SPD: SINGLE PASS DECODING FOR GENERATIVE RERANKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) achieve state-of-the-art generative ranking quality, but the ranking they produce must be *decoded*, and autoregressive decoding spends one sequential forward pass per emitted token. We observe that the only tokens a ranker must emit are the N ordinal values, corresponding to the number of positions available to rank, naming the items in ranked order, and that this narrow, permutation-structured output format admits decoding strategies which are much more efficient than left-to-right generation. We introduce SPD (Single Pass Decoding), a format-specialized decoding strategy that decodes all N ordinals in $O(1)$ forward passes. SPD reads an $N \times K$ item-position score matrix off the LLM's prefill hidden states with a lightweight self-attention head, then decodes the ordinals as the optimal bipartite assignment of that matrix via the Hungarian algorithm, yielding a valid permutation by construction rather than by repair, without enumerating candidate prefixes or re-querying the LLM. Through a systematic study of training signals and backbone adaptation, we show that LoRA-based fine-tuning combined with autoregressive LLM ranking distillation reaches **28 ms** end-to-end inference, a speed-up of $64\times$ end-to-end over the reasoning teacher while maintaining ranking quality on par. We provide a complete ablation decomposing the contributions of architecture, training signal, and backbone adaptation. Our framework connects generative ranking to combinatorial optimization, opening a path toward other $O(1)$-decode mechanisms for real-time ranking.

1 INTRODUCTION

Generative ranking, producing a complete ordering of N candidates jointly rather than scoring each independently, is critical for recommendation, advertising, and search. Autoregressive LLMs have recently emerged as powerful generative rankers, achieving superior quality by jointly reasoning over all candidates in context (Sun et al., 2023). The output of such a ranker is a permutation, and the model must *emit* that permutation as tokens: concretely, the N ordinal values that name the items in ranked order. Ranking is therefore an unusually *format-specialized* generation problem: the output alphabet is $\{1, \ldots, N\}$, the output length is known before decoding begins, and every value must appear exactly once. It is this structure that determines which decoding strategies are available, and it is the axis along which we organize both the prior work and our contribution.

Autoregressive decoding spends one sequential forward pass over the KV cache per emitted token, so decoding the N ordinals uses $O(N \cdot T)$ passes, where T is the number of tokens per item identifier. This is a bottleneck of the *decoder*, not of the model: the prefill that computes the candidate representations is already parallel and already optimized in its computational complexity. As an example, in the experiments for this paper, a teacher with an explicit reasoning trace reaches 1807 ms per request on an A100 GPU, of which 28 ms is prefill. We specifically target *late-stage re-ranking*, in which a modest candidate set retrieved by an upstream system must be ordered by relevance under strict latency constraints.

Decoding strategies. Because the ordinals are what must be decoded, the design space is a space of *decoders* over a fixed backbone. Autoregressive decoding emits them left to right, one forward pass per token: exact and requiring no architectural change, but with a pass count that grows with the slate. Parallel decoders, non-autoregressive rerankers and multi-token predictors (Ren et al.,

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

2024), and diffusion-based rerankers (Liu et al., 2026), emit all positions at once and refine them over several steps, reducing the pass count but not to one, and without guaranteeing that the emitted ordinals form a permutation. We ask whether the ordinals can be decoded in a single pass:

Can a ranker decode all N ordinals in $O(1)$ forward passes and still recover the ranking that an autoregressive LLM computes over hundreds of sequential passes?

We answer yes, with SPD, which exploits a connection between ranking and optimal assignment. The key insight is that an LLM's prefill hidden states, computed in a single parallel forward pass over all candidates, already encode the comparative information that the ordinals express, via the backbone's self-attention layers. Autoregressive decoding re-serializes that information one token at a time; SPD instead decodes it in place. A lightweight self-attention head turns the prefill states into an $N \times K$ item–position score matrix, and the Hungarian algorithm (Kuhn, 1955) decodes all N ordinals from it as the optimal bipartite assignment, with a valid permutation guaranteed by construction.

Contributions.

1. **A formulation of ordinal decoding as optimal assignment on hidden states.** We cast the emission of the N item ordinals as maximum-weight bipartite matching over an $N \times K$ item–position score matrix whose weights are read directly off the model's hidden states rather than off the output logits. This is the formulation that makes exact combinatorial decoding possible: the Hungarian algorithm recovers the optimal permutation at inference, from one pass. Output validity is guaranteed by construction rather than learned.
2. **SPD, an $O(1)$-pass decoding strategy for generative ranking.** A single prefill over all candidates yields per-item hidden states; a lightweight self-attention head maps them to the score matrix; the Hungarian solver decodes the N ordinals from it. Decoding therefore takes a constant number of forward passes, independent of N, removing the sequential decode dependency that dominates autoregressive ranking latency.
3. **An empirical study on proprietary data and an open dataset.** On our proprietary dataset, SPD achieves a $64\times$end-to-end speed-up over the reasoning autoregressive teacher while maintaining ranking performance. This total gain decomposes into $20.5\times$ from removing reasoning tokens (1807 ms to 88 ms) and $3.1\times$ from replacing the remaining autoregressive decode with SPD (88 ms to 28 ms). We also validate our approach on the open-source Amazon Beauty dataset and find similar speedups. We further ablate three scoring-head configurations and isolate the contributions of backbone adaptation and the self-attention layers.

2 Background and Related Work

2.1 LLM-Based Ranking and Generative Retrieval

Sun et al. (2023) introduced RankGPT, showing that LLMs can rerank generatively, decoding permutations autoregressively with a sliding window. Tay et al. (2022) pioneered generative retrieval with the Differentiable Search Index, encoding document identifiers directly in model parameters, and De Cao et al. (2021) showed autoregressive entity generation is viable for structured retrieval. Most relevant to our work, Reddy et al. (2024) proposed FIRST, which uses first-decoded-token logits for ranking. This is the closest prior approach to constant-pass decoding, but it reads the ordering out of the output token distribution rather than out of the hidden states. SPD keeps the same output as all of these, the item ordinals, and changes only how they are decoded.

2.2 Decoding Strategies for Generative Ranking

Beyond autoregressive decoding, several lines of work reduce the number of sequential forward passes. We organize them as autoregressive (AR), single-token decoding (STD), speculative/parallel decoding (SD), and single-forward-pass (SFP) methods.

Autoregressive listwise decoding. RankGPT (Sun et al., 2023) decodes permutations autoregressively with a sliding window; RankVicuna and RankZephyr (Pradeep et al., 2023a;b) distill

RankGPT-style permutations into open 7B models with progressive multi-pass windows. These are the quality ceiling for listwise generation but require $O(L)$ passes for L emitted tokens plus multi-round sliding windows (Sun et al., 2023).

Single-token decoding (STD). FIRST (Reddy et al., 2024) fine-tunes with a learning-to-rank loss and reranks from the logits of only the first generated identifier, completing reranking in a single pass. STD is the closest prior approach to constant-pass decoding, but it approximates the ranking from first-token signal and degrades severely on tail positions (Du et al., 2026). SPD keeps the same output—the item ordinals—but reads the full permutation out of prefill hidden states rather than out of the first-token distribution.

Speculative and parallel decoding. Speculative decoding (Leviathan et al., 2023) drafts with a small model and verifies with the target; blockwise decoding (Stern et al., 2018) predicts multiple future tokens with extra heads; Medusa (Cai et al., 2024) adds parallel decoding heads with tree-attention verification; EAGLE (Li et al., 2024), PDM (Santilli et al., 2023), and lookahead decoding (Fu et al., 2024) extend this family, while LayerSkip (Elhoushi et al., 2024) reduces target-model computation through early-exit inference. These methods reduce the cost of left-to-right generation, but remain multi-round and their pass count scales with output length. They are therefore orthogonal and potentially composable with SPD, which replaces sequential generation with a single-pass assignment. Tree attention (Spector & Ré, 2023; Cai et al., 2024) enables parallel verification of multiple prefixes in one pass; ONE-PASS (Du et al., 2026) uses it to pre-verify subset combinations ($N! \to 2^N \to K \cdot 2^{N/K}$) and reconstruct full-ranking probability with Möbius inversion plus hierarchical self-attention, as a tuning-free approximation to autoregression. SPD differs: it never enumerates prefixes and never re-queries the LLM—the permutation is decoded once, from hidden states, via exact assignment.

Efficient listwise variants. PE-Rank (Liu et al., 2025) compresses each passage to a single embedding token and constrains decoding to identifier tokens, improving prefill and decode efficiency. This is orthogonal to SPD: PE-Rank shortens the prefill input while SPD eliminates the sequential decode.

2.3 Zero-Shot Prompting Taxonomy: Pointwise, Pairwise, Listwise, Setwise

Zero-shot LLM ranking is commonly organized into four prompting families (Zhuang et al., 2024; Qin et al., 2024). *Pointwise* methods score each document independently: MonoT5 recasts relevance scoring as the generation of binary target tokens (Nogueira et al., 2020), an approach also used in general LLM evaluation via yes/no likelihoods (Liang et al., 2023), or query likelihood $P(\text{query} \mid \text{doc})$ (Sachan et al., 2022). They cost $O(N)$ calls and batch well but need logit access and calibration, and lag on effectiveness. *Pairwise* methods (PRP (Qin et al., 2024); origins in duoBERT (Nogueira et al., 2019) and duoT5 (Pradeep et al., 2021)) prompt with document pairs and aggregate via AllPairs $O(N^2)$, sorting $O(N \log N)$, or sliding-K $O(N)$, with heap/bubble sort plus top-k early stopping. They are the most effective zero-shot family but the most expensive. *Listwise* methods prompt with a window of documents and generate the ranked label list (Sun et al., 2023; Ma et al., 2023), using bottom-up sliding windows over multiple passes; they need generation only and no calibration, but are bottlenecked by context length, generation cost, and sensitivity to initial order (Zhuang et al., 2024). *Setwise prompting* (Zhuang et al., 2024) generalizes pairwise sorting to best-of-c comparisons per LLM call, reducing heap sort to $O(k \log_c N)$ and bubble sort to $O(k \cdot N/(c-1))$, plus a logit-based listwise-likelihood variant that avoids generating the label list. It gives the best effectiveness–efficiency trade-off among zero-shot methods, but still issues many LLM calls, requires no training, and provides no permutation guarantee. SPD occupies a different point: a trained, single-pass decoder that emits a valid permutation by construction (Hungarian solver) rather than accelerating a generate-and-sort loop.

Delineation. Unlike ONE-PASS, which verifies enumerated subset-prefixes through the LLM in one wide pass, SPD never enumerates prefixes and never re-queries the LLM: the ranking is read off hidden states once. Unlike Setwise, which accelerates zero-shot sorting by comparing $c > 2$ documents per LLM call, SPD replaces the generate-and-sort loop entirely with a learned head plus exact assignment.

2.4 Optimal Transport and Differentiable Permutation Learning

Our training procedure builds on differentiable relaxations of discrete permutations. Cuturi (2013) introduced entropic regularization for efficient optimal transport, providing the algorithmic foundation for the Sinkhorn operator, and Mena et al. (2018) extended this to learning latent permutations with Gumbel-Sinkhorn networks. Alternative differentiable sorting approaches include SoftSort (Prillo & Eisenschlos, 2020) and differentiable sorting networks (Petersen et al., 2021). Our contribution is connecting these permutation-learning tools to LLM hidden states.

2.5 Knowledge Distillation for Ranking

Hinton et al. (2015) introduced soft-target distillation. For ranking, Hofstätter et al. (2020) demonstrated cross-architecture distillation from cross-encoders to bi-encoders, and Pradeep et al. (2023b) distilled generative reranking ability into smaller models. We differ in two ways: we distill full permutations via a Sinkhorn cross-entropy objective, and our ablation reveals a previously undocumented interaction between signal richness and backbone capacity.

2.6 Hungarian Assignment in Deep Learning

DETR (Carion et al., 2020) uses bipartite matching between predicted and ground-truth detections for set-based loss computation. Berthet et al. (2020) provided theoretical foundations for learning with differentiable perturbed combinatorial optimizers. In mechanism design, Duetting et al. (2019) learn optimal multi-item auctions via neural networks, and for ad allocation Zheng et al. (2025) introduced non-autoregressive generative auctions with position-query cross-attention. Critically, we use Hungarian matching *not* for loss computation as in DETR, but as the inference-time decoder that replaces autoregressive generation. Similarly, DiffuRank (Liu et al., 2026) applies the Hungarian algorithm to the decoded output of the LLM. We instead apply it to the hidden states, as the decoder itself, so the permutation constraint is enforced during decoding rather than restored afterwards.

Despite this breadth of prior work, no existing method decodes a complete ranking from an LLM backbone in a constant number of forward passes, at real-time latency, with near-lossless quality and a permutation guaranteed by construction.

3 Problem Formulation

3.1 Notation and Setup

Let u denote a user context and $C = \{c_1, \ldots, c_N\}$ a slate of N candidate items retrieved by an upstream system. The re-ranking task is to produce a permutation $\pi \in \Pi_N$ that orders items by relevance. An autoregressive teacher generates π via T sequential decode passes, incurring $O(N \cdot T)$ latency. We seek a student f_θ that decodes the same N ordinals, and hence the same π, in $O(1)$ decode passes, i.e. a single forward pass.

Latency decomposition. For an autoregressive ranker, inference computation is dominated by the decode phase: each token requires a full forward pass over the KV-cache. The following examples are from the experiments in this paper. With reasoning enabled $T \approx 657$ tokens; without reasoning $T \approx 39$. In both cases the prefill phase, which processes all N candidates in parallel, takes a fixed $t_{\text{prefill}} \approx 28\,\text{ms}$. Our key observation: if the ranking can be extracted from the prefill hidden states alone, the $O(T)$ *sequential* decode computational complexity collapses to $O(1)$, the ordinals are still decoded, but not one at a time.

3.2 Decoding a Ranking as Optimal Assignment

Define a bipartite graph $G = (I \cup J, E)$ where $I = \{1, \ldots, N\}$ indexes items and $J = \{1, \ldots, K\}$ indexes rank positions. Let $\mathbf{M} \in \mathbb{R}^{N \times K}$ be a score matrix in which $\mathbf{M}_{ij}$ is the affinity of item i for rank position j. The optimal ranking is the maximum-weight bipartite matching

$$\pi^* \;=\; \arg\max_{\pi \in \Pi_N} \sum_{i=1}^{N} \mathbf{M}_{i,\pi(i)}. \tag{1}$$

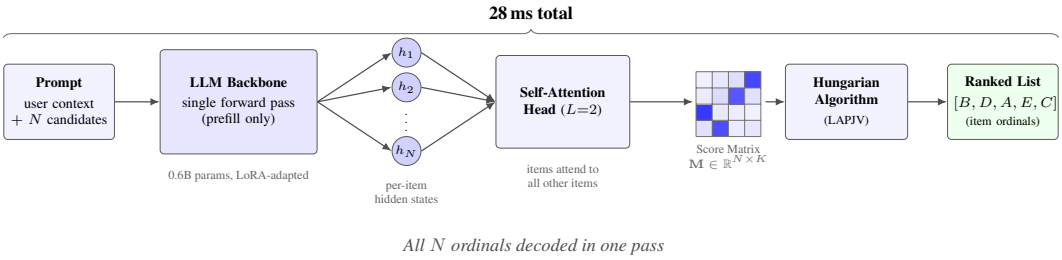


Figure 1: SPD architecture. A single prefill pass over all N candidates yields per-item hidden states, which the self-attention head maps to an $N \times K$ score matrix. The Hungarian algorithm decodes all N item ordinals from that matrix in one shot: $O(1)$ forward passes, no sequential decode. End-to-end latency is 28 ms.

Equation (1) is the linear assignment problem (LAP), solvable exactly in $O(N^3)$ by the Hungarian algorithm (Kuhn, 1955), or by the Jonker–Volgenant variant (LAPJV) (Jonker & Volgenant, 1987) which achieves $O(N^2)$ average-case complexity via shortest augmenting paths.

Complexity comparison. For $N = 50$, the autoregressive teacher requires $T = 657$ sequential forward passes. The Hungarian solver requires $O(N^3) \approx 1.25 \times 10^5$ arithmetic operations, roughly $8\,\mu$s on CPU. The full SPD pipeline (one prefill plus one Hungarian solve, i.e. $O(1)$ passes for all N ordinals) is therefore bounded by the computational complexity of prefill, making the combinatorial optimization effectively free (Section 5.5).

3.3 Differentiable Relaxation via Sinkhorn

The Hungarian algorithm is non-differentiable: it returns a discrete permutation. For gradient-based training we require a continuous relaxation. We adopt the Sinkhorn operator (Cuturi, 2013; Mena et al., 2018). When $N \neq K$, we normalize the rows and columns to satisfy the corresponding prescribed marginal constraints; in the square case $N = K$, this reduces to the usual doubly-stochastic constraint over the Birkhoff polytope. In the $N = K$ case:

$$\mathbf{S}^{(0)} = \exp(\mathbf{M}/\tau), \qquad \mathbf{S}^{(l+1)} = \mathbf{D}_r^{-1}\,\mathbf{S}^{(l)}\,\mathbf{D}_c^{-1}, \tag{2}$$

where $\mathbf{D}_r$ and $\mathbf{D}_c$ are diagonal normalization matrices enforcing unit row and column sums, respectively, with $\mathbf{D}_c$ computed after the row normalization. The temperature $\tau > 0$ controls relaxation sharpness ($\tau \to 0$ recovers a permutation matrix).

Training objective. Given a teacher permutation matrix $\mathbf{P}^{\text{teacher}} \in \{0,1\}^{N \times K}$, the Sinkhorn cross-entropy loss is

$$\mathcal{L} \;=\; -\sum_{i,j} \mathbf{P}_{ij}^{\text{teacher}}\, \log \mathbf{S}_{ij}^{(L_S)}. \tag{3}$$

This provides smooth gradients to $\mathbf{M}_\theta$ while driving the student's output toward a valid permutation. At inference we bypass Sinkhorn entirely and apply the Hungarian algorithm to $\mathbf{M}$ directly, guaranteeing an exact permutation.

4 Method: SPD

4.1 Architecture Overview

SPD comprises three components applied in sequence (Figure 1). Figure 1 depicts the winning configuration (self-attention head, $L = 2$, LoRA-adapted backbone); alternative head variants are ablated in Section 5.4.

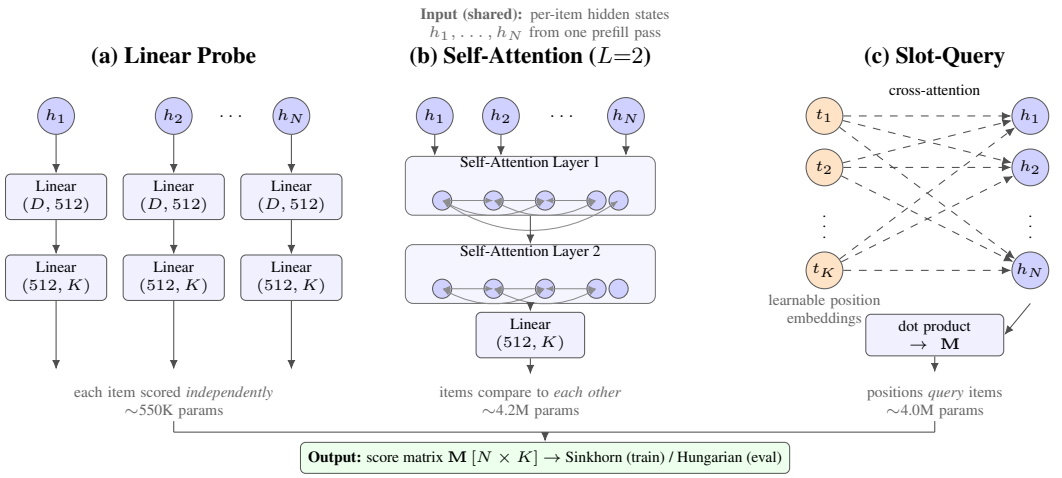


Figure 2: Scoring head variants. (a) A linear probe scores each item independently; all cross-item context arises implicitly from the backbone's self-attention during prefill. (b) The self-attention head performs explicit item-to-item comparison before scoring. (c) The slot-query head inverts the formulation: K learnable position embeddings cross-attend over items. On a frozen backbone (a) is competitive; once the backbone is LoRA-adapted, (b) is strongest.

Backbone (prefill-only, LoRA-adapted). The backbone is a 0.6B-parameter decoder-only transformer (28 layers, hidden dimension 1024, 16 attention heads) pretrained for language modeling and fine-tuned for ranking. It processes the full prompt containing the user context and all N candidates in a single forward pass. We gather the hidden state at each candidate's readout position (the final token of its description), yielding $\{\mathbf{h}_1, \ldots, \mathbf{h}_N\} \in \mathbb{R}^{N \times D}$.

Scoring head (self-attention, $L = 2$). A lightweight transformer encoder with $L = 2$ self-attention layers operates over the N candidate vectors, allowing each item to attend to all others and thereby perform explicit pairwise comparison. A linear projection maps each refined representation to K rank-position scores, producing $\mathbf{M} \in \mathbb{R}^{N \times K}$.

Assignment decoder (Hungarian). Unlike DETR (Carion et al., 2020), which uses Hungarian matching during training for loss computation, we apply the Hungarian algorithm at inference as the decoding mechanism itself. Applied to $\mathbf{M}$ it yields the optimal one-to-one assignment: a valid permutation decoded in one shot. The Hungarian solver is thus SPD's decoder in the same sense that a sampling loop is an autoregressive decoder, it is the component that turns model outputs into the N emitted ordinals. During training, Equation (3) supplies gradients.

4.2 Scoring Head Variants

We study three head architectures to isolate the contribution of cross-item comparison (Figure 2, Table 1).

4.3 Training: Offline Teacher Ranking Distillation

Training proceeds in two phases.

Phase 1 (offline, one-time). The teacher, a well-trained autoregressive generative ranking model, decodes a ranking for every training slate via greedy decoding in direct-output mode (no chain-of-thought). These target permutations are computed once and stored. We discard the small number of teacher outputs containing repeated or missing ordinals.

Phase 2 (student training). The student minimizes Equation (3) against the pre-computed teacher permutations. There is no online interaction between teacher and student; the labels are fixed. This

Head	Key property	Params
Linear probe	Each item scored independently; all cross-item context arises implicitly from backbone self-attention	550K
Self-attention (L=2)	Explicit pairwise comparison: each item attends to all others before scoring	4.2M
Slot-query	Dual formulation: K learnable position embeddings cross-attend over items ("what fills rank 1?")	4.0M

Table 1: Scoring head variants. All operate on identical frozen or LoRA-adapted backbone hidden states.

follows from the decoding strategy: SPD's decoder has no intermediate state and no rollout, $\mathbf{M}$ is conditioned only on the prompt, and the Hungarian solver is a deterministic function of $\mathbf{M}$, so the train-time and inference-time input distributions are identical by construction. There is no exposure bias for on-policy distillation to correct and no student trajectory for the teacher to score, and Equation (3) is already a whole-permutation objective rather than a per-token surrogate. One offline teacher pass over the training set therefore suffices.

4.4 Backbone Adaptation via LoRA

We apply Low-Rank Adaptation (LoRA, Hu et al., 2022) with rank 64 to all linear projections in the backbone. This allows the backbone to reshape its hidden states to encode ranking-relevant information while preserving pretrained knowledge. As Section 5.3 shows, this adaptation is the enabler: without it, even rich teacher signals cannot be absorbed by the scoring head.

5 Experiments

5.1 Setup

Internal dataset. A proprietary re-ranking dataset ($\approx$ 189K training slates, 236K test requests), with up to 150 items per slate. **Teacher:** a well-trained autoregressive generative ranking model (0.6B-parameter decoder-only transformer, fine-tuned for ranking), evaluated both with and without an explicit reasoning trace.

Amazon Beauty dataset. The "beauty" products subset of the 2014 Amazon product dataset (He & McAuley, 2016). Slates comprise 50 candidate items to rank, with 21,245 training and 1,118 evaluation slates. **Teacher:** 32B Qwen model used for ranking. We also test a 0.6B Qwen model for comparison.

Student. The same 0.6B architecture, adapted via LoRA ($r = 64$). Trained via SFT from the respective teacher for each dataset.

Metrics. AUC, Recall@$\{1, 10\}$, NDCG@1. **Hardware.** NVIDIA A100 80GB; LAPJV solver on CPU.

5.2 Main Results

We present the main results and a series of ablations below; note that ablations were performed on successive datasets. Table 2 compares SPD against both teacher configurations. Removing the reasoning trace reduces teacher latency from 1807 ms to 88 ms, a 20$\times$ reduction in decode passes, at essentially no reduction in quality, confirming that reasoning tokens are not load-bearing for ranking. SPD decodes the same ordinals in a single pass, reaching 28 ms: 64$\times$ faster than the reasoning teacher and 3.1$\times$ faster than the no-reasoning teacher. Note what the second comparison isolates. The no-reasoning teacher has already stripped every non-essential token and decodes only the ranking itself; the remaining 3.1$\times$ is attributable purely to the decoding strategy, not to shorter output. Quality is lossless on list-level metrics: SPD is indistinguishable from both teachers on NDCG@1 (0.1791 SPD vs. 0.1776 / 0.1779) and Recall@1 (0.1652 SPD vs. 0.1634 / 0.1635). AUC is the metric where SPD retains 99.9% of the teacher's quality. We attribute the list-level

Model	Latency	Speed-up	Decode passes	AUC	Recall@1	Recall@10	NDCG@1
Internal Dataset							
Teacher (with reasoning)	1807 ms	1.0×	768	0.5911	0.1634	0.7877	0.1776
Teacher (no reasoning)	88 ms	20.5×	39	**0.5912**	0.1635	0.7892	0.1779
SPD (ours)	**28 ms**	**64.5×**	**1**	0.5907	**0.1652**	**0.7912**	**0.1791**
Amazon Beauty Dataset							
Qwen3-32B	5075 ms	1.0×	78	**0.6292**	**0.1384**	0.4232	**0.1384**
Qwen3-0.6B	921 ms	5.5×	26	0.5122	0.0232	0.2062	0.0232
SPD 0.6B (ours)	**113 ms**	**44.9×**	**1**	0.6168	0.1321	**0.4268**	0.1321

Table 2: Main results. Every row decodes the same output: the N item ordinals naming the ranked slate. **Decode passes** counts the sequential forward passes required to do so; for autoregressive teachers under greedy decoding this equals the number of emitted tokens, while SPD decodes the ordinals out of the prefill pass itself, so its count is constant in N. Speed-up is relative to the teacher within each dataset.

Track	Training signal	Backbone	AUC
1	Click labels (binary)	Frozen	0.5187
2	Teacher ranking (full permutation)	Frozen	0.5573
3	Teacher ranking (full permutation)	LoRA (r=64)	**0.5842**

Table 3: Signal richness and backbone adaptation provide complementary gains. Teacher ranking targets improve over click labels with a frozen backbone (Track 2 $>$ Track 1), and adapting the backbone yields a further improvement (Track 3 $>$ Track 2).

gains to two factors: the Hungarian assignment enforces a globally consistent one-to-one ordering that autoregressive decoding only approximates, and distillation across the full training set averages out the teacher's sampling noise (Section 6.1).

5.3 Ablation: Training Signal × Backbone Adaptation

Table 3 shows gains from both richer supervision and backbone adaptation. With a frozen backbone, replacing binary click labels with full teacher rankings improves AUC from 0.5187 to 0.5573 (Track 1 vs. Track 2). Allowing the backbone to adapt to the teacher rankings raises AUC further to 0.5842 (Track 3).

5.4 Ablation: Head Architecture

Table 4 shows our ablations on the three different head architectures described. The self-attention architecture described above performs best. Removing the self-attention layers and replacing them with a linear probe reduces the ranking performance, as does adding learned position-dependent embeddings. On the backbone that is LoRA-adapted, the self-attention head is the strongest configuration, consistent with our interpretation in Section 6.1.

5.5 Solver Overhead

Table 5 decomposes end-to-end latency. The backbone prefill accounts for essentially all of it; the scoring head adds under 0.1 ms and the LAPJV solver 0.008 ms, under 0.03% of the total. This is the central efficiency argument: although the linear assignment problem is $O(N^3)$ in the worst case, at slate sizes of $N \leq 50$ it resolves in microseconds on CPU, fully overlapped by the GPU forward pass. SPD specifically targets this late-stage re-ranking regime rather than retrieval-scale candidate pools. The apparent overhead of introducing a combinatorial solver is therefore illusory in our target setting. More importantly, the decomposition shows that SPD has moved the bottleneck from a *sequential* dependency (autoregressive decode, which cannot be parallelized) to a *parallel* one (a single prefill over all candidates). Decoding all N ordinals has an impact of 0.008 ms of that budget.

Head	AUC	Recall@1
Self-attention, L=2 (SPD)	**0.5907**	**0.1652**
Linear probe	0.5841	0.1650
Slot-query	0.5816	0.1579

Table 4: Ablation on the different scoring head architectures in the ads ranking dataset.

Component	Latency
Backbone prefill, all items (GPU)	27.9 ms
Scoring head, L=2 (GPU)	< 0.1 ms
LAPJV assignment, N=50 (CPU)	0.008 ms
Total	**28 ms**

Table 5: End-to-end latency decomposition. The combinatorial solver contributes under 0.03% of total latency.

Any further latency reduction must come from the backbone itself, via pruning, quantization, or early exit, rather than from the ranking machinery, which is already effectively free.

6 Analysis

6.1 The Capacity Gap

Reproducing a full N-item permutation requires hidden states that encode fine-grained relative preferences between every pair of items. Click labels are binary and supply a simple gradient ("push clicked items up"), whereas teacher rankings provide supervision over the complete ordering. This richer signal improves performance even with a frozen backbone (Track 2 vs. Track 1), but the language-modeling-optimized representations still limit how fully the scoring head can exploit it. LoRA narrows this capacity gap with roughly 4M trainable parameters on a 600M backbone, producing the additional gain in Track 3.

6.2 Structural Guarantees

By construction SPD yields (i) 100% valid outputs, since the Hungarian algorithm always returns a valid permutation; the constraint is discharged inside the decoder rather than repaired after it; (ii) no degenerate solutions, since the one-to-one constraint forbids assigning multiple items to the same position; and (iii) deterministic inference, with no sampling temperature or beam-search artifacts. These properties are guaranteed rather than learned, and they account for part of the list-level advantage over the autoregressive teacher, which can and does emit invalid or repeated orderings.

7 Conclusion

We presented SPD, a format-specialized decoding strategy that decodes a complete ranking, all N item ordinals, in $O(1)$ forward passes, at 28 ms end-to-end. We formalize the use of the Hungarian algorithm to decode a ranking directly from hidden states. We empirically validate this approach on a proprietary dataset and an open-source benchmark, finding near-lossless ranking quality compared to autoregressive LLMs, at a 45–$64\times$ inference speedup. We profile the performance of the different parts of the architecture and our systematic ablations decompose the contributions between the backbone and the self-attention layers.

AI use statement

In this work, we used generative AI tools for implementing methods, specifically, to code our research idea and execute the experiments. We have not used generative AI tools to propose or refine hypotheses, develop theoretical models or conceptual frameworks, formulate mathematical claims,

design or provide feedback on research methodology or experiments, clean and reformat datasets, support qualitative and thematic data analysis, assist with translation, or interpret results. Generating synthetic datasets, providing critical ingredients for proving mathematical claims, and assisting in the writing of proofs are not applicable to this work.

Additionally, we used generative AI tools to create first versions of our scientific figures, edit software code, draft parts of the paper and improve its readability, identify and analyze relevant literature, format references, and iterate on the title and keywords. In short, we used generative AI tools to warm-start the work, then thoroughly vetted and improved the code, figures, and text for the final submission. AI-generated code and text were verified by at least four authors, and the background section was read and verified by at least three. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis R. Bach. Learning with differentiable perturbed optimizers. *CoRR*, abs/2002.08676, 2020. URL https://arxiv.org/abs/2002.08676.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 5209–5235. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/cai24b.html.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (eds.), *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part I*, volume 12346 of *Lecture Notes in Computer Science*, pp. 213–229. Springer, 2020. doi: 10.1007/978-3-030-58452-8_13. URL https://doi.org/10.1007/978-3-030-58452-8_13.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pp. 2292–2300, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/af21d0c97db2e27e13572cbf59eb343d-Abstract.html.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=5k8F6UU39V.

Yingpeng Du, Zhu Sun, Tianjun Wei, and Jie Zhang. ONE-PASS: single forward pass decoding for listwise reranking. In Hakim Hacid, Yoelle Maarek, Francesco Bonchi, Ido Guy, and Emine Yilmaz (eds.), *Proceedings of the ACM Web Conference 2026, WWW 2026, Dubai, United Arab Emirates, originally scheduled for April 13-17, 2026, rescheduled for June 29 - July 3, 2026*, pp. 6057–6066. ACM, 2026. doi: 10.1145/3774904.3792243. URL https://doi.org/10.1145/3774904.3792243.

Paul Duetting, Zhe Feng, Harikrishna Narasimhan, David C. Parkes, and Sai Srivatsa Ravindranath. Optimal auctions through deep learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1706–1715. PMLR, 2019. URL http://proceedings.mlr.press/v97/duetting19a.html.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed A. Aly, Beidi Chen, and Carole-Jean Wu. Layerskip: Enabling early exit inference and self-speculative decoding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 12622–12642. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.681. URL https://doi.org/10.18653/v1/2024.acl-long.681.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. Break the sequential dependency of LLM inference using lookahead decoding. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 14060–14079. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/fu24a.html.

Ruining He and Julian J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In Jacqueline Bourdeau, Jim Hendler, Roger Nkambou, Ian Horrocks, and Ben Y. Zhao (eds.), *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016*, pp. 507–517. ACM, 2016. doi: 10.1145/2872427.2883037. URL https://doi.org/10.1145/2872427.2883037.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL http://arxiv.org/abs/1503.02531.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. Improving efficient neural ranking models with cross-architecture knowledge distillation. *CoRR*, abs/2010.02666, 2020. URL https://arxiv.org/abs/2010.02666.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Roy Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987. doi: 10.1007/BF02278710. URL https://doi.org/10.1007/BF02278710.

Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: speculative sampling requires rethinking feature uncertainty. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 28935–28948. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/li24bt.html.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel J. Orr, Lucia Zheng, Mert Yüksekgönül, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter

Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models. *Trans. Mach. Learn. Res.*, 2023, 2023. URL https://openreview.net/forum?id=iO4LZibEqW.

Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. Leveraging passage embeddings for efficient listwise reranking with large language models. In Guodong Long, Michale Blumestein, Yi Chang, Liane Lewin-Eytan, Zi Helen Huang, and Elad Yom-Tov (eds.), *Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025- 2 May 2025*, pp. 4274–4283. ACM, 2025. doi: 10.1145/3696410.3714554. URL https://doi.org/10.1145/3696410.3714554.

Qi Liu, Kun Ai, Jiaxin Mao, Yanzhao Zhang, Mingxin Li, Dingkun Long, Pengjun Xie, Fengbin Zhu, and Ji-Rong Wen. Diffurank: Effective document reranking with diffusion language models. *CoRR*, abs/2602.12528, 2026. doi: 10.48550/ARXIV.2602.12528. URL https://doi.org/10.48550/arXiv.2602.12528.

Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. Zero-shot listwise document reranking with a large language model. *CoRR*, abs/2305.02156, 2023. doi: 10.48550/ARXIV.2305.02156. URL https://doi.org/10.48550/arXiv.2305.02156.

Gonzalo E. Mena, David Belanger, Scott W. Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=Byt3oJ-0W.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with BERT. *CoRR*, abs/1910.14424, 2019. URL http://arxiv.org/abs/1910.14424.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 708–718. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.FINDINGS-EMNLP.63. URL https://doi.org/10.18653/v1/2020.findings-emnlp.63.

Felix Petersen, Christian Borgelt, Hilde Kuehne, and Oliver Deussen. Differentiable sorting networks for scalable sorting and ranking supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8546–8555. PMLR, 2021. URL http://proceedings.mlr.press/v139/petersen21a.html.

Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. The expando-mono-duo design pattern for text ranking with pretrained sequence-to-sequence models. *CoRR*, abs/2101.05667, 2021. URL https://arxiv.org/abs/2101.05667.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *CoRR*, abs/2309.15088, 2023a. doi: 10.48550/ARXIV.2309.15088. URL https://doi.org/10.48550/arXiv.2309.15088.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *CoRR*, abs/2312.02724, 2023b. doi: 10.48550/ARXIV.2312.02724. URL https://doi.org/10.48550/arXiv.2312.02724.

Sebastian Prillo and Julian Martin Eisenschlos. Softsort: A continuous relaxation for the argsort operator. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7793–7802. PMLR, 2020. URL http://proceedings.mlr.press/v119/prillo20a.html.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. Large language models are effective text rankers with pairwise ranking prompting. In Kevin Duh, Helena

Gómez-Adorno, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, volume NAACL 2024 of *Findings of ACL*, pp. 1504–1518. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-NAACL.97. URL https://doi.org/10.18653/v1/2024.findings-naacl.97.

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md. Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. FIRST: faster improved listwise reranking with single token decoding. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pp. 8642–8652. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.EMNLP-MAIN.491. URL https://doi.org/10.18653/v1/2024.emnlp-main.491.

Yuxin Ren, Qiya Yang, Yichun Wu, Wei Xu, Yalong Wang, and Zhiqiang Zhang. Non-autoregressive generative models for reranking recommendation. In Ricardo Baeza-Yates and Francesco Bonchi (eds.), *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, pp. 5625–5634. ACM, 2024. doi: 10.1145/3637528.3671645. URL https://doi.org/10.1145/3637528.3671645.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. Improving passage retrieval with zero-shot question generation. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pp. 3781–3797. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.EMNLP-MAIN.249. URL https://doi.org/10.18653/v1/2022.emnlp-main.249.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. Accelerating transformer inference for translation via parallel decoding. In Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 12336–12355. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.ACL-LONG.689. URL https://doi.org/10.18653/v1/2023.acl-long.689.

Benjamin Spector and Christopher Ré. Accelerating LLM inference with staged speculative decoding. *CoRR*, abs/2308.04623, 2023. doi: 10.48550/ARXIV.2308.04623. URL https://doi.org/10.48550/arXiv.2308.04623.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 10107–10116, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/c4127b9194fe8562c64dc0f5bf2c93bc-Abstract.html.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. Is chatgpt good at search? investigating large language models as re-ranking agents. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 14918–14937. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.923. URL https://doi.org/10.18653/v1/2023.emnlp-main.923.

Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Prakash Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. Transformer memory as a differentiable search index. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9,*

2022, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/892840a6123b5ec99ebaab8be1530fba-Abstract-Conference.html.

Zuowu Zheng, Ze Wang, Fan Yang, Wenqing Ye, Weihua Huang, Wenqiang He, Teng Zhang, and Xingxing Wang. NGA: non-autoregressive generative auction with global externalities for advertising systems. *CoRR*, abs/2506.05685, 2025. doi: 10.48550/ARXIV.2506.05685. URL https://doi.org/10.48550/arXiv.2506.05685.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (eds.), *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pp. 38–47. ACM, 2024. doi: 10.1145/3626772.3657813. URL https://doi.org/10.1145/3626772.3657813.