Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Are LLMs Reasoning or Ranking? Diagnosing and Mitigating Order Bias in Knowledge Graph Completion

Kamal Hamaz[1], Assia Tebib[1], Mohamed Ali Bouanaka[1]
[1]*LIRE Laboratory, University of Constantine 2 - Abdelhamid Mehri, Constantine, Algeria*
{kamal.hamaz, assia.tebibl, mohamedali.bouanaka}@univ-constantine2.dz

ARTICLE INFO

Received: 30 Dec 2024

Revised: 19 Feb 2025

Accepted: 27 Feb 2025

ABSTRACT

Introduction: Knowledge graphs power modern question-answering, recommendation, and entity-aware language models, but are perennially incomplete. The task of knowledge graph completion (KGC) infers missing triples and is increasingly approached by finetuning a large language model (LLM) to select the correct entity from a candidate list produced by a lightweight embedding retriever. Such systems report strong Hits@1 improvements over the retriever and are interpreted as evidence that the LLM reasons about which candidate matches the query.

Objectives: This paper tests that interpretation directly. We ask two questions. First, are LLM-based discrimination rerankers actually reasoning about candidate identity, or are they exploiting candidate position in the prompt as a shortcut? Second, if the shortcut is present, can it be removed by a simple change to the training procedure?

Methods: We finetune Mistral-7B with QLoRA on FB15K-237 candidate lists produced by a TransE retriever under two training regimes: candidates always in retriever-score order, and candidates independently shuffled in every batch. Each trained model is evaluated under three inference-time orderings of the same candidate set: identity, uniform random permutation, and adversarial reversal. Differences across orderings isolate the reranker's dependence on candidate position rather than candidate content.

Results: The ordered-trained reranker reaches Hits@1 of 0.310 in distribution but collapses to 0.059 under shuffled inference and to 0.011 under reversal (below the 0.05 chance level of guessing uniformly in the candidate list), with residual accuracy falling below the TransE retriever it builds on. The shuffled-trained reranker is essentially flat across the three conditions (0.249, 0.247, and 0.244 Hits@1), reducing the across-condition spread by a factor of about sixty at the cost of approximately six absolute Hits@1 points.

Conclusions: Apparent reasoning by LLM-based KGC rerankers is largely an artifact of candidate-position exploitation. Once the shortcut is blocked, the LLM contributes little beyond the retriever it sits on top of. Robustness to candidate ordering should be a routine evaluation criterion for any LLM-augmented retrieval system, and the framing of such systems as reasoning models should be tempered with a corresponding control.

Keywords: knowledge graph completion, large language models, link prediction, shortcut learning, position bias, reranking, parameter-efficient finetuning.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

1. INTRODUCTION

Knowledge graphs, large collections of (head, relation, tail) factual triples, power modern question-answering, recommendation, and entity-aware language models [1, 2]. They are perennially incomplete, which motivates the knowledge graph completion (KGC) or link prediction task of inferring missing tails (and heads) from observed triples. Classical KGC approaches train an embedding model that maps each entity and relation to a low-dimensional vector and scores triples by a closed-form function of these vectors; TransE [3], DistMult [4], ComplEx [5], RotatE [6] and several others (see §2.1) remain competitive baselines on the standard benchmarks [7].

More recently, a wave of methods has cast KGC as a downstream task for large language models (LLMs): scoring triples with masked language models [8], generating the missing entity with sequence-to-sequence LLMs [9, 10], and using an LLM as a *discrimination reranker* on top of a lightweight embedding retriever [11, 12, 13, 14]. The third pattern is currently the most accurate. An embedding retriever produces the top-m candidate entities; the candidate list is rendered into a natural-language prompt; the LLM, finetuned with LoRA [15] or QLoRA [16] over a strong base such as LLaMA-2 [17] or Mistral [18], is asked to choose the correct candidate. DIFT [11] is a recent example, reporting Hits@1 figures that exceed the embedding retriever by 4–8 percentage points.

The implicit framing of this line of work is that the LLM *reasons* about the semantic match between query and candidate. This paper tests that interpretation directly. The candidate list is presented in retriever-score order, a strong positional cue: the correct entity is far more likely to appear near the top of the list than at the bottom. An LLM trained on such prompts can therefore reach high Hits@1 by an entirely position-mediated mechanism (learning to pick whichever entity appears early), without consulting candidate identities at all. DIFT itself reports a brief ablation [11, §4.6] noting that shuffling candidates at inference drops Hits@1 from 0.616 to 0.233 on WN18RR; the implications of that observation are not pursued in their work. The question is precise: are these systems *reasoning* about candidates, or simply *ranking* by position?

The question fits two established themes. Shortcut learning [19] is the well-documented preference of deep networks for surface regularities over intended task structure [20, 21, 22, 23]. Position bias in LLMs is the equally well-documented sensitivity of LLMs to the order of items in their input, in in-context learning [24, 25, 26], long-context reading [27], and multiple-choice questioning [28, 29, 30]. KGC discrimination prompts sit at the intersection of these themes.

This paper investigates the order-bias question empirically and proposes a simple mitigation. We (i) formalise the candidate-order perturbation framework (§3); (ii) finetune Mistral-7B with QLoRA on FB15K-237 [7] in two training regimes and evaluate each model under three inference orderings (§4); (iii) show that the ordered-trained reranker collapses by 81% under shuffled inference and 96% under reversal, with residual accuracy below its retriever (§5.2, §5.3); (iv) show that shuffled-candidate training eliminates the shortcut (§5.4, §5.5); and (v) argue that robustness to candidate ordering should be a routine evaluation criterion for any LLM-augmented retrieval system.

2. RELATED WORK

This section reviews five strands of prior work that bear on our diagnosis. We first situate our retriever in the KG-embedding landscape (§2.1), then trace the line of LLM-based KGC approaches culminating in discrimination-style rerankers (§2.2). The phenomenon we observe is a special case of shortcut learning (§2.3) and is closely related to position bias in LLMs (§2.4). We close with permutation-invariant architectures and the broader retrieval-augmented setting (§2.5).

2.1 Knowledge graph embedding methods

Translation-based methods score a triple (h, r, t) by treating the relation as a translation in entity space: TransE [3] is the canonical instance; RotatE [6] generalises to complex-valued rotations. Bilinear methods include RESCAL [33], DistMult [4], ComplEx [5] and TuckER [31]. ConvE [32] and CompGCN [34] capture higher-order interactions; NodePiece [35] reduces parameter count via entity tokenisation. These methods remain competitive on the standard benchmarks [7] and serve as upstream retrievers in subsequent LLM-augmented systems.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

2.2 LLMs for knowledge graph completion

Three lines of work integrate LLMs with KGC. KG-BERT [8] uses a masked language model to score triple plausibility. KGT5 [10] and KG-S2S [9] finetune T5-family models to generate the missing entity. Discrimination-based approaches, KG-LLM [13], KICGPT [12], AutoKG [14] and DIFT [11], finetune a large LLM with LoRA to select the correct entity from a candidate list; this is currently the most accurate pattern and is the architecture studied here. DIFT [11], the most direct precursor of our methodology, finetunes LLaMA-2-7B with LoRA on TransE [3] candidate lists, reporting state-of-the-art Hits@1 on FB15K-237 [7] and WN18RR [32]. Its authors note in passing [11, §4.6] that shuffling candidates at inference time drops Hits@1 from 0.616 to 0.233 on WN18RR, but do not investigate the phenomenon further; the present paper does. SimKGC [36] applies contrastive learning on pre-trained language models without a discrimination step. Pan et al. [2] survey the broader LLM and KG landscape.

2.3 Shortcut learning in NLP

Shortcut learning [19] is the systematic preference of neural models for surface heuristics that correlate with labels in training. Niven and Kao [22] show BERT achieving near-human accuracy on argument-reasoning purely through lexical cues. McCoy et al. [21] show NLI models relying on syntactic heuristics rather than entailment. Gururangan et al. [20] and Poliak et al. [23] document annotation artifacts in SNLI/MNLI that let hypothesis-only classifiers reach over 65% accuracy. The KGC reranking shortcut documented here is consistent with these findings: a feature that correlates with the label in training (candidate position) is exploited in place of the intended task (semantic matching).

2.4 Position bias in LLMs

Three recent works are most directly relevant. Pezeshkpour and Hruschka [29] document multiple-choice accuracy fluctuations of up to 75% as the order of options changes across several LLMs and benchmarks, attributing the effect to positional bias and uncertainty between top candidates. Zheng et al. [30] show that twenty LLMs are systematically biased toward specific option-ID tokens (e.g., “A”) in MCQs and propose an inference-time debiasing method (PriDe). Koo et al. [28] benchmark six cognitive biases, including order bias, in LLMs used as evaluators and find that scaling up model size does not reduce them. KGC discrimination prompts as used by DIFT [11] and its predecessors [12, 13, 14] are MCQs over candidate entities, so these findings directly predict an order-bias effect in KGC. Related earlier work shows the same broader pattern in other settings: in-context demonstration order can shift few-shot accuracy by 30 percentage points [24]; LLMs exhibit characteristic biases that warrant calibration [26]; demonstration content matters less than its structural placement [25]; and information in the middle of long contexts is systematically under-used [27].

2.5 Permutation-invariant architectures and retrieval

Deep Sets [37] and Set Transformer [38] provide architectural primitives for processing unordered sets, useful when input order is uninformative. An autoregressive LLM cannot be easily made permutation-invariant without major structural changes, motivating our training-time mitigation. Retrieval-augmented generation [1] and dense passage retrieval [39] introduced the retriever and reader pattern of which discrimination rerankers are a specialisation; the same diagnostic question arises in those settings too.

3. Preliminaries

This section fixes the notation used throughout the paper, following standard conventions of the KGC literature.

3.1 Knowledge graph and link prediction

A knowledge graph is a triple $G = (E, R, T)$, where E is a finite set of entities, R is a finite set of binary relations, and $T \subseteq E \times R \times E$ is a set of observed factual triples. Each triple $(h, r, t) \in T$ expresses that entity $h \in E$ stands in relation $r \in R$ to entity $t \in E$. KGs are typically incomplete; the knowledge graph completion (KGC) task is to recover missing triples. A tail-prediction query q is a partial triple in which the tail has been removed, written $q = (h, r, ?)$; the set of all test queries is Q. Given a query q with ground truth $t \in E$, a KGC model produces a ranking of E by some scoring function. We adopt the standard filtered protocol of [3, 7]: for each query, the rank of t is computed after removing

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

other known true tails for the same (h, r) from the candidate set. The reported metrics are MRR and Hits@k for k ∈ {1, 3, 10}:

$$\mathrm{MRR} = \frac{1}{|Q|}\sum_{q\in Q}\frac{1}{\mathrm{rank}(q)}, \qquad \mathrm{Hits@}k = \frac{1}{|Q|}\left|\left\{q\in Q : \mathrm{rank}(q)\le k\right\}\right|.$$

By construction, Hits@1 ≤ Hits@3 ≤ Hits@10. The metric of primary interest in this paper is Hits@1, since it isolates the question of whether the model selected the correct entity as its single most plausible answer.

3.2 Embedding retriever

An embedding-based KGC model, referred to here as a retriever, is a function that assigns a real-valued score to every candidate triple. The canonical instance is TransE [3], which represents each entity and relation by a learned vector in $\mathbb{R}^d$ and scores triples as:

$$M_E(h, r, t) = -\left\|\mathbf{e}_h + \mathbf{e}_r - \mathbf{e}_t\right\|_1,$$

where e_x ∈ $\mathbb{R}^d$ denotes the embedding of entity or relation x. For a query q = (h, r, ?), evaluating M_E(h, r, ·) over all e ∈ E induces a ranking of the entity set. Retaining the top-m entries yields the candidate list

$$C(q) = \left\langle e_{(1)},\ e_{(2)},\ \ldots,\ e_{(m)}\right\rangle,$$

an ordered tuple of length m in which the i-th element e_(i) is the entity with the i-th highest score under M_E after filtering. Throughout this paper m = 20.

3.3 Discrimination reranker

A discrimination reranker [11, 12, 13, 14] is a model M_R that consumes a query q together with its candidate list C and emits a predicted position in C:

$$M_R : Q \times E^m \to \{1, \ldots, m\}, \qquad \hat{e} = e_{(M_R(q, C))}.$$

The predicted entity ê is then placed at the top of the retriever ranking and metrics are recomputed. By construction, M_R can only ever predict an entity already in C, which eliminates the grounding step needed by generation-based KGC methods [9, 10].

3.4 Candidate-order perturbation

The central experimental device of this paper is a perturbation of the candidate list at inference time. Let S_m denote the symmetric group on {1, ..., m}. Given π ∈ S_m we define

$$\pi(C) = \left\langle e_{(\pi(1))},\ e_{(\pi(2))},\ \ldots,\ e_{(\pi(m))}\right\rangle.$$

We consider three families of π. The identity π = id leaves C unchanged. A uniform random permutation π ~ Uniform(S_m) destroys all positional information. The reversal permutation π = rev, with π(i) = m + 1 − i, is adversarial. Crucially, π does not alter the candidate set; only the order in which candidates appear inside the prompt fed to the reranker. Any difference in M_R(q, π(C)) across choices of π therefore isolates the reranker's dependence on candidate position rather than candidate content.

4. METHODOLOGY

Our pipeline implements the retriever and reranker pattern at full scale. The retriever, reranker, training regime, and inference-time candidate ordering are independently manipulable. Figure 1 gives the pipeline view; each subsection below also shows the relevant step of a worked FB15K-237 example.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

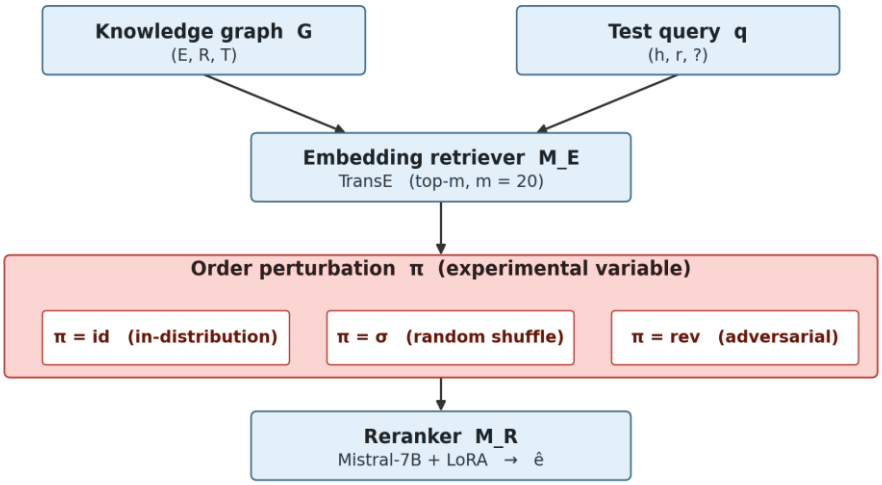

Figure 1. Pipeline architecture. A test query q = (h, r, ?) is scored by the embedding retriever M_E against every entity in E; the top-m ranked entities form the candidate list C. The perturbation block π, the experimental variable, reorders C; the reranker M_R consumes the resulting prompt P(q, π(C)) and emits the predicted entity ê.

4.1 Retriever training

We train TransE [3] on FB15K-237 [7] with embedding dimension d = 200, margin γ = 6.0, Adam optimiser with learning rate 10^{-3}, batch size 4096, and 500 epochs. Training minimises the standard margin-ranking loss $L = \Sigma \max(0, \gamma + M\_E(h, r, t') - M\_E(h, r, t))$ over positive triples $(h, r, t) \in T$ and corrupted negatives $(h, r, t') \in T^-$. For each positive triple we sample one negative by uniformly corrupting either the head or the tail. Entity embeddings are L^2-renormalised to unit norm at every step.

4.2 Candidate generation

For every query in the validation and test splits we compute the filtered TransE scores against every entity, sort in descending order, and retain the top m = 20 entries. The choice m = 20 follows DIFT [11] and balances coverage (the answer is in the top-20 of TransE for 63.8% of FB15K-237 test queries) against prompt length. Queries whose answer lies outside the top-20 are retained for evaluation but discarded during reranker training, since no supervision signal is available. To make the procedure concrete we trace one query through the pipeline. The machine identifiers (MIDs) and the relation /film/film/language below follow the FB15K-237 schema; the human-readable labels in parentheses are illustrative and are intended to keep the data flow legible to the reader, not as assertions about specific Freebase mappings.

Step 1 — Test triple

(/m/0bs1g , /film/film/language , /m/02h40lc)

head = (film) relation = language tail = (language)

The query is constructed by hiding the tail:

Step 2 — Query construction

q = (/m/0bs1g , /film/film/language , ?)

TransE then scores every entity and we retain the top-20:

Step 3 — TransE scoring

score every entity $e \in E$ by $f(h, r, e) = - || e\_h + e\_r - e\_e ||\_1$

rank all 14,505 entities, retain top m = 20 (filtered)

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Step 4 — Candidate list C (TransE-score order)

1. /m/04306rv (language A) −4.81
2. /m/02h40lc (gold tail ← correct) −4.92
3. /m/0653m (language B) −5.07
4. /m/02bjrlw (language C) −5.18
5. /m/064_8sq (language D) −5.21

...

20. /m/01gp_x (language R) −6.14

4.3 Prompt construction

Each candidate list is rendered into a multiple-choice prompt with letter labels $\Lambda = \{A, B, \ldots, T\}$. The reranker is asked to emit a single letter; the predicted letter decodes to a position in C, which decodes to an entity. This restriction guarantees that the model can only ever predict an entity actually in C, eliminating the grounding step required by generation-based methods [9, 10]. The prompt template is shown below; placeholders in curly braces are filled in per query, and the candidate list C populates positions 1–20 in column-major order under letters A through T:

Prompt template

Head: {h.mid}

Relation: {r.path}

Choose the most plausible tail entity:

A) {C[1]}	F) {C[6]}	K) {C[11]}	P) {C[16]}
B) {C[2]}	G) {C[7]}	L) {C[12]}	Q) {C[17]}
C) {C[3]}	H) {C[8]}	M) {C[13]}	R) {C[18]}
D) {C[4]}	I) {C[9]}	N) {C[14]}	S) {C[19]}
E) {C[5]}	J) {C[10]}	O) {C[15]}	T) {C[20]}

Answer with a single letter. Answer:

The same template is used for training and for inference. Under ordered-candidate training the positions 1–20 are filled in TransE-score order; under shuffled-candidate training they are filled in a per-batch permuted order (§4.5). Continuing the example, the instantiated prompt for our test query is:

Step 5 — Prompt P(q, C)

Head: /m/0bs1g

Relation: /film/film/language

Choose the most plausible tail entity:

A) /m/04306rv	F) /m/03_9r	K) /m/06b_j	P) /m/05qjt
B) /m/02h40lc	G) /m/01jb8r	L) /m/02ztjwg	Q) /m/01c7y
C) /m/0653m	H) /m/04gfy7	M) /m/06nm1	R) /m/04h9h
D) /m/02bjrlw	I) /m/02hxhz	N) /m/0jzc	S) /m/02h40b1
E) /m/064_8sq	J) /m/02hxc3j	O) /m/064_v8q	T) /m/01gp_x

Answer with a single letter. Answer:

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4.4 Reranker

We use Mistral-7B-v0.3 [18] as the reranker (Figure 2), with its native BPE tokenizer (32K vocabulary). Base weights are held at 4-bit NF4 precision with double quantisation [16]. LoRA adapters [15] of rank r = 64 and α = 16 are inserted on the query-projection (q_proj) and value-projection (v_proj) matrices of every self-attention layer; dropout 0.1, fp16 compute, gradient checkpointing. The maximum sequence length is 1024 tokens, which comfortably accommodates the 20-candidate prompt (typical length ≈ 350–500 tokens). For the running example, the reranker emits a single letter, decoded back to an entity:

Step 6 — Reranker output

M_R(q, C) = B

decoded entity ê = /m/02h40lc (gold tail) ✓ correct

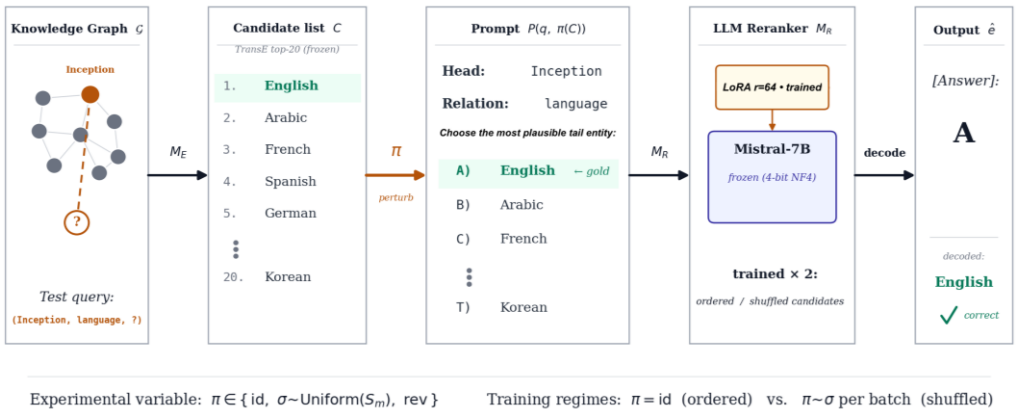

Figure 2. Fine-tuning procedure. A LoRA adapter (r = 64) on frozen Mistral-7B is trained by cross-entropy on the gold letter ℓ under two regimes: ordered (π = id) and shuffled (π ∼ σ resampled per batch).*

4.5 Two training regimes

The reranker is trained twice in two regimes that differ only in how candidates are presented during finetuning. Under *ordered-candidate training* candidates are always in TransE-score order, the regime under which similar systems [11, 12, 13, 14] are typically trained. Under *shuffled-candidate training* an independent permutation π ∼ Uniform(S_m) is applied to every example in every batch; the gold position varies uniformly across {1, ..., m}, so position alone is no longer informative of correctness and the model must consult candidate identities to minimise the loss. Training data is drawn from the FB15K-237 validation split, yielding 17,526 supervised examples (the training split is memorised by TransE and would place the gold answer at position 1 of nearly every example). Both runs share hyper-parameters: 2 epochs, batch size 4 with gradient accumulation 4 (effective batch 16), learning rate 2×10^{-4} with cosine schedule and 3% warmup, AdamW with weight decay 0.01, gradient clip 1.0. The loss is the standard cross-entropy on the answer-letter token; the prompt tokens are masked from the loss.

4.6 Inference-time evaluation protocol

Each trained reranker is evaluated under all three π ∈ {id, σ, rev}. The identity ordering is applied as-is. For σ, a single fresh permutation is sampled per test query with a fixed global random seed, so all reported numbers are reproducible. For each query we feed the perturbed candidate list to M_R, restrict next-token logits to the m letter tokens in Λ, take the argmax, and decode the selected letter back to an entity. This entity is placed at the top of the retriever ranking and the filtered Hits@1/3/10 and MRR are recomputed.

5. EXPERIMENTS

All experiments below were conducted on a single G4 GPU. §5.1 describes the dataset and the TransE retriever baseline; §5.2 quantifies the order-bias collapse under ordered-candidate training; §5.3 decomposes the reranker's

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

headline Hits@1; §5.4 reports the shuffled-candidate-training mitigation; §5.5 places both regimes side by side on Hits@1; §5.6 extends to MRR, Hits@3 and Hits@10; §5.7 summarises.

5.1 Dataset and retriever baseline

We evaluate on FB15K-237 [7], the standard 14,541-entity, 237-relation filtering of Freebase with 272,115 training triples, 17,535 validation triples and 20,466 test triples. Twenty-eight test triples and nine validation triples involve entities unseen at training and are excluded by standard pre-processing, leaving 20,438 test and 17,526 validation queries. Table 1 lists the statistics; Table 2 reports the TransE retriever baseline on the test split, with figures consistent with published TransE numbers on FB15K-237.

Dataset	|E|	|R|	Train	Validation	Test
FB15K-237	14,541	237	272,115	17,535	20,466

Table 1. Statistics of the FB15K-237 benchmark [7].

Retriever	MRR	Hits@1	Hits@3	Hits@10
TransE [3]	0.364	0.267	0.402	0.552

Table 2. TransE [3] retriever baseline on FB15K-237 (test, filtered).

5.2 The order-bias collapse (ordered-candidate training)

We first train the Mistral-7B reranker under the ordered-candidate regime and evaluate it under each of the three inference orderings. Table 3 reports the metrics. Under matched in-distribution inference the reranker reaches Hits@1 = 0.310, improving on TransE by +0.043 absolute. Under shuffled inference Hits@1 falls to 0.059, a relative drop of 81%. Under adversarial reversal it collapses to 0.011, below the chance level 1/m = 0.05 of guessing uniformly in the candidate list. Same trained model, same test queries; only the candidate order differs. This effect is qualitatively the same as the WN18RR observation of DIFT [11, §4.6], here measured systematically on FB15K-237 and across two further perturbations.

Inference order π	MRR	Hits@1	Hits@3	Hits@10
Ordered (π = id, in-distribution)	0.387	0.310	0.418	0.555
Shuffled ($\pi \sim \sigma$)	0.136	0.059	0.113	0.329
Reversed (π = rev, adversarial)	0.056	0.011	0.023	0.088

Table 3. Mistral-7B reranker trained on candidates in TransE order, evaluated on the FB15K-237 test split. Same trained model in all three rows; only the candidate-order perturbation π differs.

5.3 Decomposing the reranker's accuracy

If Hits@1 under shuffled inference is treated as a lower bound on what the reranker has learned about candidate content (since the position signal is now uninformative), then the gap between ordered and shuffled inference is the contribution of the position shortcut. Figure 3 makes the decomposition explicit. The position-blind component (0.059) is well below the TransE retriever (0.267). Of the reranker's 0.310 in-distribution Hits@1, only 0.059 remains when position is removed; the other 0.251 (81% of its accuracy) is attributable to candidate ordering rather than to learned ranking knowledge.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

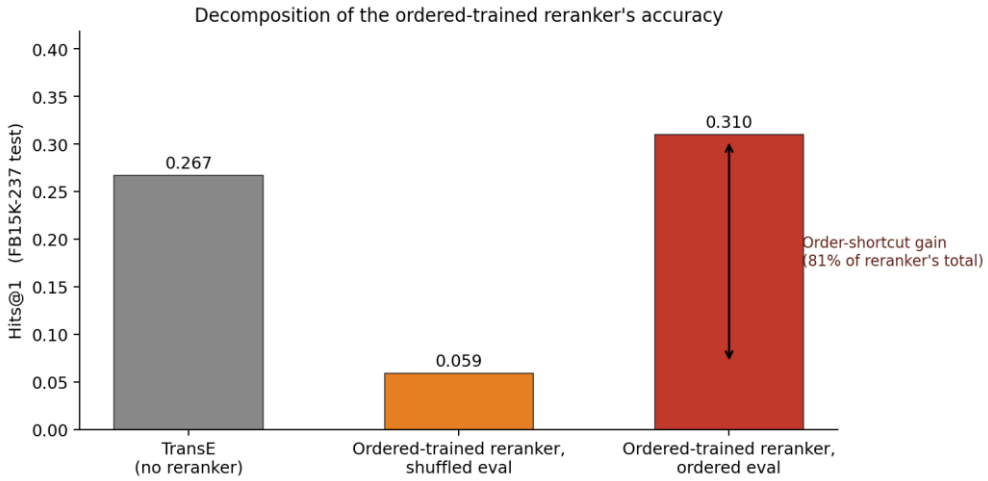

Figure 3. Decomposing the ordered-candidate-trained Mistral-7B reranker's Hits@1 on FB15K-237. Grey: TransE retriever baseline. Orange: the same reranker evaluated under shuffled inference (its position-blind component). Red: the same reranker under ordered (in-distribution) inference. The vertical bracket indicates the order-shortcut gain, which accounts for 81% of the red bar.

5.4 Mitigation via shuffled-candidate training

Having quantified the shortcut, we ask whether it can be eliminated by a single change to the training procedure. We retrain the same Mistral-7B base with identical hyper-parameters, validation split and optimiser steps but with candidates independently permuted in every batch. The resulting model is evaluated under all three inference orderings; Table 4 reports the metrics. Hits@1 of 0.249, 0.247 and 0.244 under ordered, shuffled and reversed inference: a max-minus-min spread of 0.005 across the three conditions, compared with 0.299 for the ordered-trained model. The shortcut has been eliminated.

Inference order π	MRR	Hits@1	Hits@3	Hits@10
Ordered (π = id)	0.366	0.249	0.443	0.567
Shuffled (π ~ σ)	0.308	0.247	0.308	0.444
Reversed (π = rev)	0.280	0.244	0.261	0.319

Table 4. Mistral-7B reranker trained on candidates independently shuffled every batch, evaluated on the FB15K-237 test split. The three rows are within 0.005 Hits@1 of each other across very different inference orderings, demonstrating order-invariance.

5.5 Side-by-side comparison (Hits@1)

Figure 4 places all six combinations (two training regimes × three inference orderings) on a single horizontal axis. Red bars correspond to the ordered-trained reranker (solid for ordered evaluation, diagonally hatched for shuffled, cross-hatched for reversed) and exhibit the collapse documented in §5.2. Blue bars correspond to the shuffled-trained reranker and are essentially flat across the three inference conditions. The dashed grey line marks the TransE retriever's Hits@1 of 0.267.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

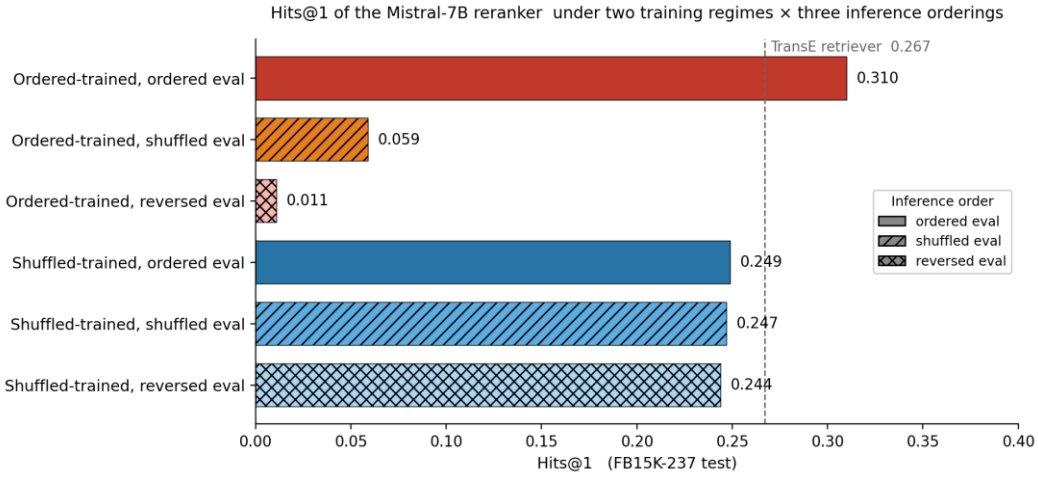

Figure 4. Hits@1 on the FB15K-237 test split for both training regimes (red and blue rows) under each of the three inference-time candidate orderings. Solid: ordered evaluation. Diagonal hatch: shuffled evaluation. Cross-hatch: reversed evaluation. The dashed grey line marks the TransE retriever's Hits@1 of 0.267.

5.6 Beyond Hits@1: MRR, Hits@3, Hits@10

Figure 5 reports the remaining three metrics across the six conditions. The Hits@1 pattern holds for MRR (ordered-trained: 0.387 → 0.136 → 0.056; shuffled-trained: 0.366 → 0.308 → 0.280) and is sharper for Hits@3 (0.418 → 0.113 → 0.023 vs. 0.443 → 0.308 → 0.261). Hits@10 degrades more gracefully because ten of the twenty positions are within the top half of any permutation; even so the ordered-trained reranker drops by 0.226 absolute (0.555 → 0.329) under shuffling. The shuffled-trained reranker matches or exceeds the TransE baseline on Hits@3 in all three test conditions (0.443 / 0.308 / 0.261 vs. 0.402) and on Hits@10 in two of three (0.567 / 0.444 / 0.319 vs. 0.552). Once the shortcut is removed, the reranker's ranking quality at higher k is comparable to the retriever's, but its top-1 selection adds little.

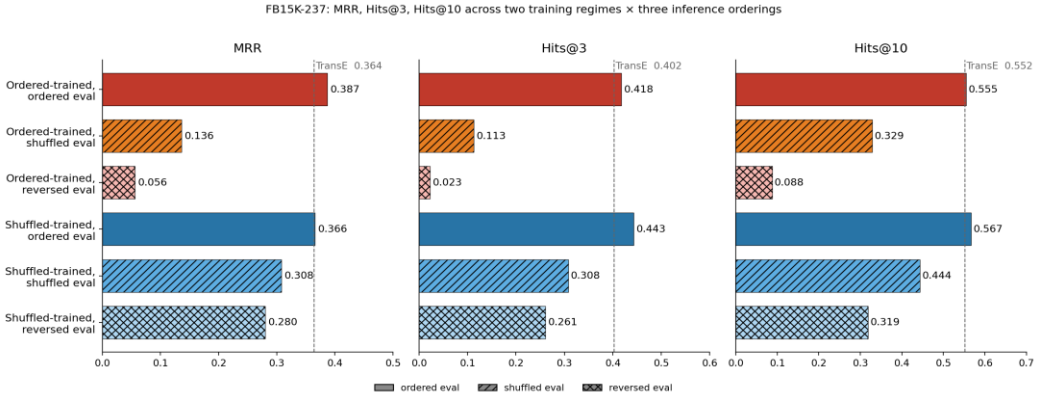

Figure 5. MRR, Hits@3 and Hits@10 on the FB15K-237 test split, across the six combinations of training regime and inference-time candidate ordering. Colour and hatch encoding follow Figure 3. Dashed grey lines mark the TransE retriever baseline in each panel.

5.7 Takeaway

An ordered-candidate-trained Mistral-7B reranker on FB15K-237 loses 81% of its in-distribution Hits@1 under shuffled inference and 96% under reversal, with residual position-blind accuracy (0.059) below the TransE retriever it builds on (0.267). The headline +0.043 improvement is overwhelmingly an artifact of candidate position. Shuffling candidates every batch eliminates the shortcut, reducing the across-condition Hits@1 spread by a factor of about sixty at the cost of approximately six absolute Hits@1 points. The honest, order-invariant Hits@1 of the LLM reranker is close to that of the retriever alone; the apparent gap is the shortcut.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

6. DISCUSSION

This section interprets the experimental findings. §6.1 returns to the title's reasoning-or-ranking question. §6.2 turns the answer into a concrete evaluation recommendation. §6.3 extends the diagnosis beyond KGC, and §6.4 states the scope and limitations of the claims.

6.1 Reasoning, or ranking?

The discrimination-reranker setting answers the reasoning-or-ranking question directly. An LLM trained on retriever-ordered candidates does neither task in the sense usually claimed. In-distribution accuracy is real, but residual accuracy with position information removed drops below the embedding retriever. Conditional on retriever ordering, the LLM re-emits a positional prior with little independent knowledge added. When trained with shuffled candidates the LLM achieves near-retriever accuracy under all three orderings (order-invariant but offering only modest reranking benefit). Neither regime exhibits the rich semantic reasoning that the published interpretation of LLM-based KGC implies; the *reasoning* framing of such systems is, on this evidence, incompatible with their behaviour under a basic order control.

6.2 Implications for evaluation

LLM-based discrimination rerankers should be evaluated under at least one non-identity candidate ordering as a matter of course. The +0.043 Hits@1 improvement of an ordered-trained reranker over its TransE retriever is, by our measurement, almost entirely an artifact of candidate position; reporting it without a robustness check overstates the LLM's contribution. The same logic applies to retrieval-augmented systems more broadly [1, 39]. DIFT [11] observes the shuffled-inference drop on WN18RR but draws no methodological consequence; our results indicate that the consequence is significant.

6.3 Broader implications

The mechanism is not specific to KGC. Any system that asks an LLM to choose from an ordered candidate list, including multiple-choice QA [28, 29, 30], retrieval reranking, and ranked recommendation, is vulnerable to the same shortcut if the candidate ordering correlates with the label in training. The mitigation is generic: present training-time candidates in random order, forcing the model to consult their identities. Permutation-invariant architectural choices [37, 38] are an alternative when feasible, although autoregressive LLMs cannot be easily made order-invariant without major structural changes.

6.4 Limitations

Our experiments are confined to FB15K-237 [7] with Mistral-7B [18] as the reranker. Different base models or KGC benchmarks might modulate the magnitude of the shortcut, although the same mechanism (position correlating with correctness in the training distribution) is generic to the discrimination paradigm. We do not finetune the retriever; stronger retrievers, for example RotatE [6], NodePiece [35], or SimKGC [36], would shift the absolute ceiling but would not eliminate the shortcut. We have not investigated whether the shortcut diminishes with model scale; MCQ-bias work at the largest scales [30] suggests it does not.

7. CONCLUSION

We diagnosed an order-bias shortcut in LLM-based discrimination rerankers for knowledge graph completion. An ordered-candidate-trained Mistral-7B reranker on FB15K-237 loses 81% of its in-distribution Hits@1 under shuffled inference and 96% under reversal, with residual accuracy below its TransE retriever. Shuffling candidates during training eliminates the shortcut, reducing across-condition spread by a factor of about sixty at the cost of approximately six absolute Hits@1 points. The honest LLM contribution, once the shortcut is blocked, is close to zero in Hits@1 terms; the apparent gap is the shortcut. We recommend that robustness to candidate ordering be a routine evaluation criterion for any LLM-augmented retrieval system, and that the framing of such systems as reasoning models be tempered with a corresponding control.

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

REFERENCES

[1] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems*, 33, 9459–9474. https://arxiv.org/abs/2005.11401

[2] Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., & Wu, X. (2023). Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*. https://arxiv.org/abs/2306.08302

[3] Bordes, A., Usunier, N., García-Durán, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Advances in Neural Information Processing Systems*, 26, 2787–2795.

[4] Yang, B., Yih, W.-t., He, X., Gao, J., & Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. *International Conference on Learning Representations*. https://arxiv.org/abs/1412.6575

[5] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., & Bouchard, G. (2016). Complex embeddings for simple link prediction. *International Conference on Machine Learning*, 2071–2080. https://arxiv.org/abs/1606.06357

[6] Sun, Z., Deng, Z.-H., Nie, J.-Y., & Tang, J. (2019). RotatE: Knowledge graph embedding by relational rotation in complex space. *International Conference on Learning Representations*. https://arxiv.org/abs/1902.10197

[7] Toutanova, K., & Chen, D. (2015). Observed versus latent features for knowledge base and text inference. *Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality*, 57–66. https://doi.org/10.18653/v1/W15-4007

[8] Yao, L., Mao, C., & Luo, Y. (2019). KG-BERT: BERT for knowledge graph completion. https://arxiv.org/abs/1909.03193

[9] Chen, C., Wang, Y., Li, B., & Lam, K.-Y. (2022). Knowledge is flat: A Seq2Seq generative framework for various knowledge graph completion. *Proceedings of the 29th International Conference on Computational Linguistics (COLING)*, 4005–4017. https://aclanthology.org/2022.coling-1.352/

[10] Saxena, A., Kochsiek, A., & Gemulla, R. (2022). Sequence-to-sequence knowledge graph completion and question answering. *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2814–2828. https://aclanthology.org/2022.acl-long.201/

[11] Liu, Y., Yao, Y., Ton, J.-F., Zhang, X., Cheng, R. G. H., Klochkov, Y., Taufiq, M. F., & Li, H. (2024). Finetuning generative large language models with discrimination instructions for knowledge graph completion. *International Semantic Web Conference (ISWC)*. https://arxiv.org/abs/2407.16127

[12] Wei, Y., Huang, Q., Zhang, Y., & Kwok, J. T. (2023). KICGPT: Large language model with knowledge in context for knowledge graph completion. *Findings of the Association for Computational Linguistics: EMNLP 2023*, 8667–8683. https://aclanthology.org/2023.findings-emnlp.580/

[13] Zhang, Y., Chen, Z., Zhang, W., & Chen, H. (2023). Making large language models perform better in knowledge graph completion. https://arxiv.org/abs/2310.06671

[14] Zhu, Y., Wang, X., Chen, J., Qiao, S., Ou, Y., Yao, Y., Deng, S., Chen, H., & Zhang, N. (2023). LLMs for knowledge graph construction and reasoning: Recent capabilities and future opportunities. https://arxiv.org/abs/2305.13168

[15] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations*. https://arxiv.org/abs/2106.09685

[16] Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. *Advances in Neural Information Processing Systems*, 36. https://arxiv.org/abs/2305.14314

[17] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. (2023). LLaMA 2: Open foundation and fine-tuned chat models. https://arxiv.org/abs/2307.09288

[18] Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Le Scao, T., Lavril, T., Wang, T., Lacroix, T., & El Sayed, W. (2023). Mistral 7B. https://arxiv.org/abs/2310.06825

[19] Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., & Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11), 665–673. https://doi.org/10.1038/s42256-020-00257-z

[20] Gururangan, S., Swayamdipta, S., Levy, O., Schwartz, R., Bowman, S. R., & Smith, N. A. (2018). Annotation artifacts in natural language inference data. *Proceedings of NAACL*, 107–112. https://aclanthology.org/N18-2017/

[21] McCoy, R. T., Pavlick, E., & Linzen, T. (2019). Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *Proceedings of ACL*, 3428–3448. https://aclanthology.org/P19-1334/

[22] Niven, T., & Kao, H.-Y. (2019). Probing neural network comprehension of natural language arguments. *Proceedings of ACL*, 4658–4664. https://aclanthology.org/P19-1459/

[23] Poliak, A., Naradowsky, J., Haldar, A., Rudinger, R., & Van Durme, B. (2018). Hypothesis only baselines in natural language inference. *Joint Conference on Lexical and Computational Semantics (*SEM)*, 180–191. https://aclanthology.org/S18-2023/

[24] Lu, Y., Bartolo, M., Moore, A., Riedel, S., & Stenetorp, P. (2022). Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *Proceedings of ACL*, 8086–8098. https://aclanthology.org/2022.acl-long.556/

[25] Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., & Zettlemoyer, L. (2022). Rethinking the role of demonstrations: What makes in-context learning work? *Proceedings of EMNLP*, 11048–11064. https://aclanthology.org/2022.emnlp-main.759/

[26] Zhao, Z., Wallace, E., Feng, S., Klein, D., & Singh, S. (2021). Calibrate before use: Improving few-shot performance of language models. *International Conference on Machine Learning*, 12697–12706. https://arxiv.org/abs/2102.09690

[27] Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., & Liang, P. (2023). Lost in the middle: How language models use long contexts. https://arxiv.org/abs/2307.03172

[28] Koo, R., Lee, M., Raheja, V., Park, J. I., Kim, Z. M., & Kang, D. (2024). Benchmarking cognitive biases in large language models as evaluators. *Findings of the Association for Computational Linguistics: ACL 2024*, 517–545. https://aclanthology.org/2024.findings-acl.29/

[29] Pezeshkpour, P., & Hruschka, E. (2024). Large language models sensitivity to the order of options in multiple-choice questions. *Findings of the Association for Computational Linguistics: NAACL 2024*, 2006–2017. https://aclanthology.org/2024.findings-naacl.130/

[30] Zheng, C., Zhou, H., Meng, F., Zhou, J., & Huang, M. (2024). Large language models are not robust multiple choice selectors. *International Conference on Learning Representations (Spotlight)*. https://arxiv.org/abs/2309.03882

[31] Balažević, I., Allen, C., & Hospedales, T. (2019). TuckER: Tensor factorization for knowledge graph completion. *Proceedings of EMNLP*, 5184–5193. https://aclanthology.org/D19-1522/

[32] Dettmers, T., Minervini, P., Stenetorp, P., & Riedel, S. (2018). Convolutional 2D knowledge graph embeddings. *Proceedings of AAAI*, 1811–1818. https://arxiv.org/abs/1707.01476

[33] Nickel, M., Tresp, V., & Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. *International Conference on Machine Learning*, 809–816.

[34] Vashishth, S., Sanyal, S., Nitin, V., & Talukdar, P. (2020). Composition-based multi-relational graph convolutional networks. *International Conference on Learning Representations*. https://arxiv.org/abs/1911.03082

[35] Galkin, M., Denis, E., Wu, J., & Hamilton, W. L. (2022). NodePiece: Compositional and parameter-efficient representations of large knowledge graphs. *International Conference on Learning Representations*. https://arxiv.org/abs/2106.12144

[36] Wang, L., Zhao, W., Wei, Z., & Liu, J. (2022). SimKGC: Simple contrastive knowledge graph completion with pre-trained language models. *Proceedings of ACL*, 4281–4294. https://aclanthology.org/2022.acl-long.295/

[37] Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., & Smola, A. J. (2017). Deep Sets. *Advances in Neural Information Processing Systems*, 30, 3391–3401. https://arxiv.org/abs/1703.06114

Journal of Information Systems Engineering and Management
2026, 11(3s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

[38] Lee, J., Lee, Y., Kim, J., Kosiorek, A. R., Choi, S., & Teh, Y. W. (2019). Set Transformer: A framework for attention-based permutation-invariant neural networks. *International Conference on Machine Learning*, 3744–3753. https://arxiv.org/abs/1810.00825

[39] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., & Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. *Proceedings of EMNLP*, 6769–6781. https://aclanthology.org/2020.emnlp-main.550/