mass over valid docID tokens based on their ground-truth relevance. Experiments on WordNet and ESCI datasets show that our loss suppresses invalid docID generations and improves ranking metrics beyond top-1 retrieval.

1. Introduction

Ranking documents by their relevance to a user query is a cornerstone of modern information retrieval. The dominant paradigm for this task is a multi-stage pipeline. The first stage, retrieval, aims to efficiently select a candidate set of documents from a massive corpus. This is typically accomplished using **Dual Encoder (DE)** models, which represent queries and documents as dense vectors and use fast approximate nearest neighbor (ANN) search to find the top matches (Karpukhin et al., 2020; Guo et al., 2020). While efficient, the expressive power of DEs is inherently limited, as the rich interaction between a query and a document is compressed into a single vector similarity. Consequently, a second reranking stage is often required. In this stage, a more powerful but computationally expensive **Cross Encoder (CE)** model jointly processes the query and each candidate document to produce a more accurate relevance score (Nogueira & Cho, 2019). CEs are infeasible for first-stage retrieval due to their prohibitive cost, which scales linearly with corpus size.

The success of LLMs has inspired a new, unified approach to retrieval and ranking in lieu of the two-stage approach (Metzler et al., 2021; Li et al., 2023a; Lee et al., 2024a). In this paradigm, an LLM is trained to function as a generative retrieval model, autoregressively decoding the relevant documents or their identifiers (docIDs) token-by-token, conditioned on the query (Tay et al., 2022). For ranking, this model can generate a sorted list of documents by employing beam search, effectively ranking documents by their conditional generation probability. This approach, which we dub **Autoregressive Ranking (ARR)**, offers several compelling advantages over the traditional DE/CE pipeline. It collapses the two-stage process into a single model, eliminating the need for a separate ANN index. Furthermore, it is more efficient than CEs as it does not require scoring every document individually, while preserving CEs' cross-attention over already generated docID tokens. Crucially, it has also been argued that *by generating documents token-by-token*, ARR models can capture deeper and more nuanced query-document interactions than DEs (Yuan et al., 2024); while intuitive, the evidence for this has been primarily empirical. In this paper, we address two

[*]Work done during an internship at Google Research.

Accepted to FoGen 2026: Foundations of Deep Generative Models: Understanding Memorization, Generalization, and Reasoning, an ICML 2026 workshop (non-archival).

critical gaps in the understanding and optimization of ARR models for ranking:

- **We provide a theoretical foundation for the superior expressive capacity of ARR over DEs.** A rigorous analysis of the embedding geometry required for ranking shows that for a DE to achieve any ordering of k documents, its embedding dimension must grow linearly with k. In contrast, we prove that an ARR model with constant hidden dimension is theoretically sufficient to rank an arbitrary number of documents. This offers a formal explanation for the advantages of ARR, under an infinite-capacity assumption which we make explicit in Section 3.
- **We introduce a rank-aware generalization of next-token prediction loss for enhanced ranking capabilities.** Most existing generative retrieval methods are trained with standard next-token prediction, an objective which fails to explicitly model the relative ordering of documents, which is crucial for ranking. We introduce a new loss function designed to make the LLM aware of the entire ranked list during training, while keeping the generation and signal length on the order of a single docID rather than a full list, enabling ranking via beam search. We test our method on WordNet and a shopping queries dataset, demonstrating improved ranking performance relative to rank-agnostic next-token prediction.

2. Related Work

Classical IR Losses Training objectives for retrieval and ranking can be broadly grouped into the categories of *pointwise*, *pairwise*, and *listwise* (Liu et al., 2009). A *pointwise* loss as in Chen et al. (2009) of the form $\mathcal{L}(q, d)$ encourages the model to assign high scores to more relevant items and vice versa; each training example consists of a single document to penalize independently of the rest. *Pairwise* loss functions (Cao et al., 2007) of the form $\mathcal{L}(q, d^+, d^-)$ strive to maximize the margin between relevant and irrelevant documents. *Listwise* losses such as (Xia et al., 2008) of the form $\mathcal{L}(q, d_1, \ldots, d_{n_q})$ jointly take as input the model scores of all documents under consideration for a given query, and can encourage the most flexible scoring configurations at the granularity of permutations rather than individual items and commonly rely on the Plackett-Luce probability model over permutations (Luce et al., 1959; Plackett, 1975).

Generative Reranking Autoregressive decoder LLMs introduce the facet of text generation on top of the simple API of "model-as-document-scorer". The score-based loss formulations discussed above remain applicable insofar as a docID's probability can be obtained from the probabilities of its constituent tokens; notable LLM-based reranking methods (Sachan et al., 2022; Drozdov et al., 2023) rely specifically on LLM-based log-likelihood to obtain scores. Meanwhile, the terminology of "pointwise" vs "listwise" naturally acquires a second meaning, i.e., whether the text string which the LLM generates comprises a single docID or a ranked list of docIDs (separated by a delimiter and in descending order, as per common convention). Our "pointwise" approach relates to both these senses (scoring just one document, or generating just a single docID string), since steering LLM probabilities to be more rank-aware also directly impacts the rankings produced by a decoding strategy like beam search at inference time. *The generative approach naturally allows us to perform ranking with a single model call, in contrast to CEs which evaluate the score for every document separately. The compact pointwise output format mitigates snowballing errors and is efficient to train under our rank-aware modification to next-token prediction.*

A handful of examples illustrate the diverse reranking paradigms afforded by generative LLMs. RankVicuna (Pradeep et al., 2023a) and RankZephyr (Pradeep et al., 2023b) explore training/distillation choices, using instruction tuning with next token prediction, to achieve a robust open-source zero-shot LLM for listwise ranking. LTRGR (Li et al., 2024), on top of a next-token-prediction-based "learning-to-generate" phase, introduces a "learning-to-rank" phase that maximizes the margin between the autoregressively-computed scores of relevant and irrelevant passage identifiers. RecRanker (Luo et al., 2024) trains an LLM with instruction tuning on a combination of pointwise, pairwise, and listwise ranking objectives for top-k sequential recommendation, and at inference time aggregates the three types of responses into a hybrid ranking by means of respective utility functions. GenRank (Huang et al., 2025) identify the autoregressive generation mechanism as critical to the effectiveness of generative ranking, more so than the pre-training paradigm. (Sun et al., 2023) explore the ranking capabilities of ChatGPT and GPT-4 and propose a distillation technique to learn smaller specialized rankers. Gupta et al. (2025) introduce BlockRank, efficiently tailoring the attention mechanism to in-context ranking. *We argue that autoregressive methods which employ next-token prediction or instruction fine-tuning are relying on a token-level loss which is fundamentally not rank-aware. Our proposed loss function is a simple, principled and effective injection of rank-awareness into next-token prediction — a loss function which is ubiquitous to autoregressive models.*

Parametric Generative Retrieval and DocID Creation Differentiable Search Index (DSI) (Tay et al., 2022) instills the docID index in model weights as part of the training objective. A critical component of parametric generative retrieval is effective docID design (Li et al., 2023b) — we take up this question for the ESCI dataset in Section 5.1. Note that we opt to use in-context ranking (Gupta et al., 2025), which leverages the LLM's in-context reasoning and copying abilities to perform retrieval and ranking from a pool of documents, with the docIDs provided as part of the prompt (Wang et al., 2024). For in-context ranking, one can interpret the longer prompt as an "augmented query"; our theory (Section 3) and loss function (Section 4), however, generalizes to parametric as well as in-context ranking.

3. The Capacity for Ranking

Let $\mathcal{Q}$ be a set of queries and $\mathcal{D}$ be a corpus of documents (or their docIDs). Towards establishing a theory of the ARR's model capacity for ranking, we start with a formal definition of the ranking task as follows.

Definition 3.1 (Ranking Task)**.** We consider a **ranking task** where for each query $q \in \mathcal{Q}$, the goal is to generate a list $\mathscr{L}(q) = [d_1(q), \ldots, d_k(q)]$ of relevant documents, $d_i(q) \in \mathcal{D}$. Ordering is significant: $\forall i < j$, $d_i(q)$ is considered more relevant to q than $d_j(q)$, denoted by $d_i(q) \succ d_j(q)$.

We refer to **Autoregressive Ranking (ARR)** as the use of a causal language model for ranking tasks. Such a model predicts the probability of a sequence of tokens conditioned on its preceding context. Given a query q, an ARR model defines a conditional probability distribution over document token sequences. The probability of any document $d \in \mathcal{D}$ conditioned on a query $q \in \mathcal{Q}$ is the product of d's token probabilities, calculated autoregressively.[1]

Both ARR and dual encoders (DEs) are examples of a **scoring-based ranking architecture**. Such architectures generate a score for each document given a query q, with the objective that the scores, when sorted in descending order, align with the ground-truth ordering of the relevant documents, and that scores for irrelevant documents are smaller than scores for relevant ones. We state this formally below.

Definition 3.2 (Scoring-based Ranking Architecture)**.** A (parametric) **scoring-based ranking architecture** is a function $f(q, d; \theta) : \mathcal{Q} \times \mathcal{D} \to \mathbb{R}$, which takes a pair q, d and produces a scalar score interpreted as the similarity. Here, θ denotes learnable model parameters.

In particular, we say that a scoring-based architecture $f(q, d; \theta)$ **can solve a ranking task** if there exists a θ such that the scores solve the ranking task for all queries in $\mathcal{Q}$.

To simplify our analysis, we consider an idealized case of the ranking task defined as follows.

Definition 3.3 (Complete Ranking Task)**.** Given $\mathcal{Q}$ and $\mathcal{D}$, we call a ranking task **complete** if:

- $\forall q \in \mathcal{Q}$, all documents are relevant, i.e., $\mathcal{D}(q) = \mathcal{D}$
- $\forall$ ordering $\pi(\mathcal{D})$ of the documents $\mathcal{D}$, $\exists q \in \mathcal{Q}$ s.t. its ranked list of relevant documents $\mathscr{L}(q) \equiv \pi(\mathcal{D})$.

We proceed to show that the architectural capacity of ARR for ranking is not limited in the same way as that of a DE.

3.1. Ranking via Dual Encoders

DEs are a popular choice for information retrieval owing to their computational efficiency. Unfortunately, DEs are usually insufficient due to their limited capacity compared to cross encoders (CEs), which are often critically employed in a subsequent reranking stage. Rigorous analysis of the capacity of DEs has been performed in the case of *retrieval*, where the ranking of the relevant documents is not considered. In particular, (Guo et al., 2019) shows that an embedding space dimension that grows linearly with the number of relevant documents is *sufficient* for a DE to correctly separate relevant documents from irrelevant ones. While this result hints at the potential difficulties of DEs in handling many relevant documents, it remains unclear whether such a dimension is also *necessary*.

In this section, we start with an analysis of a DE's capacity for the *ranking* task. Our result shows that an embedding dimension that grows linearly with the number of relevant documents is *necessary* for a DE to correctly rank them.

[1] Fortunately, the autoregressive nature of causal models offers a more efficient solution: Instead of scoring, we can directly generate the high-probability documents using greedy decoding or beam search. By employing beam search, we can generate the most likely documents conditioned on the query, directly yielding a ranked list without the need for exhaustive computation.

In DEs, the query and document are embedded into the same Euclidean space, with the negated Euclidean distance as the similarity score. Specifically, a dual encoder generates a similarity score for each pair of $q \in \mathcal{Q}$ and $d \in \mathcal{D}$ as

$$f_{\text{DE}}(q, d; \theta) = -\|E_Q(q; \theta_{\mathcal{Q}}) - E_D(d; \theta_{\mathcal{D}})\|_2 \tag{1}$$

where $E_Q : \mathcal{Q} \to \mathbb{R}^n$ and $E_D : \mathcal{D} \to \mathbb{R}^n$ are query and document encoders, respectively, and n is the dimension of the embedding space. Here, we assume that E_Q and E_D can have an arbitrary architecture. This means that they can be taken to be as expressive as needed when studying whether or not DEs are expressive enough to solve the ranking task. Our main result for DEs is the following.

Proposition 3.1 (DE Insufficiency for Complete Ranking)**.** *Let $f_{DE}(q, d; \theta)$ be a dual encoder architecture defined in Equation (1) with some embedding dimension n, and let $k := |\mathcal{D}|$. Then there does not exist a θ such that $f_{DE}(q, d; \theta)$ solves a complete ranking task when*

$$n < \frac{\ln k!}{2 \ln k}, \tag{2}$$

Proof. Let $N_{n,p}(k)$ denote the maximum number of distinct distance permutations generated by k sites in $\mathbb{R}^n$ with the L_p metric. From Skala (2009) Corollary 8, $N_{n,2}(k) \leq k^{2n}$. When Inequality (2) holds, we have $k^{2n} < k!$, and thus n-dimensional embeddings are unable to capture all $k!$ permutations required in a complete ranking task. □

Using Stirling's formula which states $\ln(k!) = k \ln k - k + O(\ln(k))$, the condition in Equation (2) becomes $n < \frac{k}{2} - \frac{k}{2 \ln k} + O(1)$. Since $k/\ln k$ grows sublinearly in k, **this result states that the embedding dimension n of the DE needs to grow linearly with the number of documents k.**

Note that the result holds regardless of the choice of the particular architecture for the encoders E_Q and E_D. This means that even if the encoder architectures are expressive enough to realize any function, a DE will not be able to solve a complete ranking task if n is not sufficiently large.

3.2. Ranking via (Pointwise) Autoregressive Rankers

We turn our attention to the ability of an ARR to rank documents using docIDs.[2] As is standard, we assume the ARR uses token embeddings $\mathrm{E} \in \mathbb{R}^{|\mathcal{V}| \times n}$ and computes next-token probabilities over the vocabulary $\mathcal{V}$ as $P(\cdot \mid c) = \sigma(\mathrm{E}\, \varphi(c))$ where $\varphi(c) \in \mathbb{R}^n$ is a hidden vector produced by the ARR given context c and σ is the softmax.

In the setting of document ranking, documents can be represented by unique identifiers, or docIDs, where each docID is a sequence of tokens from $\mathcal{V}_{\text{docIDs}} \subseteq \mathcal{V}$. To ensure the model only generates valid docID tokens we can make use of constrained decoding, which restricts next-token probabilities to those tokens in $\mathcal{V}_{\text{docIDs}}$ by computing

$$P(v \mid c) = \begin{cases} \sigma(\mathrm{E}_{\text{docIDs}}\, \varphi(c)) & \text{if} \quad v \in \mathcal{V}_{\text{docIDs}}, \\ 0 & \text{otherwise.} \end{cases}$$

where $\mathrm{E}_{\text{docIDs}} \in \mathbb{R}^{|\mathcal{V}_{\text{docIDs}}| \times n}$ represents a matrix whose rows are the token embeddings associated with $\mathcal{V}_{\text{docIDs}}$. To study the rank requirements of $\mathrm{E}_{\text{docIDs}}$, we introduce for convenience the matrix $\mathrm{E}' := [\mathrm{E}_{\text{docIDs}}\, \mathbf{1}_{|\mathcal{V}_{\text{docIDs}}|}] \in \mathbb{R}^{|\mathcal{V}_{\text{docIDs}}| \times (n+1)}$, i.e., $\mathrm{E}_{\text{docIDs}}$ augmented with a column of ones.

We consider an idealized "infinite-capacity" setting where the hidden vector $\varphi(c)$ can be any vector in $\mathbb{R}^n$.

Proposition 3.2 (Arbitrary Distributions over Tokens)**.** *An infinite-capacity ARR can produce any (strictly positive) probability distribution over tokens in $\mathcal{V}_{docIDs}$ via constrained decoding iff* $\mathrm{rank}(\mathrm{E}') = |\mathcal{V}_{docIDs}|$.

The contrapositive of the forward implication is commonly known as the "softmax bottleneck" (Yang et al., 2017).

We note that many existing language models come with "special tokens" which can be used as $\mathcal{V}_{\text{docIDs}}$, and for any existing language model which does not come with pre-allocated "special tokens" we can expand the vocabulary with n new tokens and extend its embedding matrix with the $I_{|\mathcal{V}_{\text{docIDs}}|}$ identity matrix, which would satisfy the requirements. An implicit limitation of this is that $|\mathcal{V}_{\text{docIDs}}| \leq n$, however the chain-rule of probability implies the following.

[2] Proofs for this section can be found in Appendix A.

Corollary 3.1 (Arbitrary Distributions over Sequences)**.** *An infinite-capacity ARR can produce any (strictly positive) probability distribution over sequences of tokens $\mathcal{V}_{docIDs}$ via constrained decoding iff* $\mathrm{rank}(\mathrm{E}') = |\mathcal{V}_{docIDs}|$.

Producing arbitrary probability distributions over sequences clearly entails the ability to produce arbitrary *rankings* over sequences. One may wonder, however, whether the requirement of Proposition 3.2 / Corollary 3.1 is also *necessary* for ranking. Surprisingly, at a single token position the requirement for ranking is the same as for distributions:

Proposition 3.3 (Arbitrary Permutations over Tokens)**.** *An infinite-capacity ARR can produce any permutation over tokens in $\mathcal{V}_{docIDs}$ via constrained decoding iff* $\mathrm{rank}(\mathrm{E}') = |\mathcal{V}_{docIDs}|$.

We observe that the capacity to produce arbitrary permutations over *sequences* depends on the capacity to express all $|\mathcal{V}_{\text{docIDs}}|!$ rankings at *a single token position*. Since the requirement for this is the same as the requirement to represent arbitrary distributions at both token- and sequence-level, we conclude that the rank requirement expressed in Proposition 3.2, Corollary 3.1, and Proposition 3.3 is not only sufficient but necessary for the ranking task.

Remark: *Is the theoretical requirement on E' satisfied by LLMs used in practice for reranking, and why are those LLMs successful even when this requirement clearly can not be satisfied? Our experiments on WordNet and ESCI (Section 5.1) reflect two common scenarios. In WordNet, where docIDs comprise English nouns, the docID tokens span almost all of $|\mathcal{V}|$. Since the model dimension is much smaller, E' can not achieve a rank even close to $|\mathcal{V}_{docIDs}|$.*[3] *However, most of the possible sequences are simply never encountered, and the model is tasked with learning a very small subset of all possible permutations. In ESCI, by contrast, we choose numerical docIDs, and the resulting $|\mathcal{V}_{docIDs}|$ is therefore closer to the model dimension, allowing the rank requirement on E' to be matched more accurately.*

At either the single token position or the sequence level, we see that the same requirement on the docID token embeddings (namely, that $\mathrm{rank}(\mathrm{E}') = |\mathcal{V}_{\text{docIDs}}|$, i.e., slightly weaker than $\mathrm{E}_{\text{docIDs}}$ being full-rank) **is both necessary and sufficient to enable a fixed-dimension ARR to generate arbitrary probability distributions or arbitrary permutations for an unrestricted number of documents.** Under the infinite-capacity assumption, this ability is strictly superior to that of DEs, whose embedding dimension must grow linearly with the number of documents to achieve arbitrary rankings.

4. Generalized Rank-Aware Training Loss

Given an example "point" of the form (q, d_r, r) with the query, a docID, and the true rank, we propose a generalized loss function for autoregressive ranking which can be viewed as a weighted item-level loss, with the weight coefficient $\lambda(r)$ for a given docID being a function of the document's rank r. The item-level loss decomposes as a sum of cross entropy losses between the model predictions and target distributions $\mathbf{y}$ at each timestep (token position) of the docID under consideration. Note that the target distributions are a function of the timestep t and potentially also of the current docID's rank r, hence $\mathbf{y}(r, t)$. Our generalized next-token prediciton loss $\mathcal{L}(q, d_r, r; \theta)$ for ranking is given below. θ represents model parameters, $\mathcal{T}$ the tokenizer, and $\bar{q}$ the "augmented query" (cf. Section 5.1):

$$\lambda(r) \sum_{t=1}^{|\mathcal{T}(d_r)|} \Big(\mathrm{CE}\Big(\mathbf{y}(r,t), p_\theta(\mathcal{T}(d_r)[t] \mid \mathcal{T}(\bar{q}), \mathcal{T}(d_r)_{<t}) \Big) \Big) \tag{3}$$

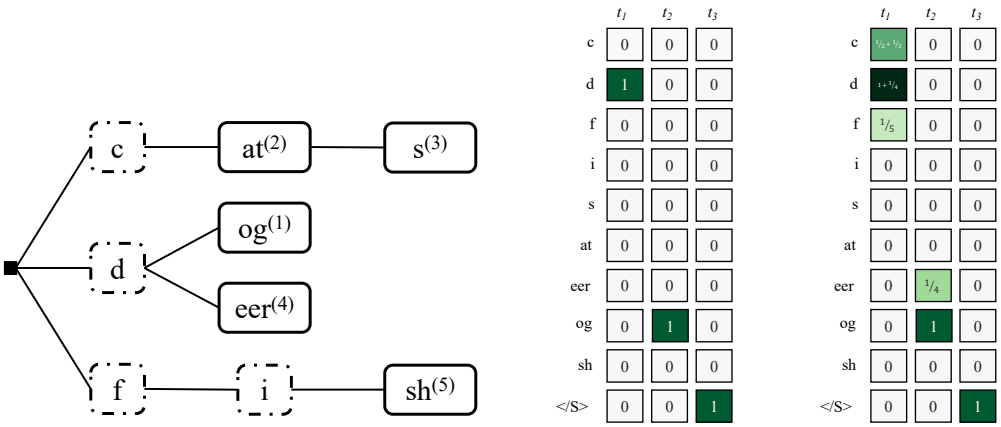
Observe that if $\mathbf{y}(r, t)$ is set to a one-hot vector, the summation across a docID's timesteps simplifies to the negative log-likelihood of that docID. More generally, we can view our loss as a weighted sum of item-level log-likelihoods, for some reweighting function $\lambda(r)$, and for some customizable construction $\mathbf{y}(r, t)$ of target distributions per timestep.

The most common target distribution $\mathbf{y}(r, t)$ is one-hot, i.e., $\mathbf{y}(t)_i \coloneqq \mathbb{I}_{[v_t = i]}$. Section 4.2 presents a rank-aware improvement to $\mathbf{y}(r, t)$.

4.1. Rank-Aware Item-Level Reweighting

A simple improvement to next-token prediction is to reweight the loss contribution of $d_r, r \in [1, n_q]$ by setting $\lambda(r)$ to a function that decreases as rank r increases (rank 1 represents the most relevant document). We consider two simple variants: a *fractional* reweighting function $\lambda(r) = \frac{1}{r^\alpha}$ where α is a temperature hyperparameter, or a *stepwise* reweighting function $\lambda(r) = \frac{n_q - r + 1}{n_q}$ where n_q is the total number of documents for the query. If we decide to omit

[3] Mistral-7B-v0.3-it has $|\mathcal{V}| = 32,768$ and $d = 4,096$.



(a) Trie over ranked tokenized docIDs *(b)* One-hot *(c)* Marginalization

Figure 1. Illustration of (a) a prefix-tree; target distributions derived from it via (b) one-hot supervision versus (c) marginalization.

some documents from the training (for example, train a top-1 retrieval model only) we can still express this by setting our reweighting function to the *indicator* function ($\lambda(r) = \mathbb{I}_{r=1}$ in the top-1 case).

4.2. Prefix Tree for Rank-Aware Token-Level $\mathbf{y}(r, t)$

A prefix tree (aka trie) groups strings according to their shared prefixes for quickly determining (by traversing left-to-right, i.e., from the root down) whether a given sequence belongs to the valid set of sequences or not. Building a prefix tree over the list of tokenized docIDs allows us to examine, at each timestep, which continuation docIDs are permissible given the partially-generated docID, and to distribute mass accordingly among valid tokens in the target distribution.

Given tokenized target docIDs in the following toy example — d_1: "`dog`" $\Rightarrow$ ([`d`, `og`]); d_2: "`cat`" $\Rightarrow$ ([`c`, `at`]); d_3: "`cats`" $\Rightarrow$ ([`c`, `at`, `s`]); d_4: "`deer`" $\Rightarrow$ ([`d`, `eer`]); d_5: "`fish`" $\Rightarrow$ ([`f`, `ish`]) — we can construct a prefix tree over their tokens, as shown in Figure 1a. The leaf nodes and potentially some nonterminal nodes of the trie, which we term *docID nodes* and represent with solid borders in Figure 1a, will each represent the tokenization of a docID d_r, and to such nodes we assign the score $\mu(r) = \frac{1}{r^\beta}$ for temperature hyperparameter β. To score any given node in the trie (e.g., for a partially-generated docID), we can marginalize over the scores of all docIDs which that node governs.[4]

Given an empty string (ε) at timestep t_1, the trie tells us that permissible continuation tokens for [ε] are {`c, d, f`}. At timestep t_2, if our highest-ranked document is $d_1 =$ ``dog'', we will have teacher-forced token `d` as the t_2 input to the model, and the prefix tree will consequently yield tokens {`og, eer`} as valid continuations for [ε, `d`]. We illustrate the (pre-normalized) target distributions produced by this trie-based marginalization process in Figure 1c.[5]

5. Experiments

5.1. Datasets, Experimental Settings, and Metrics

WordNet WordNet (Fellbaum, 1998) includes a hierarchy over noun concepts, aka "synsets", where the top-level node is `entity.n.01` (the "`n`" suffix means *noun*, and "`01`" that this is the first of multiple possible word senses). To obtain a query, we randomly sample a noun synset at depth one or greater (i.e., excluding the root `entity.n.01` itself). The relevant documents for the sampled query synset are its hypernym synsets, meaning all the more abstract synsets on the path from the query up to the root. For each query, we additionally sample a negative document not

[4]At marginalization, the fractional $\mu(r)$ prevents a continuation which leads to the top-r docID from being outweighed by a continuation leading to multiple lower-ranked docIDs. As $\beta \to \infty$, the resulting distributions converge to one-hots.

[5]Technically, for timesteps $t > 1$, teacher forcing causes a mismatch between the rank-aware training described here and the model's actual decision-making at inference time. For example, if the model chooses ``c'' at timestep $t$=1 during inference, we will not have smoothed over ``at'' at $t = 2$ during training to encourage the second-best ``cat'' to be generated (without constrained decoding, we might well generate ``c'' + ``og'' $\to$ ``cog'' in such a scenario). Nevertheless, akin to Reddy et al. (2024), our method does encourage full rank-awareness at $t = 1$. We leave the question of mitigating this rank-aware training-inference mismatch for $t > 1$ to future work.

Table 1. WordNet results (evaluated over 5000 (query, targets)-examples)

		CVR (↓)	nDCG (↑)	R@1 (↑)	R@2 (↑)	R@3 (↑)	R@4 (↑)	R@5 (↑)
NTP: $\lambda(r) = \mathbb{I}_{r=1}$, $\mathbf{y}$ one-hot		27.66%	94.89	**99.96**	62.43	55.3	55.16	63.42
$\lambda(r) = \frac{1}{r^\alpha}$ $\mathbf{y}$ one-hot	$\alpha = 1$	0.0%	99.6	91.1	87.68	88.81	92.13	93.94
	$\alpha = 2$	0.0%	**99.83**	97.74	95.69	95.48	**96.89**	**96.58**
	$\alpha = 3$	0.0%	99.81	99.22	97.35	**96.16**	96.07	95.53
	$\alpha = 4$	0.02%	99.78	99.36	**97.46**	95.63	96.03	95.31
	$\alpha = 5$	0.04%	99.67	99.6	97.14	94.55	95.71	93.52
$\lambda(r) = \frac{n_q - r + 1}{n_q}$, $\mathbf{y}$ one-hot		0.0%	99.6	51.62	69.88	82.09	89.6	93.34
$\lambda(r) = \mathbb{I}_{r=1}$ $\mathbf{y}$ marg. over trie w/ leaf scores $\frac{1}{r^\beta}$	$\beta = 1$	1.48%	96.54	98.1	72.93	63.06	59.64	64.25
	$\beta = 2$	1.38%	96.54	99.3	67.31	57.51	56.98	62.45
	$\beta = 3$	2.04%	96.34	99.44	67.58	59.95	58.58	63.17
	$\beta = 4$	4.54%	96.36	99.58	68.53	62.25	62.12	67.41
	$\beta = 5$	13.04%	96.47	99.78	70.84	65.12	64.81	70.43

on the hypernym path to be scored alongside the relevant documents. We provide a zero-shot prompt for WordNet in Appendix D.1. For example (we omit the suffixes for brevity):

- **Query:** deer
- **Ranked docIDs:** ruminant $\succ$ even-toed ungulate $\succ$ ungulate $\succ$ placental $\succ$ mammal $\succ$ vertebrate $\succ$ chordate $\succ$ animal $\succ$ organism $\succ$ living thing $\succ$ whole $\succ$ object $\succ$ physical entity [$\succ$ entity]
- **Negative docID:** solarization

ESCI Shopping Queries We process the *Shopping Queries* dataset of (Reddy et al., 2022) as follows to obtain a reranking dataset. We use the Gecko (Lee et al., 2024b) embeddings (specifically from the `gecko-1b-en` model) of the queries and product titles to obtain a matrix of inner products between all queries and all product titles. To aggregate the relevant product titles for a query, we take the $1^{\text{st}}, 100^{\text{th}}, 200^{\text{th}}, \ldots, 10000^{\text{th}}$ product titles that have largest dot product with the query, in descending order. We choose step size 100 to ensure that the difference in relevance between the nth and the $(n+1)^{\text{st}}$ product title is meaningful. Finally, we filter out all non-English queries and product titles, and preserve only those examples that have a non-empty set of relevant English product titles.

Unlike the WordNet dataset (Section 5.1), the documents of ESCI are (*a*) substantially longer, i.e., on the order of dozens of tokens instead of ten, and (*b*) and have not been sampled from a taxonomy. Issue (*a*) motivates the creation of docIDs that are both short enough to generate as pointwise targets, and structured enough so that a prefix tree can be built over them for target training distributions as discussed in Section 4.2. We create such docIDs via sparse dictionary learning over the Gecko product title embeddings to obtain sparse vectors, each of dimension 100 with 3 nonzero values.[6] The docID for a given product title is then taken to be the concatenation of the indices of the nonzero entries in its corresponding sparse vector, sorted in descending order by the absolute values of the entries (e.g., "`25,36,39`"). A zero-shot prompt for ESCI is provided in Appendix D.2.

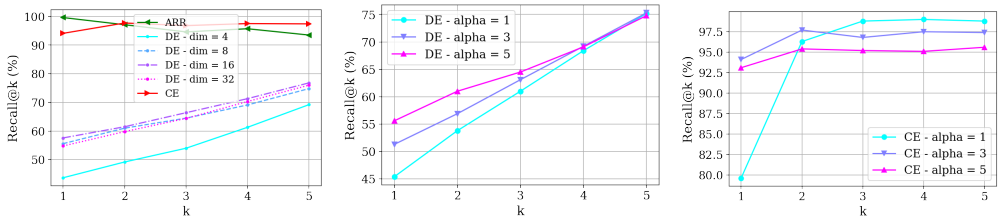
Task Setup for Autoregressive Ranking The in-context input to our Causal Transformer is a query q and an unordered docID set $\text{shuffle}(\{d_1, \ldots, d_{n_q}\})$; assume the ground truth ranking is $d_1 \succ \ldots \succ d_{n_q}$. We construct the prompt, denoted by $\bar{q}$ for the query augmented with the corpus and system prompt, as $\bar{q} := \text{prompt}(q, \text{shuffle}(\{d_1, \ldots, d_{n_q}\}))$, cf. Appendix D. We emphasize that this augmented prompt construction reflects an *in-context* approach to ranking (Gupta et al., 2025) in which the full corpus is provided as context — while this contrasts with DEs' and CEs' pairwise scoring approach, it is natural to ARR, which aims to exploit the in-context copying and reasoning abilities of decoder LLMs. For our evaluation, as a simple proxy to beam search, we perform greedy decoding for $\max_{i \in [1, n_q]} |\mathcal{T}(d_i)|$ steps, where $\mathcal{T}$ is the tokenizer, and score any document by averaging the log-probabilities of its constituent tokens, i.e., $\text{score}(d_j) = \frac{1}{|\mathcal{T}(d_j)|} \sum_{k=1}^{|\mathcal{T}(d_j)|} \log p_\theta(\mathcal{T}(d_j)[k] \mid \mathcal{T}(\bar{q}), \mathcal{T}(d_j)_{<k})$.[7] We use Mistral-7B-v0.3-it (Jiang, 2024) for all fine-tuning experiments in this paper, with settings for the generative approach in Appendix B.1.

[6] We use Orthogonal Matching Pursuit from `scikit-learn` (Pedregosa et al., 2011), `n_components=100` and `transform_n_nonzero_coefs=3`.

[7] We emphasize that in practice, a top $-k$ ranking can still be directly generated via beam search using beam size k.

Table 2. ESCI results (evaluated over 310 (query, targets)-examples)

		nDCG (↑)	R@1 (↑)	R@2 (↑)	R@5 (↑)	R@10 (↑)	R@25 (↑)	R@50 (↑)
NTP: $\lambda(r) = \mathbb{I}_{r=1}$, y one-hot		95.23	**95.16**	52.58	27.64	23.51	37.98	62.99
$\lambda(r) = \mathbb{I}_{r=1}$ y marg. over trie w/ leaf scores $\frac{1}{r^\beta}$	$\beta = 1$	**97.21**	70.0	56.61	45.57	46.27	54.78	**69.58**
	$\beta = 2$	**97.21**	70.32	**58.06**	**51.07**	**48.08**	**54.96**	69.03
	$\beta = 3$	97.11	68.71	56.13	48.03	45.78	52.31	67.63
	$\beta = 4$	96.96	69.03	54.84	44.6	43.9	50.29	67.7
	$\beta = 5$	96.89	66.77	56.13	44.21	42.4	50.72	67.61



(a) Comparison with DE and CE *(b)* Effect of α for DE *(c)* Effect of α for CE

Figure 2. Results on WordNet with Dual Encoders (DEs) and Cross Encoders (CEs). (a) Comparison of (pointwise) autoregressive ranking with DEs of varying embedding dimension $n \in \{4, 8, 16, 32\}$ and a 4-layer CE. (b, c) Effect of α in the reweighting function $\lambda(r) = \frac{1}{r^\alpha}$ of the training loss for DEs and CEs, respectively.

Metrics We adopt the following metrics – the first one a heuristic to assess the constraint satisfaction of our approach, the other two commonly-used metrics for ranking:

- For each example of the WordNet dataset, we use the sampled negative document to assess the **constraint violation rate (CVR)**, a heuristic measurement we define as follows. Given n positive documents $d_1^+, \ldots, d_n^+$ and 1 negative (irrelevant) document d^-, we say that the model has incurred a violation if the log-probability the model assigns to any of the d_i^+ is lower than the log-probability it assigns to d^-.
- We measure the **normalized discounted cumulative gain (nDCG)** between "gold" scores and predicted log-probabilities (including for the sampled negative document). The "gold" are in log-scale in decreasing order of relevance: $\text{score}(d_1^+) = \log(n_q + 1), \text{score}(d_2^+) = \log(n_q) \ldots, \text{score}(d_{n_q-1}^+) = \log(3), \text{score}(d_{n_q}^+) = \log(2), \text{score}(d_q^-) = \log(1) = 0$.
- We use the log-probabilities to argsort over the docIDs, and use the gold ranking π and predicted ranking $\hat{\pi}$ to measure **recall-at-k (R@k)** as $\frac{|\pi[:k] \cap \hat{\pi}[:k]|}{k}$.

5.2. Main Results

In Tables 1 and 2 we present our experimental results on the WordNet and ESCI datasets, respectively. Each of our **ranking-aware loss functions can be seen to drastically reduce the constraint violation rate** (cf. Table 1) — i.e., the model learns to score irrelevant docIDs lower than relevant docIDs for a query. Such a shift in probability mass from invalid to valid docIDs makes the resulting LLM more aligned, as it were "by construction", with the constraints of generating valid docIDs. Such realignment would naturally ease the computational burden of a debiasing algorithm for constrained decoding, e.g., as described in Ye et al. (2025).

Fixing our **y** to one-hots, we observe for WordNet in Table 1 that *fractional* ($\lambda(r) = \frac{1}{r^\alpha}$) item-level reweighting is more effective than *stepwise* ($\lambda(r) = \frac{n_q - r + 1}{n_q}$), while in the limit $\alpha \to \infty$ for fractional reweighting, the target distributions look more and more like a top-1 training curriculum $\frac{1}{r^\alpha} \to \mathbb{I}_{r=1}$ (cf. Section 4.1). We separately explore the effect of rank-aware target distributions $\mathbf{y}(r, t)$ (cf. Section 4.2), with a single rank-aware target per query (rather than per document d_r). We similarly observe decreased constraint violation rate, improved nDCG and R@K for $K > 2$, but not as dramatic as for the item-level reweighting loss.

The ESCI dataset contains a much higher number of product titles to be reranked per query on average than the WordNet dataset (cf. Section C). Training with item-level losses until convergence would treat each docID d_r for $r \in [1, n_q]$ as a separate target, and would thus require a proportionally larger number of training steps to iterate over the entire dataset. Hence, we run our ESCI experiments under the same budget constraint as next-token prediction, providing a single

target per every query by setting $\lambda(r) = \mathbb{I}_{r=1}$, and inject ranking information via trie-based marginalization. While hurting R@1 performance, Table 2 improves R@K for all $K > 1$ as well as the aggregate nDCG score. **Trie-based marginalization can thus be viewed as an economical alternative to the item-level reweighting**, even if it performs second to item-level reweighting under the assumption of unlimited training examples. Additionally, the improved ESCI ranking scores highlight the **synergy between this token-level loss and the ESCI docIDs, which we specifically learned to be trie-friendly**. We hope this result stimulates future research into optimal combinations of docID learning algorithms with token-based ranking losses.

5.3. Comparison with DEs and CEs

We compare (pointwise) autoregressive ranking with DEs and CEs. We conduct our experiment on WordNet, and report the results in Figure 2a. It can be seen that autoregressive ranking is on par with CEs while being significantly more powerful than DEs; comparisons against industry-scale retrieval and reranking systems on larger benchmarks are warranted as future work. Below we detail our choice of DEs and CEs, with hyperparameters in Sections B.2 and B.3, respectively.

Since the number of noun synsets in WordNet is relatively small (i.e., 82,155), we simply use an embedding table of size 82,155$\times n$ to map each synset to an n-dimensional embedding (as opposed to using a tokenizer). Denote this embedding lookup function as $\mathrm{E}_\phi(\cdot)$, where ϕ is the trainable embedding table. Given a pair (q, d), we use DEs with the inner product $\langle \mathrm{E}_\phi(q), \mathrm{E}_\phi(d)\rangle$ as their similarity score. In other words, we use lookup table DEs which is a common choice on this dataset (Dhingra et al., 2018; Parmar & Narayan, 2022; An et al., 2023). Given a batch of positive data $\{(q_i, d_i, r_i)\}_{i=1}^B$, we train the DE by minimizing the following **weighted** batch softmax loss $\mathcal{L}_{\text{DE}}(\{q_i, d_i, r_i\}_{i=1}^B; \phi)$:

$$-\frac{1}{B}\sum_{i=1}^{B}\left(\lambda(r_i)\cdot\log\frac{\exp\Big(\langle \mathrm{E}_\phi(q_i), \mathrm{E}_\phi(d_i)\rangle/\tau\Big)}{\sum_{j=1}^{B}\exp\Big(\langle \mathrm{E}_\phi(q_i), \mathrm{E}_\phi(d_j)\rangle/\tau\Big)}\right). \tag{4}$$

where τ is a scalar that we fix to be 0.05. Note that this formulation incorporates the reweighting factor $\lambda(r)$. If $\lambda(r) = 1$ then the loss reduces to the standard batch softmax loss commonly used in information retrieval (Wu et al., 2024). Here, we follow the same idea as in (3) and use $\lambda(r) = \frac{1}{r^\alpha}$. We report the results for varying α in Figure 2b, which shows that an increasing α improves Recall@k, particularly for small k. This result shows that our reweighting factor can be useful broadly for ranking problems, beyond the particular autoregressive ranking approach.

For CE, we concatenate the embeddings into a $2n$-dimensional vector which is passed to an MLP $f_{\text{MLP}}(\text{concat}(\mathrm{E}_\phi(q), \mathrm{E}_\phi(d)); \psi) : \mathbb{R}^{2n} \mapsto \mathbb{R}$, where ψ is trainable parameters. We use ReLU activation function, three hidden activations of size $2n$, and a final output of dimension 1 as the similarity. To train a CE, one cannot use the same batch softmax loss as for the DEs since scoring each query against all in-batch negatives is costly. Here, for a batch of positive data $\mathcal{P} = \{(q_i, d_i, r_i)\}_{i=1}^B$ we create a negative set of the same size given by $\mathcal{N} = \{(q_i, d_{\rho(i)})\}_{i=1}^B$, where $\rho : \{1, \ldots, B\} \mapsto \{1, \ldots, B\}$ is a randomly-chosen permutation. Then we minimize $\mathcal{L}_{\text{CE}}(\mathcal{P}, \mathcal{N}; \phi, \psi)$:

$$-\sum_{i=1}^{B}\left(\frac{\lambda(r_i)}{\sum_{j=1}^{B}\lambda(r_j)}\log\sigma\Big(f_{\text{MLP}}(\text{concat}(\mathrm{E}_\phi(q_i), \mathrm{E}_\phi(d_i)); \psi)\Big) + \frac{1}{B}\log\sigma\Big(-f_{\text{MLP}}(\text{concat}(\mathrm{E}_\phi(q_i), \mathrm{E}_\phi(d_{\rho(i)})); \psi)\Big)\right), \tag{5}$$

with $\sigma(\cdot)$ the sigmoid function. A reweighting term $\lambda(r_i)$ is again added to positive pairs (effect illustrated in Figure 2c).

6. Conclusion

In this paper, we established a theoretical foundation for Autoregressive Ranking, proving that — under an infinite-capacity assumption — while DEs require embedding dimensions to grow with corpus size, ARR models generating multi-token docIDs can solve complete ranking tasks with a constant hidden dimension, given a mild condition on the rank of the embedding matrix for the docID tokens. We proposed a generalized rank-aware training loss for (pointwise) autoregressive ranking that relies on item-level reweighting and prefix-tree marginalization to distribute probability mass over valid docID tokens based on their ground-truth relevance. Experiments on WordNet and ESCI show this approach successfully suppresses invalid docID generations and improves on key ranking metrics; extending this empirical validation to larger benchmarks and industry-scale systems is a natural next step.

Acknowledgements. The authors would like to thank Aditya K Menon for valuable discussions about this paper and generative information retrieval more broadly.

Impact Statement. This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

An, K., Zheng, C., Gao, B., Zhao, H., and Chang, B. Coarse-to-fine dual encoders are better frame identification learners. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 13455–13466, 2023.

Cao, Z., Qin, T., Liu, T.-Y., Tsai, M.-F., and Li, H. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pp. 129–136, 2007.

Chen, W., Liu, T.-Y., Lan, Y., Ma, Z.-M., and Li, H. Ranking measures and loss functions in learning to rank. *Advances in Neural Information Processing Systems*, 22, 2009.

Dhingra, B., Shallue, C., Norouzi, M., Dai, A., and Dahl, G. Embedding text in hyperbolic spaces. In *Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12)*, pp. 59–69, 2018.

Drozdov, A., Zhuang, H., Dai, Z., Qin, Z., Rahimi, R., Wang, X., Alon, D., Iyyer, M., McCallum, A., Metzler, D., et al. Parade: Passage ranking using demonstrations with large language models. *arXiv preprint arXiv:2310.14408*, 2023.

Fellbaum, C. *WordNet: An electronic lexical database*. MIT press, 1998.

Guo, C., Mousavi, A., Wu, X., Holtmann-Rice, D. N., Kale, S., Reddi, S., and Kumar, S. Breaking the glass ceiling for embedding-based classifiers for large output spaces. *Advances in Neural Information Processing Systems*, 32, 2019.

Guo, R., Sun, P., Lindgren, E., Geng, Q., Simcha, D., Chern, F., and Kumar, S. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pp. 3887–3896. PMLR, 2020.

Gupta, N., You, C., Bhojanapalli, S., Kumar, S., Dhillon, I., and Yu, F. Scalable in-context ranking with generative models. *arXiv preprint arXiv:2510.05396*, 2025.

Huang, Y., Chen, Y., Cao, X., Yang, R., Qi, M., Zhu, Y., Han, Q., Liu, Y., Liu, Z., Yao, X., et al. Towards large-scale generative ranking. *arXiv preprint arXiv:2505.04180*, 2025.

Jiang, F. Identifying and mitigating vulnerabilities in llm-integrated applications. Master's thesis, University of Washington, 2024.

Karpukhin, V., Oguz, B., Min, S., Lewis, P. S., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pp. 6769–6781, 2020.

Lee, J., Chen, A., Dai, Z., Dua, D., Sachan, D. S., Boratko, M., Luan, Y., Arnold, S. M., Perot, V., Dalmia, S., et al. Can long-context language models subsume retrieval, rag, sql, and more? *arXiv preprint arXiv:2406.13121*, 2024a.

Lee, J., Dai, Z., Ren, X., Chen, B., Cer, D., Cole, J. R., Hui, K., Boratko, M., Kapadia, R., Ding, W., et al. Gecko: Versatile text embeddings distilled from large language models. *arXiv preprint arXiv:2403.20327*, 2024b.

Li, L., Zhang, Y., Liu, D., and Chen, L. Large language models for generative recommendation: A survey and visionary discussions. *arXiv preprint arXiv:2309.01157*, 2023a.

Li, Y., Yang, N., Wang, L., Wei, F., and Li, W. Multiview identifiers enhanced generative retrieval. *arXiv preprint arXiv:2305.16675*, 2023b.

Li, Y., Yang, N., Wang, L., Wei, F., and Li, W. Learning to rank in generative retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 8716–8723, 2024.

Liu, T.-Y. et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3): 225–331, 2009.

Luce, R. D. et al. *Individual choice behavior*, volume 4. Wiley New York, 1959.

Luo, S., He, B., Zhao, H., Shao, W., Qi, Y., Huang, Y., Zhou, A., Yao, Y., Li, Z., Xiao, Y., et al. Recranker: Instruction tuning large language model as ranker for top-k recommendation. *ACM Transactions on Information Systems*, 2024.

Metzler, D., Tay, Y., Bahri, D., and Najork, M. Rethinking search: making domain experts out of dilettantes. In *Acm sigir forum*, volume 55, pp. 1–27. ACM New York, NY, USA, 2021.

Nogueira, R. and Cho, K. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

Parmar, M. and Narayan, A. Hyperbox: A supervised approach for hypernym discovery using box embeddings. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pp. 6069–6076, 2022.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12: 2825–2830, 2011.

Plackett, R. L. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2): 193–202, 1975.

Pradeep, R., Sharifymoghaddam, S., and Lin, J. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *arXiv preprint arXiv:2309.15088*, 2023a.

Pradeep, R., Sharifymoghaddam, S., and Lin, J. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*, 2023b.

Reddy, C. K., Màrquez, L., Valero, F., Rao, N., Zaragoza, H., Bandyopadhyay, S., Biswas, A., Xing, A., and Subbian, K. Shopping queries dataset: A large-scale esci benchmark for improving product search. *arXiv preprint arXiv:2206.06588*, 2022.

Reddy, R. G., Doo, J., Xu, Y., Sultan, M. A., Swain, D., Sil, A., and Ji, H. First: Faster improved listwise reranking with single token decoding. *arXiv preprint arXiv:2406.15657*, 2024.

Sachan, D. S., Lewis, M., Joshi, M., Aghajanyan, A., Yih, W.-t., Pineau, J., and Zettlemoyer, L. Improving passage retrieval with zero-shot question generation. *arXiv preprint arXiv:2204.07496*, 2022.

Shazeer, N. and Stern, M. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Skala, M. Counting distance permutations. *Journal of Discrete Algorithms*, 7(1):49–61, 2009.

Sun, W., Yan, L., Ma, X., Wang, S., Ren, P., Chen, Z., Yin, D., and Ren, Z. Is chatgpt good at search? investigating large language models as re-ranking agents. *arXiv preprint arXiv:2304.09542*, 2023.

Tay, Y., Tran, V., Dehghani, M., Ni, J., Bahri, D., Mehta, H., Qin, Z., Hui, K., Zhao, Z., Gupta, J., et al. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35:21831–21843, 2022.

Wang, L., Yang, N., Huang, X., Yang, L., Majumder, R., and Wei, F. Large search model: Redefining search stack in the era of llms. In *ACM SIGIR Forum*, volume 57, pp. 1–16. ACM New York, NY, USA, 2024.

Wu, J., Wang, X., Gao, X., Chen, J., Fu, H., and Qiu, T. On the effectiveness of sampled softmax loss for item recommendation. *ACM Transactions on Information Systems*, 42(4):1–26, 2024.

Xia, F., Liu, T.-Y., Wang, J., Zhang, W., and Li, H. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pp. 1192–1199, 2008.

Yang, Z., Dai, Z., Salakhutdinov, R., and Cohen, W. W. Breaking the softmax bottleneck: A high-rank rnn language model. *arXiv preprint arXiv:1711.03953*, 2017.

Ye, H., Jain, H., You, C., Suresh, A. T., Lin, H., Zou, J., and Yu, F. Efficient and asymptotically unbiased constrained decoding for large language models. *arXiv preprint arXiv:2504.09135*, 2025.

Yuan, P., Wang, X., Feng, S., Pan, B., Li, Y., Wang, H., Miao, X., and Li, K. Generative dense retrieval: Memory can be a burden. In *18th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2024*, pp. 2835–2845. Association for Computational Linguistics (ACL), 2024.

A. Proofs

A.1. Proof of Proposition 3.2

Proposition 3.2 (Arbitrary Distributions over Tokens)**.** *An infinite-capacity ARR can produce any (strictly positive) probability distribution over tokens in $\mathcal{V}_{docIDs}$ via constrained decoding iff* $\mathrm{rank}(\mathrm{E}') = |\mathcal{V}_{docIDs}|$.

Proof. ($\Rightarrow$) Assume, *reductio ad absurdum*, that $\mathrm{rank}(\mathrm{E}') < |\mathcal{V}_{\text{docIDs}}|$. This implies that there exists a vector $h \in \mathbb{R}^{|\mathcal{V}_{\text{docIDs}}|}$, $h \neq \mathbf{0}_{|\mathcal{V}_{\text{docIDs}}|}$, that is orthogonal to the span of E', i.e., for any $\varphi' \in \mathbb{R}^{n+1}$:

$$h^T(\mathrm{E}'\,\varphi') = \sum_{i=1}^{|\mathcal{V}_{\text{docIDs}}|} h_i(\mathrm{E}'\,\varphi')_i = 0. \tag{6}$$

Let $\mathcal{P} \coloneqq \{i \mid h_i > 0\}$ and $\mathcal{N} \coloneqq \{i \mid h_i < 0\}$. Then we can write

$$\sum_{i\in\mathcal{P}} h_i(\mathrm{E}'\,\varphi')_i + \sum_{i\in\mathcal{N}} h_i(\mathrm{E}'\,\varphi')_i = 0 \tag{7}$$

from which it follows that

$$\sum_{i\in\mathcal{P}} h_i(\mathrm{E}'\,\varphi')_i = \sum_{i\in\mathcal{N}} |h_i|(\mathrm{E}'\,\varphi')_i, \tag{8}$$

and since $\mathbf{1}_{|\mathcal{V}_{\text{docIDs}}|}$ is clearly in the span of E', we also have

$$S \coloneqq \sum_{i\in\mathcal{P}} h_i = \sum_{i\in\mathcal{N}} |h_i|. \tag{9}$$

Since our ARR can produce any strictly positive probability distribution over $\mathcal{V}_{\text{docIDs}}$, it must be able to produce hidden vector $\varphi(c)$ corresponding to logits $z = \mathrm{E}_{\text{docIDs}}\,\varphi(c)$ with $z_i > z_j$ for all $i \in \mathcal{N}, j \in \mathcal{P}$ — such logits correspond to a probability distribution in which all token indices for which h takes negative values are assigned greater probability mass than any of the token indices for which h takes positive values.

We can choose $\varphi' \in \mathbb{R}^{n+1}$ such that $\varphi'_{1:n} = \varphi(c)$, and $z' = \mathrm{E}'\,\varphi' = \mathrm{E}_{\text{docIDs}}\,\varphi(c) + \varphi'_{n+1}\mathbf{1}$ will have the same relation between values indexed by $\mathcal{N}$ and by $\mathcal{P}$ as does z, i.e.,

$$a \coloneqq \max\{z'_i \mid i \in \mathcal{P}\} < \min\{z'_i \mid i \in \mathcal{N}\} =: b. \tag{10}$$

Recall from Equation (8) that $\sum_{i\in\mathcal{P}} h_i z'_i = \sum_{i\in\mathcal{N}} |h_i| z'_i$. Now,

$$\sum_{i\in\mathcal{P}} h_i z'_i \overset{(10)}{\leq} \sum_{i\in\mathcal{P}} h_i a = a \sum_{i\in\mathcal{P}} h_i \overset{(9)}{=} aS, \tag{11}$$

and

$$\sum_{i\in\mathcal{N}} |h_i| z'_i \overset{(10)}{\geq} \sum_{i\in\mathcal{N}} |h_i| b = b \sum_{i\in\mathcal{N}} |h_i| \overset{(9)}{=} bS. \tag{12}$$

Since $aS < bS$, we get $\sum_{i\in\mathcal{P}} h_i z'_i < \sum_{i\in\mathcal{N}} |h_i| z'_i$, which is a contradiction ($\perp$) with Equation (8).

($\Leftarrow$) Assume that $\mathrm{rank}(\mathrm{E}') = |\mathcal{V}_{\text{docIDs}}|$, meaning the span of E' is all of $\mathbb{R}^{|\mathcal{V}_{\text{docIDs}}|}$, which, when viewed as the space of logits, produces all probability distributions over $\mathcal{V}_{\text{docIDs}}$. In other words, for any $p \in \Delta^{|\mathcal{V}_{\text{docIDs}}|-1}$ there exists a

$\varphi' \in \mathbb{R}^{n+1}$ such that $\sigma(\mathrm{E}'\,\varphi') = p$. But $\sigma(\mathrm{E}'\,\varphi') = \sigma(\mathrm{E}_{\text{docIDs}}\,\varphi'_{1:n} + \varphi'_{n+1}\mathbf{1}) = \sigma(\mathrm{E}_{\text{docIDs}}\,\varphi'_{1:n})$, where the first equality follows from the definition of E' and the second from the shift-invariance of softmax. Since we assume our ARR has infinite capacity, given a context c it can produce $\varphi(c) = \varphi'_{1:n}$, and hence $\sigma(\mathrm{E}_{\text{docIDs}}\,\varphi(c)) = p$. □

A.2. Proof of Proposition 3.3

Proposition 3.3 (Arbitrary Permutations over Tokens)**.** *An infinite-capacity ARR can produce any permutation over tokens in $\mathcal{V}_{docIDs}$ via constrained decoding iff* $\mathrm{rank}(\mathrm{E}') = |\mathcal{V}_{docIDs}|$.

Proof. (⇒) The proof is entirely analogous to the forward proof of Proposition 3.2, with the exception that the contradiction targets *permutations* in which all token indices for which h takes negative values *precede* all token indices for which h takes negative values.

(⇐) This is an immediate consequence of Proposition 3.2 — if the ARR can produce any probability distribution over $\mathcal{V}_{\text{docIDs}}$, it can also produce any ranking over $\mathcal{V}_{\text{docIDs}}$. □

B. Hyperparameter Settings

B.1. Generative

For the generative experiments, we take Mistral-7B-v0.3-it (Jiang, 2024) as the base model for all fine-tuning experiments in this paper, with the following fine-tuning hyperparameters:

- **Optimizer**: Adafactor (Shazeer & Stern, 2018) with $\beta_1 = 0.9$
- **Learning Rate**: 1×10^{-5}
- **Learning Rate Schedule**: linear warmup for 50 steps followed by a cosine decay.
- **Dataset-Specific Settings** ($10k$ steps):
  - **WordNet**
    - **Max Input length**: 1024
    - **Global Batch Size**: 64, accumulated across replicas
    - Top-1 docID ($\lambda(r) = \mathbb{I}_{r=1}$): 7.65 epochs over $84k$ examples
    - Each docID ($\lambda(r) = \frac{1}{r^\alpha}$ or $\lambda(r) = \frac{n_q - r + 1}{n_q}$): 0.91 epochs over $700k$ examples
  - **ESCI**
    - **Max Input Length**: 8192
    - **Global Batch Size**: 32, accumulated across replicas
    - Top-1 docID ($\lambda(r) = \mathbb{I}_{r=1}$): 3.76 epochs over $85k$ examples
- **Weight Decay**: no weight decay
- **Gradient Clipping**: gradient norm clipped to 1.0

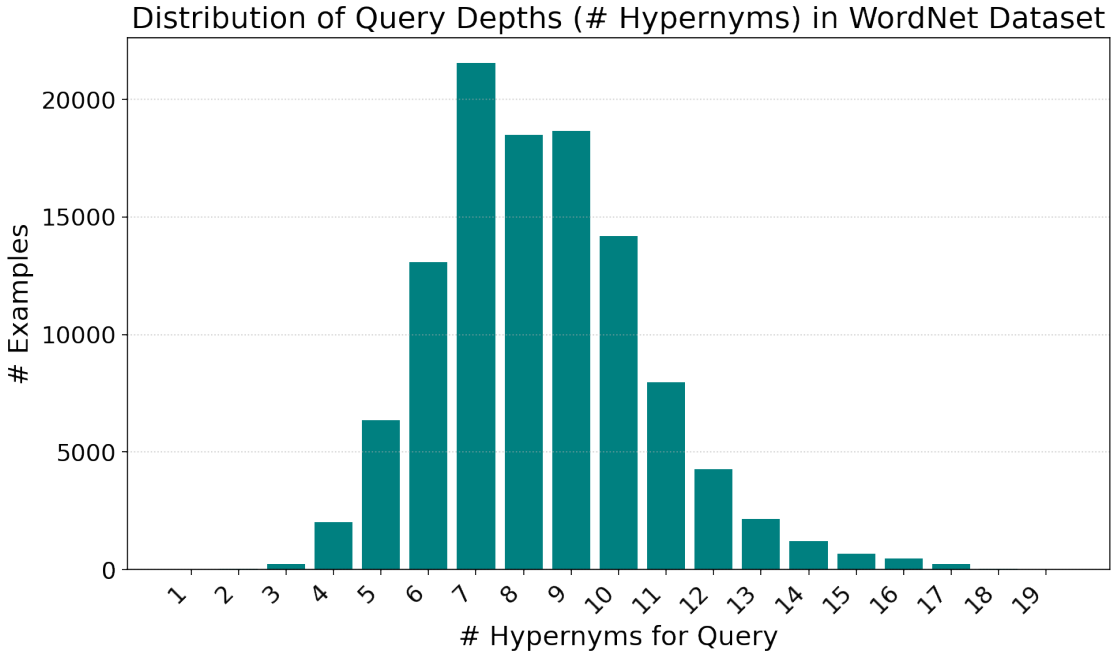
B.2. Dual Encoders

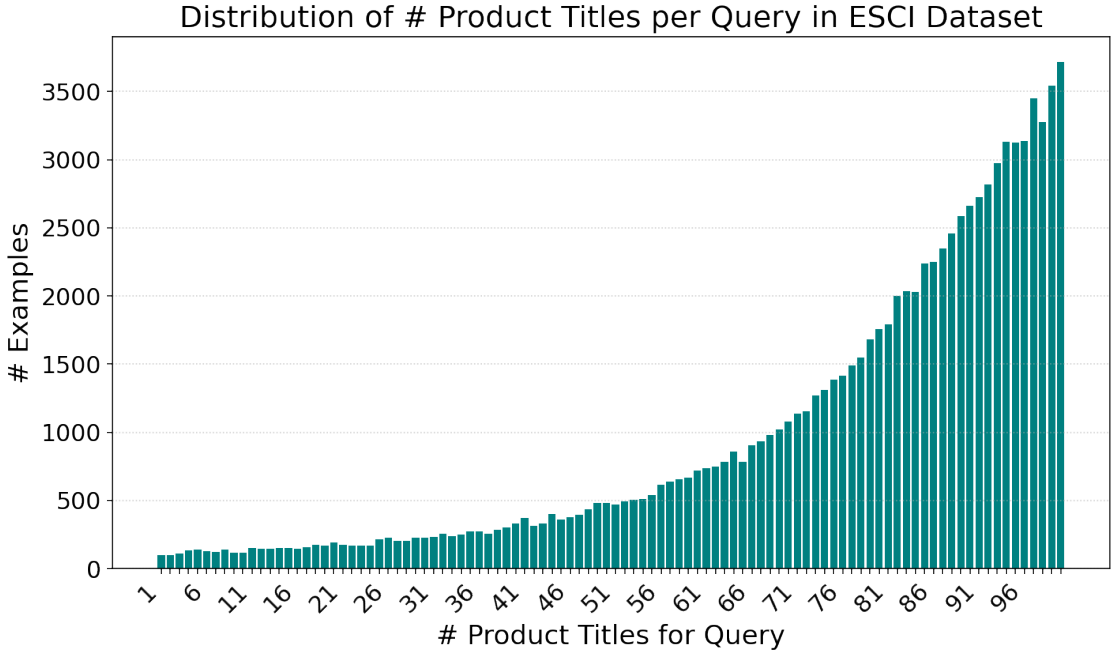
For Dual Encoders we train for 50k steps with a batch size of 2k. We use SGD optimizer with a learning rate of 1.0 and momentum of 0.9. The embeddings are normalized by projecting each row of the embedding table to the unit ℓ_2 sphere after each SGD update.

B.3. Cross Encoders

For Cross Encoders we train for 20k steps with a batch size of 2k. We use ADAM optimizer with a learning rate of 0.001 and weight decay of 0.001. All weight matrices in MLP are initialized with an i.i.d. Gaussian of zero mean and a variance of $1/(2n)$. All bias terms in MLP are initialized as zero vectors.

C. Dataset Statistics





D. Zero-Shot Prompt Examples

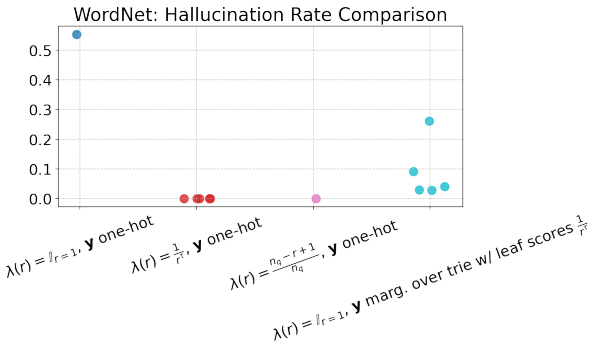
D.1. WordNet Prompt

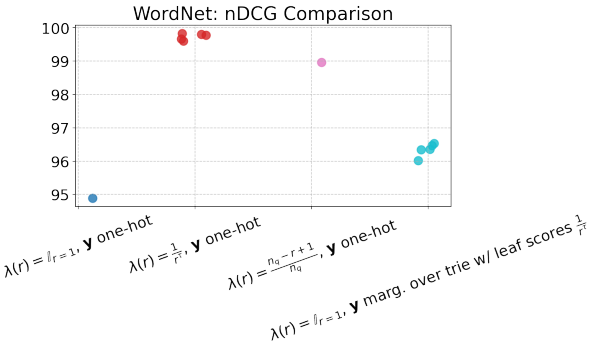
```
You are a helpful assistant that can answer questions about WordNet.  You
are given a noun synset from WordNet and a set of hypernyms for that synset.
The hypernyms are separated by the delimiter '||'.  Your task is to output
the most specific hypernym for the synset out of the given set of hypernyms.
Synset:  deer.n.01 Hypernyms:  object.n.01 || whole.n.02 || chordate.n.01 ||
physical_entity.n.01 || mammal.n.01 || ruminant.n.01 || vertebrate.n.01 ||
placental.n.01 || ungulate.n.01 || even-toed_ungulate.n.01 || animal.n.01 ||
entity.n.01 || living_thing.n.01 || organism.n.01 Most specific hypernym:
```

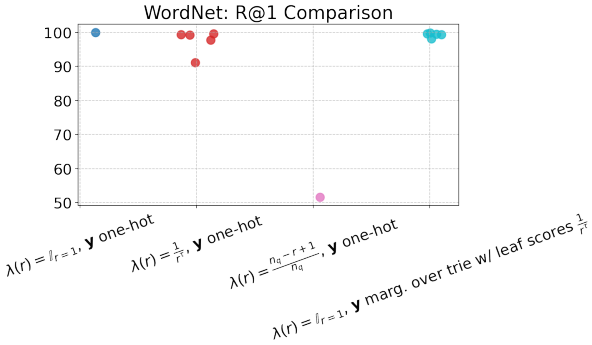
D.2. ESCI Prompt

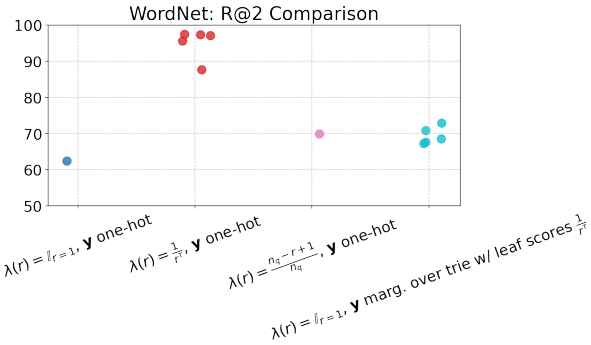
```
You are a helpful assistant that can answer questions about the ESCI shopping
queries dataset.  You are given a query and a list of product titles for that
query.  Each product title is preceded by its docID in square brackets, and the
product titles are separated by the delimiter '||'.  Your task is to generate
the docID of the most relevant product title to the query out of the given
set of product titles and their docIDs.  Query:  aukey usb c Product titles:
[25,36,39] Syntech USB C to USB Adapter, 2 Pack USB C to USB3 Adapter,USB Type
C to USB,Thunderbolt 3 to USB Female Adapter OTG Cable || [25,69,59] SSK USB C
10Gbps Hub, 4-in-1 SuperSpeed USB 10Gbps Type C Multiport Adapter with 2 USB C
2 USB A 3.1/3.2 Gen2 10Gbps Ports,USB C Dock for iMac/MacBook/Pro/Air/Surface
Pro and More Type C Devices || [25,36,26] AUKEY Focus 63W USB C Charger 60W
PD Charger Power Delivery 3.0 [ GaN Power ] Fast Charger Dual Port USB C Wall
Charger for MacBook Pro Air 13" 15" iPhone 11, Pro,Max SE, Google Pixel 4 3 XL,
Switch || [25,59,45] USB Type C Female to USB Male Adapter USB to Type C Laptops
Adapter for iPhone 11 Pro Max Airpods iPad for Samsung Galaxy S20 Ultra Google
Pixel 2Pack || [25,59,2] USB C Cable,Aupek 10FT USB Type C Cable Nylon Braided
Fast Charger Cord for Nintendo Switch, Google Pixel,Samsung Galaxy Note 8 S9 S8
S8 Plus S9(Purple) || [25,39,59] AUKEY USB Car Charger, 18w Quick Charge 3.0,
Flush Fit Cell Phone Adapter for iPhone 12 Pro Max/11 Pro Max/XS/XR, Samsung
Galaxy Note 9 / S9 / Note 10 / S10, and More || [26,2,77] AUKEY Mechanical
Keyboard Blue Switch, 104-Key RGB Backlit Gaming Keyboard with Customizable
Lighting Effects, Aluminium USB Wired Keyboard for Gaming and Typing || [25,2,46]
AUKEY Power Strip with 4 AC Outlets and 4 USB Charging Ports, 5-Foot Extension
Cord for Smartphone, Laptop, Tablet, Home, Office and More (White) || [25,36,97]
AUKEY Focus iPhone Fast Charger 30W 2-Port USB C Charger for iPhone 12/12 Mini/12
Pro Max, PD 3.0 Fast Charger, USB C Wall Charger for iPhone 11 Pro Max/8 Plus,
Pixel 5, MacBook Air, iPad Pro, Switch || [36,25,31] USB C Cable, 2Pack 3ft 5ft
|| [25,26,46] AUKEY USB C Cargador con GAN, Cargador de Pared USB con 60W Power
Delivery 3.0, Compatible con MacBook Pro 13", iPhone 11 Pro, DELL XPS 13, HP
Spectre, Lenevo Thinkpad, Nintendo Switch || [25,36,45] AUKEY USB C Adapter, [3
Pack] USB C to USB 3.0 Adapter Compatible with MacBook Pro 2017/2016 , Google
Chromebook Pixelbook , Samsung Galaxy S9 S8 S8+ Note8, Google Pixel 2/2XL - Black
|| [25,36,39] USB C to USB Adapter (3 Pack), Warmstor Type C Female to USB 3.0 A
Male Converter Support Data Sync & Charging Compatible with iPhone, iPad, Samsung,
Pixel, Laptops, PC, Power Banks, Chargers Most relevant product title:
```

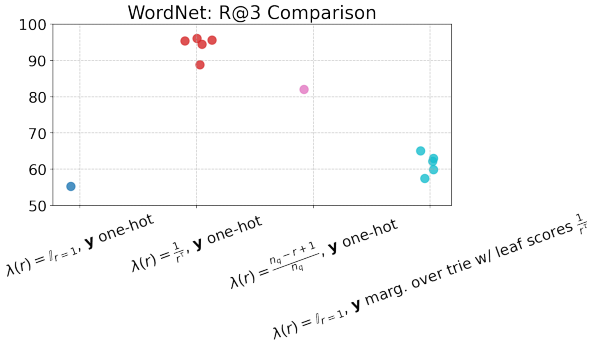
E. WordNet: Table 1 Visualizations

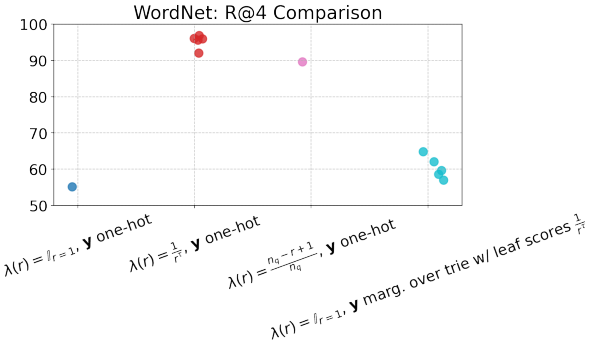


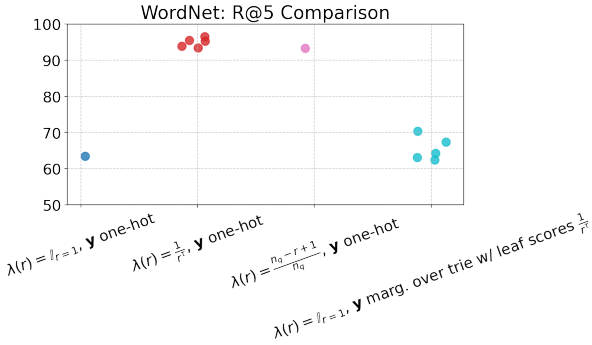












F. WordNet: Combined Item-and-Token Loss Results

We provide supplemental results for fractional item-level reweighting *combined with* the marginalized-over-trie target distributions $\mathbf{y}(r, t)$, i.e., leveraging the full expressivity of Equation (3). We take the temperature $\alpha = 2$ which yielded best performance for R@4 and R@5 in Table 1, and sweep over a range of β temperatures for assigning scores to docID nodes in the prefix tree to be summed over.

In order to construct $\mathbf{y}(r, t)$ for docID of rank r, we only use docIDs $d_r, d_{r+1}, \ldots, d_{n_q}$ in the trie creation process. For example, if we are considering the contribution of the 3rd-ranked docID, which is already down-weighted by $\lambda(3) = \frac{1}{3^\alpha}$, we want $\mathbf{y}(r, t)$ to smooth over the top-3rd, top-4th, top-5th docID paths in the prefix tree without the target distributions being skewed by the top-1st and top-2nd docIDs.

		CVR (↓)	nDCG (↑)	R@1 (↑)	R@2 (↑)	R@3 (↑)	R@4 (↑)	R@5 (↑)
$\lambda(r) = \frac{1}{r^2}$ $\mathbf{y}$ marg. over trie w/ leaf scores $\frac{1}{r^\tau}$	$\beta = 1$	0.0%	99.12	73.38	83.37	86.83	90.08	91.51
	$\beta = 2$	0.02%	99.74	92.36	94.45	94.73	95.39	**96.11**
	$\beta = 3$	0.0%	99.71	94.28	94.1	94.49	95.46	96.06
	$\beta = 4$	0.0%	99.7	94.9	94.3	94.28	95.53	95.67
	$\beta = 5$	0.0%	99.74	95.92	94.54	95.17	95.38	95.7
	$\beta = 6$	0.0%	99.77	97.68	**96.29**	**95.67**	95.7	95.69
	$\beta = 7$	0.0%	**99.83**	97.82	95.74	94.83	**95.79**	95.93
	$\beta = 9$	0.0%	99.76	**98.32**	95.72	94.86	94.85	95.19
	$\beta = 25$	0.0%	99.71	97.38	94.42	94.64	94.82	95.05

Table 3. WordNet results for combined item-and-token loss (evaluated over 5000 (query, targets)-examples)

Note that as $\beta \to \infty$, the target distributions $\mathbf{y}(r, t)$ looks more and more like one-hots and the results in Table 3 should converge to those obtained with just item-level reweighting $\lambda(r) = \frac{1}{r^\alpha}, \alpha = 2$ in Table 1. While the nDCG score in Table 3 improves as β increases from 1 to 7, unfortunately this trend seems to stop for larger β and we do not see it surpass the $\lambda(r) = \frac{1}{r^2}$ results with $\mathbf{y}$ one-hots from Table 1 at any point. We leave optimizing for the "sweet spot" combination of item-level and token-level hyperparameters to future work.