Autoregressive Ranking

Bridging the Gap Between Dual and Cross Encoders

Ben Rozonoyer talk at GDM, February 10, 2026

Autoregressive Ranking

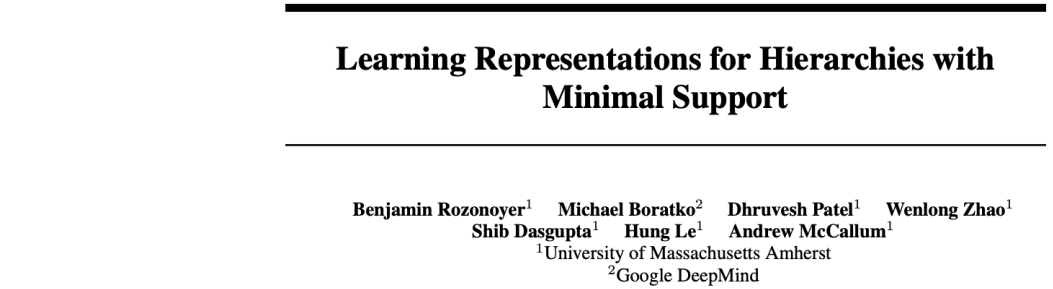
Bridging the Gap Between Dual and Cross Encoders

Ben Rozonoyer talk at GDM, February 10, 2026

Collaboration with **Chong You**, **Michael Boratko**, **Himanshu Jain**, **Nilesh Gupta**, **Srinadh Bhojanapalli**, **Andrew McCallum** and **Felix Yu**

About me

- PhD candidate (4th) at Information Extraction and Synthesis Lab
- Advised by Andrew McCallum
- Current research focus — discrete diffusion models
- Internships:
  - Transformer-based authorship recognition (Summer 2023)
  - LLM agents for debugging (Summer 2024)
  - Accelerating MoE inference (Summer 2025)
  - Generative Ranking (Spring 2025)



Outline for This Talk

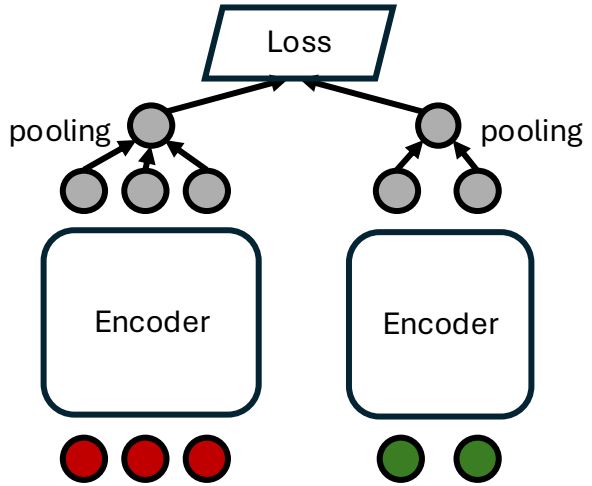
- Introduction
- The Capacity of Autoregressive LLMs for Ranking
  - Ranking via Dual Encoders
  - Ranking via Pointwise Autoregressive Rankers
- A Rank-Aware Loss Function
  - Item-Level Reweighting
  - Prefix Tree for Rank-Aware Token-Level Supervision
- Experiments
  - WordNet
  - ESCI

Introduction

Introduction

Prevailing IR paradigm is multistage pipeline:

Introduction



Prevailing IR paradigm is multistage pipeline:

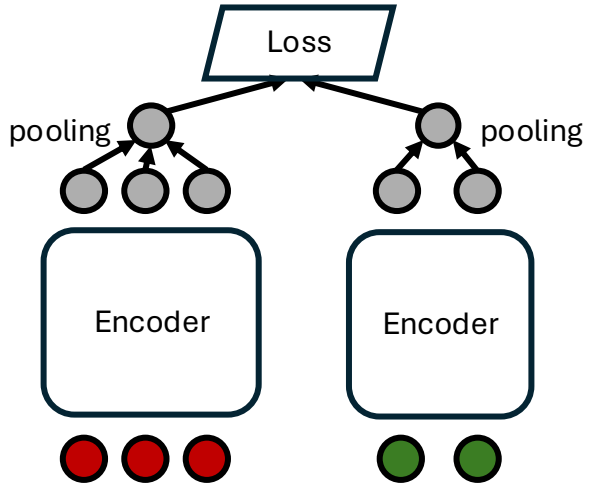
1. **Retrieval** *(vector similarity)*
   Dual Encoders (DEs)
   Approximate Nearest Neighbors (ANN)

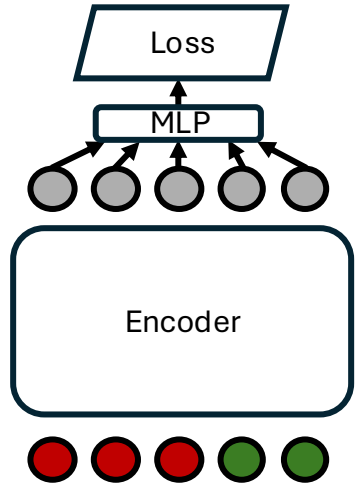
Introduction

Prevailing IR paradigm is multistage pipeline:

1. **Retrieval** *(vector similarity)*
   Dual Encoders (DEs)
   Approximate Nearest Neighbors (ANN)

2. **Reranking** *(prohibitive for first-stage retrieval)*
   Cross Encoders (CEs)

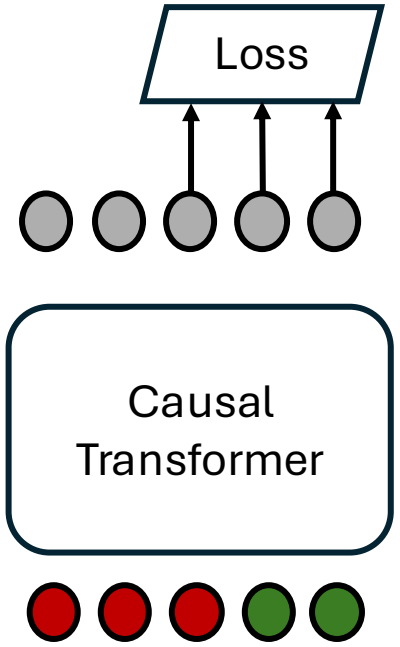




Introduction

Introduction

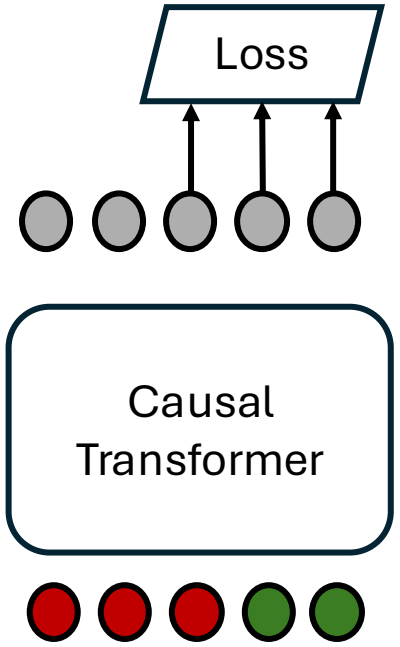
LLMs inspire unified “**Autoregressive Ranking**”:



Introduction

LLMs inspire unified “**Autoregressive Ranking**”:

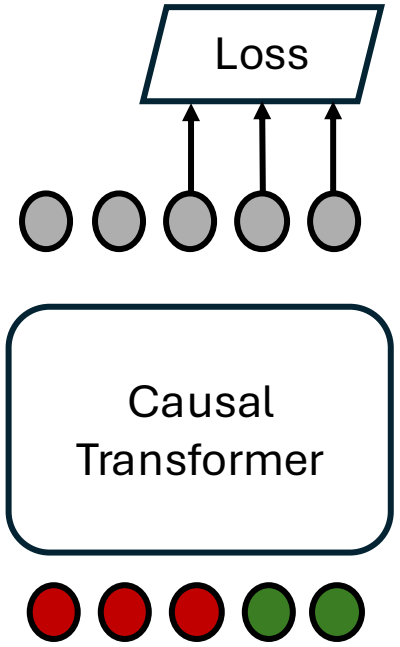
- Generate docIDs token-by-token → ranking via beam-search
- 2-stage pipeline → single model



Introduction

LLMs inspire unified "**Autoregressive Ranking**":

- Generate docIDs token-by-token → ranking via beam-search
- 2-stage pipeline → single model
- *More efficient than CEs*
- *More expressive than DEs (empirical evidence)*



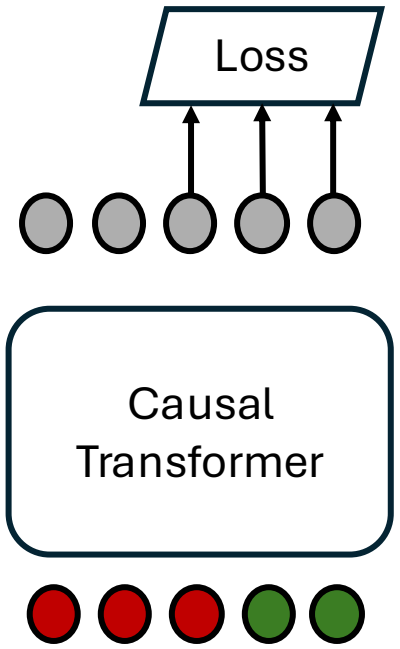
Introduction

LLMs inspire unified "**Autoregressive Ranking**":

- Generate docIDs token-by-token → ranking via beam-search
- 2-stage pipeline → single model
- *More efficient than CEs*
- *More expressive than DEs (empirical evidence)*

Many LLM rankers are currently trained with NTP

- *rank-agnostic!*



Contributions

Contributions

1. Theoretical foundation for superior expressive capacity of ARRs over DEs

Contributions

1. Theoretical foundation for superior expressive capacity of ARRs over DEs
   - DEs require embedding dimension to grow linearly with # documents

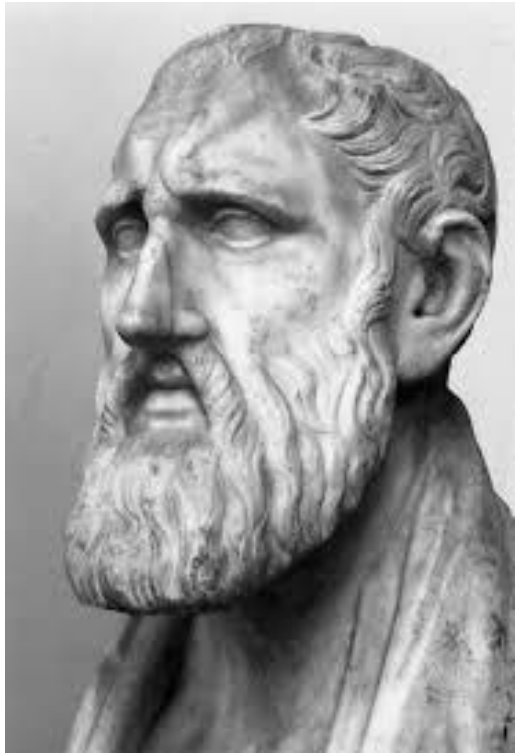
Contributions

1. Theoretical foundation for superior expressive capacity of ARRs over DEs
   - DEs require embedding dimension to grow linearly with # documents
   - ARRs which generate multitoken docIDs can achieve arbitrary rankings with fixed dimension

Contributions

1. Theoretical foundation for superior expressive capacity of ARRs over DEs
   - DEs require embedding dimension to grow linearly with # documents
   - ARRs which generate multitoken docIDs can achieve arbitrary rankings with fixed dimension

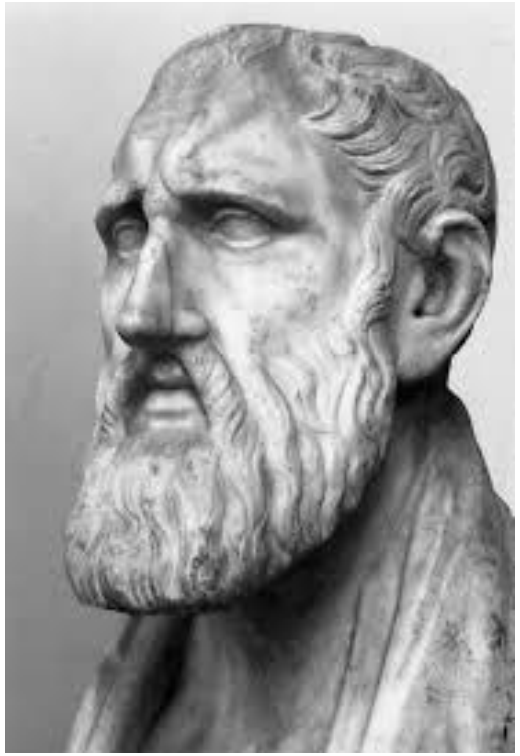
2. STOICAL: **S**imple **To**ken-**I**tem **Ca**librated **L**oss



Contributions

1. Theoretical foundation for superior expressive capacity of ARRs over DEs
   - DEs require embedding dimension to grow linearly with # documents
   - ARRs which generate multitoken docIDs can achieve arbitrary rankings with fixed dimension

2. STOICAL: **S**imple **To**ken-**I**tem **Ca**librated **L**oss
   - Rank-Aware Generalization to Next-Token Prediction



The Capacity for Ranking

Superior Expressive Capacity of ARRs over DEs

Ranking Task

Ranking Task

- $\mathcal{Q}$: queries
- $\mathcal{D}$: documents

Ranking Task

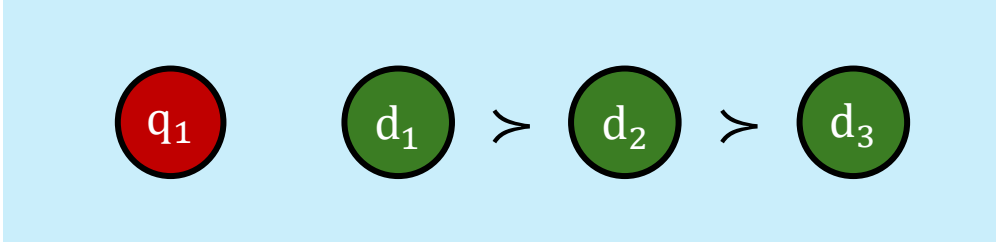
- $\mathcal{Q}$: queries
- $\mathcal{D}$: documents
- **Objective**
  - For each query $q \in \mathcal{Q}$:

Ranking Task

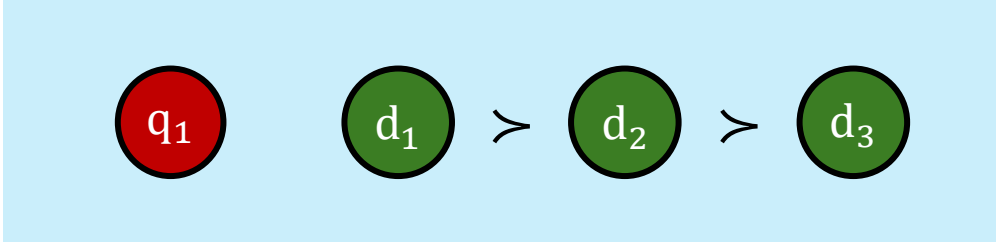
- $\mathcal{Q}$: queries
- $\mathcal{D}$: documents
- **Objective**
  - For each query $q \in \mathcal{Q}$:
  - **Want to generate $\mathcal{L}(\boldsymbol{q}) = [\boldsymbol{d_1}(\boldsymbol{q}), \ldots, \boldsymbol{d_k}(\boldsymbol{q})]$ with $\boldsymbol{d_i}(\boldsymbol{q}) \in \mathcal{D}$**
    - $\forall i < j$, $d_i(q)$ is considered more relevant to q than $d_j(q)$

Ranking Task

- $\mathcal{Q}$: queries
- $\mathcal{D}$: documents
- **Objective**
  - For each query $q \in \mathcal{Q}$:
  - **Want to generate $\mathcal{L}(q) = [d_1(q), \ldots, d_k(q)]$ with $d_i(q) \in \mathcal{D}$**
    - $\forall i < j$, $d_i(q)$ is considered more relevant to q than $d_j(q)$

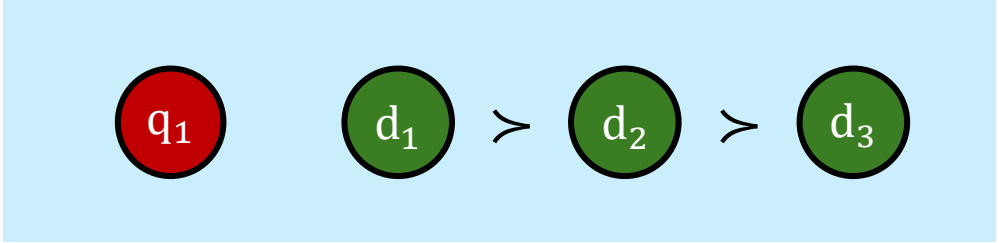


Complete Ranking Task



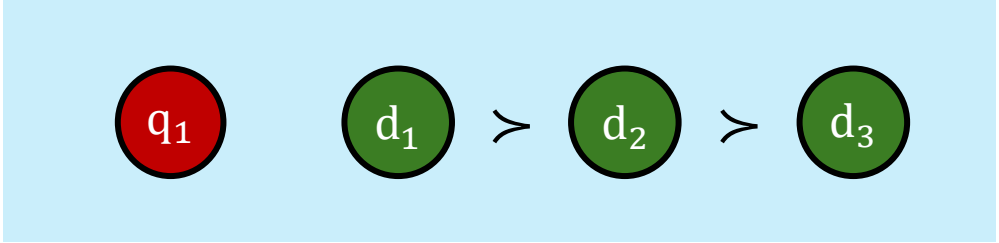
Complete Ranking Task

- Given $\mathcal{Q}$ and $\mathcal{D}$, we call a ranking task ***complete*** if:



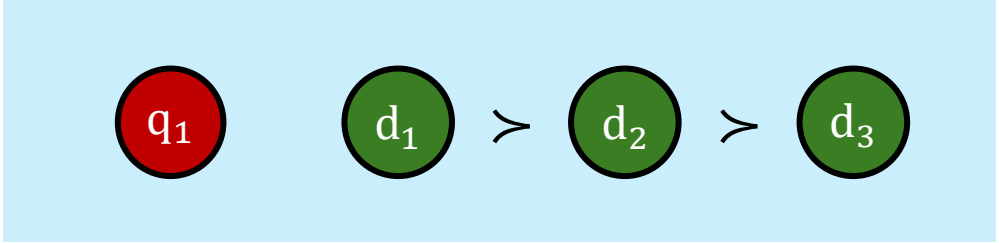
Complete Ranking Task

- Given $\mathcal{Q}$ and $\mathcal{D}$, we call a ranking task ***complete*** if:
  - $\forall q \in \mathcal{Q}$, **all documents are relevant**, i.e., $\mathcal{D}(q) = \mathcal{D}$



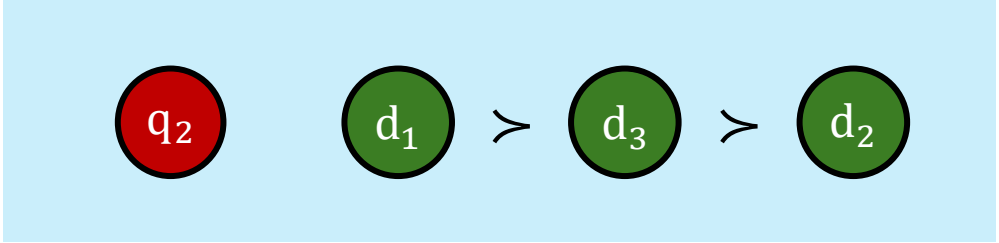
Complete Ranking Task

- Given $\mathcal{Q}$ and $\mathcal{D}$, we call a ranking task ***complete*** if:
  - $\forall q \in \mathcal{Q}$, **all documents are relevant**, i.e., $\mathcal{D}(q) = \mathcal{D}$
  - **Every permutation over $\mathcal{D}$ is involved**: For all orderings $\pi(\mathcal{D})$ of the documents $\mathcal{D}$, there exists a $\exists q \in \mathcal{Q}$ such that its ranked list of relevant documents $\mathcal{L}(q) \equiv \pi(\mathcal{D})$.



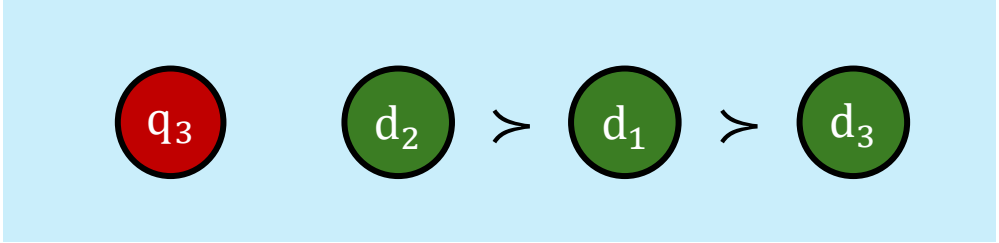
Complete Ranking Task

- Given $\mathcal{Q}$ and $\mathcal{D}$, we call a ranking task ***complete*** if:
  - $\forall q \in \mathcal{Q}$, **all documents are relevant**, i.e., $\mathcal{D}(q) = \mathcal{D}$
  - **Every permutation over $\mathcal{D}$ is involved**: For all orderings $\pi(\mathcal{D})$ of the documents $\mathcal{D}$, there exists a $\exists q \in \mathcal{Q}$ such that its ranked list of relevant documents $\mathcal{L}(q) \equiv \pi(\mathcal{D})$.



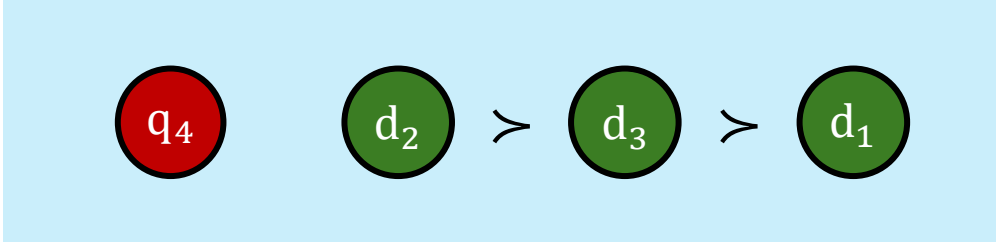
Complete Ranking Task

- Given $\mathcal{Q}$ and $\mathcal{D}$, we call a ranking task ***complete*** if:
  - $\forall q \in \mathcal{Q}$, **all documents are relevant**, i.e., $\mathcal{D}(q) = \mathcal{D}$
  - **Every permutation over $\mathcal{D}$ is involved**: For all orderings $\pi(\mathcal{D})$ of the documents $\mathcal{D}$, there exists a $\exists q \in \mathcal{Q}$ such that its ranked list of relevant documents $\mathcal{L}(q) \equiv \pi(\mathcal{D})$.



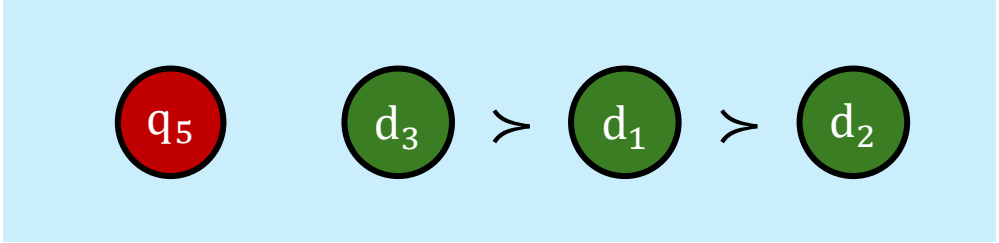
Complete Ranking Task

- Given $\mathcal{Q}$ and $\mathcal{D}$, we call a ranking task ***complete*** if:
  - $\forall q \in \mathcal{Q}$, **all documents are relevant**, i.e., $\mathcal{D}(q) = \mathcal{D}$
  - **Every permutation over $\mathcal{D}$ is involved**: For all orderings $\pi(\mathcal{D})$ of the documents $\mathcal{D}$, there exists a $\exists q \in \mathcal{Q}$ such that its ranked list of relevant documents $\mathcal{L}(q) \equiv \pi(\mathcal{D})$.



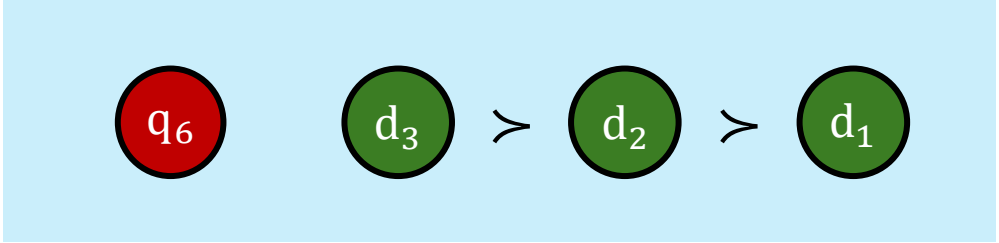
Complete Ranking Task

- Given $\mathcal{Q}$ and $\mathcal{D}$, we call a ranking task ***complete*** if:
  - $\forall q \in \mathcal{Q}$, **all documents are relevant**, i.e., $\mathcal{D}(q) = \mathcal{D}$
  - **Every permutation over $\mathcal{D}$ is involved**: For all orderings $\pi(\mathcal{D})$ of the documents $\mathcal{D}$, there exists a $\exists q \in \mathcal{Q}$ such that its ranked list of relevant documents $\mathcal{L}(q) \equiv \pi(\mathcal{D})$.



Complete Ranking Task

- Given $\mathcal{Q}$ and $\mathcal{D}$, we call a ranking task ***complete*** if:
  - $\forall q \in \mathcal{Q}$, **all documents are relevant**, i.e., $\mathcal{D}(q) = \mathcal{D}$
  - **Every permutation over $\mathcal{D}$ is involved**: For all orderings $\pi(\mathcal{D})$ of the documents $\mathcal{D}$, there exists a $\exists q \in \mathcal{Q}$ such that its ranked list of relevant documents $\mathcal{L}(q) \equiv \pi(\mathcal{D})$.



Insufficiency of DEs for Complete Ranking

Let $f_{\mathrm{DE}}(\mathrm{q}, \mathrm{d}; \theta)$ be a dual encoder architecture with embedding dimension n, and let $k = |\mathcal{D}|$. Then there does not exist a θ such that $f_{\mathrm{DE}}(\mathrm{q}, \mathrm{d}; \theta)$ solves a complete ranking task when

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

Insufficiency of DEs for Complete Ranking

Let $f_{\mathrm{DE}}(\mathrm{q}, \mathrm{d}; \theta)$ be a dual encoder architecture with embedding dimension n, and let $k = |\mathcal{D}|$. Then there does not exist a θ such that $f_{\mathrm{DE}}(\mathrm{q}, \mathrm{d}; \theta)$ solves a complete ranking task when

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

Proof [*Counting Distance Permutations, Skala 2009*]:
Let $N_{n,p}(k)$ denote the maximum number of distinct distance permutations generated by k sites in $\mathbb{R}^n$ with the L_p metric. From Skala 2009, Corollary 8, $N_{n,2}(k) \leq k^{2n}$. When above inequality holds, $k^{2n} < k!$ ■

Insufficiency of DEs for Complete Ranking

Let $f_{\mathrm{DE}}(\mathrm{q}, \mathrm{d}; \theta)$ be a dual encoder architecture with embedding dimension n, and let $k = |\mathcal{D}|$. Then there does not exist a θ such that $f_{\mathrm{DE}}(\mathrm{q}, \mathrm{d}; \theta)$ solves a complete ranking task when

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

By Stirling's formula $\ln(k!) = k \ln k - k + O(\ln(k))$, this becomes $n < \frac{k}{2} - \frac{k}{2 \ln k} + O(1)$, i.e., if $\boldsymbol{n < O(k) = O(|\mathcal{D}|)}$.

Insufficiency of DEs for Complete Ranking

Let $f_{\mathrm{DE}}(\mathrm{q}, \mathrm{d}; \theta)$ be a dual encoder architecture with embedding dimension n, and let $k = |\mathcal{D}|$. Then there does not exist a θ such that $f_{\mathrm{DE}}(\mathrm{q}, \mathrm{d}; \theta)$ solves a complete ranking task when

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

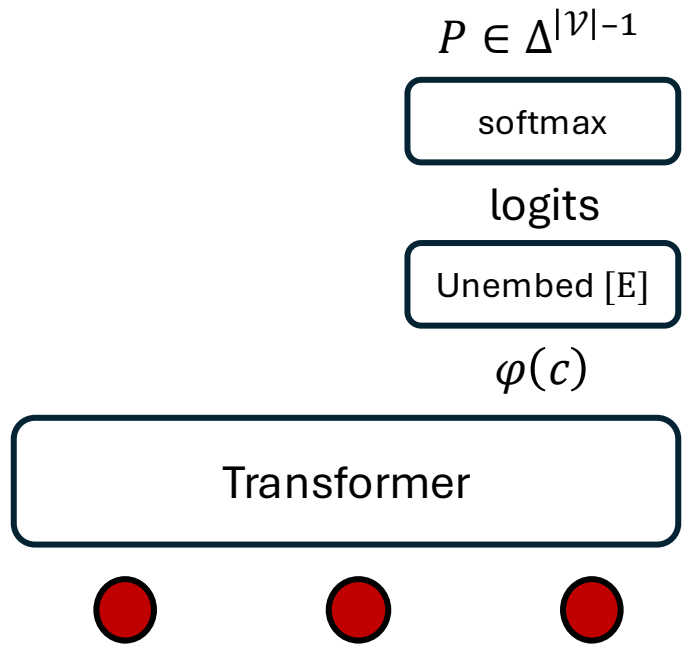
By Stirling's formula $\ln(k!) = k \ln k - k + O(\ln(k))$, this becomes $n < \frac{k}{2} - \frac{k}{2 \ln k} + O(1)$, i.e., if $\boldsymbol{n < O(k) = O(|\mathcal{D}|)}$.

DE embedding dimension

Must grow linearly with corpus size

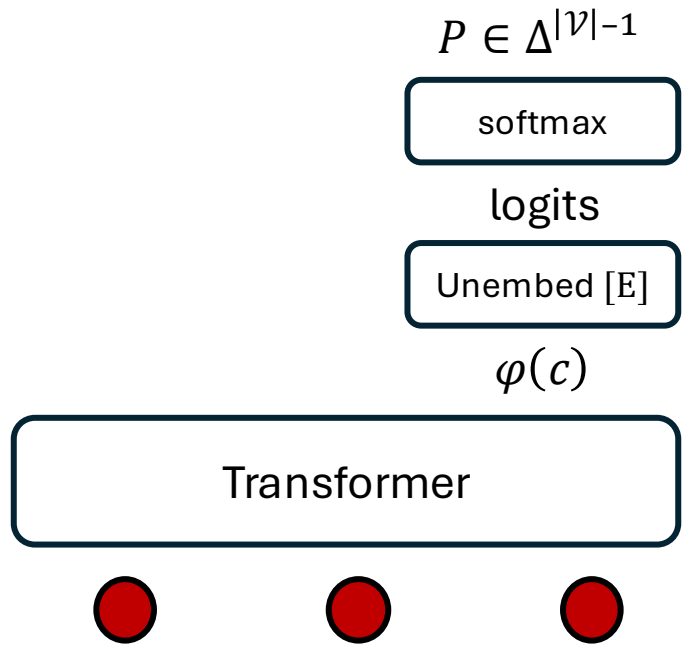
Ranking via (Pointwise) Autoregressive Rankers

Ranking via (Pointwise) Autoregressive Rankers



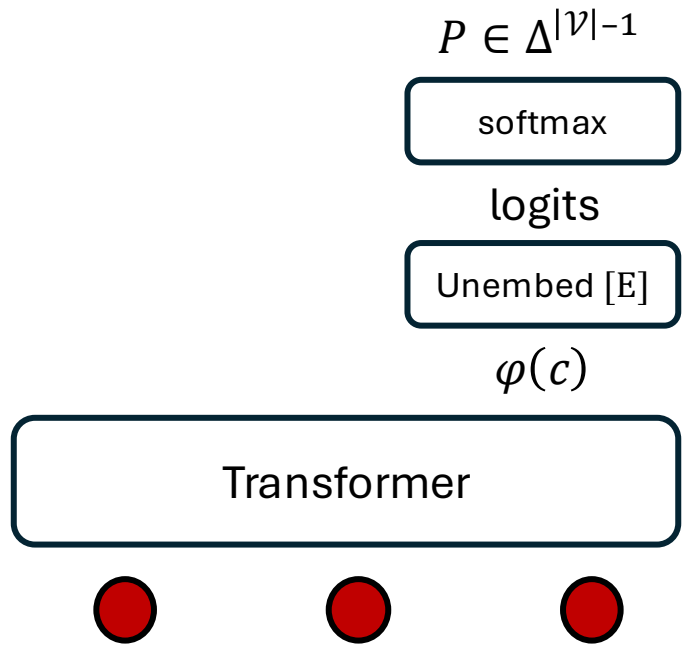
Ranking via (Pointwise) Autoregressive Rankers

- ARR uses token (un)embeddings $\mathrm{E} \in \mathbb{R}^{|\mathcal{V}| \times n}$



Ranking via (Pointwise) Autoregressive Rankers

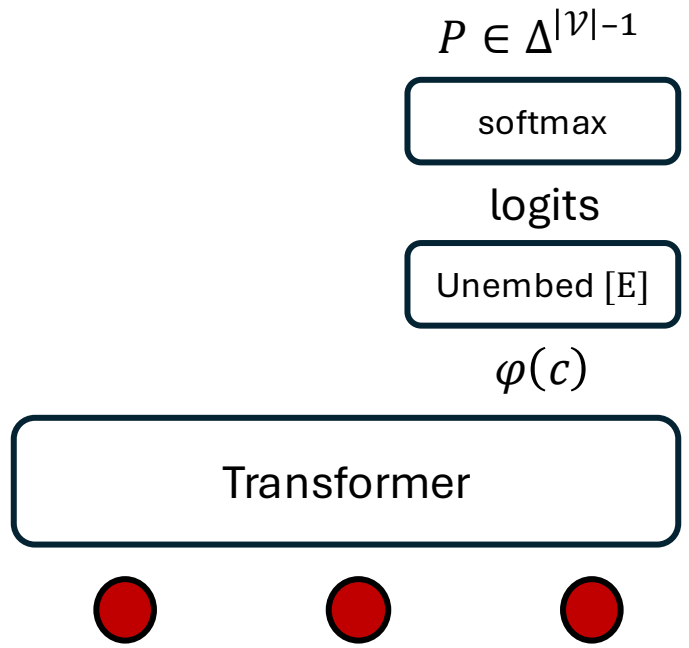
- ARR uses token (un)embeddings $\mathrm{E} \in \mathbb{R}^{|\mathcal{V}| \times n}$
- Next-token probs over vocab $\mathcal{V}$ as $P(\cdot \mid c) = \mathrm{softmax}(\mathrm{E}\,\varphi(c))$



Ranking via (Pointwise) Autoregressive Rankers

- ARR uses token (un)embeddings $\mathrm{E} \in \mathbb{R}^{|\mathcal{V}| \times n}$
- Next-token probs over vocab $\mathcal{V}$ as $P(\cdot \mid c) = \mathrm{softmax}(\mathrm{E}\,\varphi(c))$

- For ranking, documents represented by **docIDs**

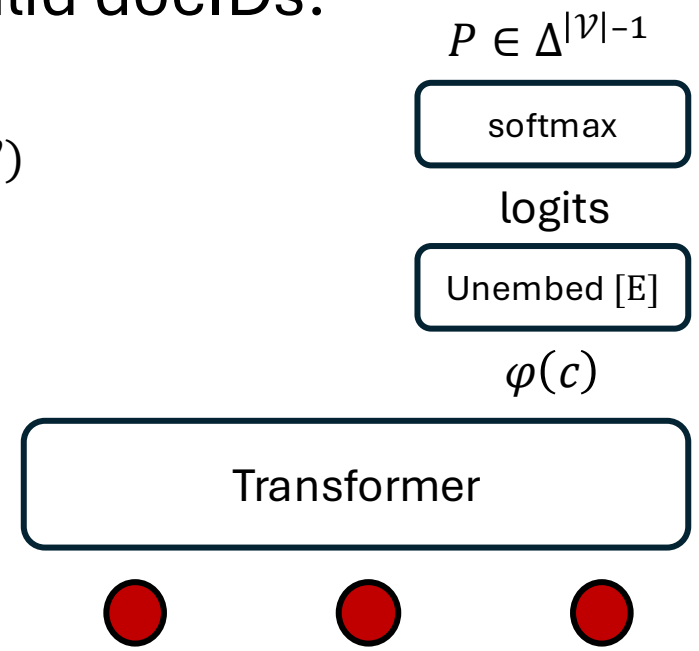


Ranking via (Pointwise) Autoregressive Rankers

- ARR uses token (un)embeddings $\mathrm{E} \in \mathbb{R}^{|\mathcal{V}| \times n}$
- Next-token probs over vocab $\mathcal{V}$ as $P(\cdot \mid c) = \mathrm{softmax}(\mathrm{E}\,\varphi(c))$

- For ranking, documents represented by **docIDs**
- In practice, use **constrained decoding** to ensure valid docIDs:

$$P(v \mid c) = \begin{cases} \mathrm{softmax}\left(\mathrm{E}_{\mathrm{docIDs}}\varphi(c)\right) & \text{if } v \in \mathcal{V}_{\mathrm{docIDs}} \ \ (\mathcal{V}_{\mathrm{docIDs}} \subset \mathcal{V}) \\ 0 & \text{otherwise} \end{cases}$$



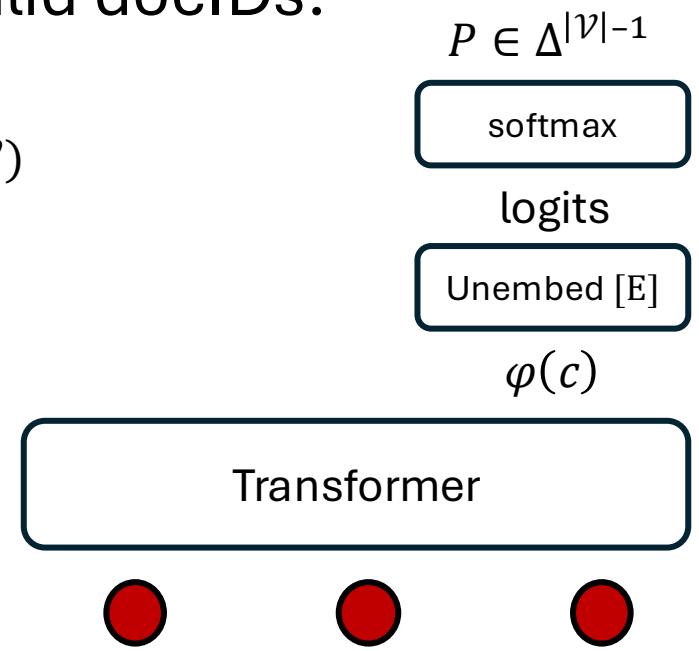
Ranking via (Pointwise) Autoregressive Rankers

- ARR uses token (un)embeddings $\mathrm{E} \in \mathbb{R}^{|\mathcal{V}| \times n}$
- Next-token probs over vocab $\mathcal{V}$ as $P(\cdot \mid c) = \mathrm{softmax}(\mathrm{E}\,\varphi(c))$

- For ranking, documents represented by **docIDs**
- In practice, use **constrained decoding** to ensure valid docIDs:

$$P(v \mid c) = \begin{cases} \mathrm{softmax}\left(\mathrm{E}_{\mathrm{docIDs}}\varphi(c)\right) & \text{if } v \in \mathcal{V}_{\mathrm{docIDs}} \ \ (\mathcal{V}_{\mathrm{docIDs}} \subset \mathcal{V}) \\ 0 & \text{otherwise} \end{cases}$$

- Define $\mathrm{E}' = [\mathrm{E}_{\mathrm{docIDs}}\ \mathbf{1}_{|\mathcal{V}_{\mathrm{docIDs}}|}]$



ARRs: Arbitrary Distributions over Tokens

An infinite-capacity ARR can produce **any (strictly positive) probability distribution over tokens in $\mathcal{V}_{\text{docIDs}}$** via constrained decoding

$\Leftrightarrow$

$$\text{rank}(\text{E}') = |\mathcal{V}_{\text{docIDs}}|$$

Proof by contradiction. If $\text{rank}(\text{E}') < |\mathcal{V}_{\text{docIDs}}|$*, then we can show that there exists a distribution which can't be hit.*

ARRs: Arbitrary **Distributions over Sequences**

The chain rule of probability implies…

An infinite-capacity ARR can produce **any (strictly positive) probability distribution over sequences in** $\mathcal{V}_{\text{docIDs}}$ via constrained decoding

$\Leftrightarrow$

$$\text{rank}(\text{E}') = |\mathcal{V}_{\text{docIDs}}|$$

ARRs: Arbitrary **Permutations over Tokens**

*Clearly, arbitrary distributions over sequences implies arbitrary rankings over sequences. But is the rank requirement also **necessary** for arbitrary rankings over sequences?*

An infinite-capacity ARR can produce **any (strictly positive) permutations over tokens in $\mathcal{V}_{\text{docIDs}}$** via constrained decoding

$\Leftrightarrow$

$$\text{rank}(\text{E}') = |\mathcal{V}_{\text{docIDs}}|$$

Rank requirement is both **necessary and sufficient** to enable a fixed-dimension ARR to generate arbitrary probability distributions or arbitrary permutations for an unrestricted number of docIDs.

Practical Implications

- Is theoretical requirement on E' satisfied by LLMs used for reranking?
- Why are LLMs successful even if requirement can't be satisfied?
- WordNet
  - docIDs are English nouns $\Rightarrow |\mathcal{V}_{\mathrm{docIDs}}| \approx |\mathcal{V}|$.
  - $n \ll |\mathcal{V}| \Rightarrow \mathrm{rank}(\mathrm{E}') \ll |\mathcal{V}_{\mathrm{docIDs}}|$
  - `Mistral-7B-v0.3-it` has $|\mathcal{V}| = 32{,}768$ and $d = 4{,}096$
  - Most possible sequences never encountered; model needs to learn tiny subset of all possible permutations.
- ESCI Shopping Queries
  - we choose numerical docIDs $\Rightarrow n \approx |\mathcal{V}_{\mathrm{docIDs}}|$
  - rank requirement on E' can be matched more accurately

A Rank-Aware Loss Function

Incorporating Rank-Awareness into Next Token Prediction

STOICAL: **S**imple **To**ken-**I**tem **Ca**librated **L**oss

$$\mathcal{L}(q, d_r, r; \theta)$$

STOICAL: **S**imple **To**ken-**I**tem **Ca**librated **L**oss

$$\mathcal{L}(q, d_r, r; \theta) = \lambda(r) \sum_{t=1}^{|\mathcal{T}(d_r)|} \left(\mathrm{CE}\left(y(r,t), p_\theta\left(\mathcal{T}(d_r)[t] \mid \mathcal{T}(\bar{q}), \mathcal{T}(d_r)_{<t} \right) \right) \right)$$

STOICAL: **S**imple **To**ken-**I**tem **Ca**librated **L**oss

$$\mathcal{L}(q, d_r, r; \theta) = \boldsymbol{\lambda(r)} \sum_{t=1}^{|\mathcal{T}(d_r)|} \left(\mathrm{CE}\left(\boldsymbol{y(r,t)}, p_\theta\left(\mathcal{T}(d_r)[t] \mid \mathcal{T}(\bar{q}), \mathcal{T}(d_r)_{<t} \right) \right) \right)$$

STOICAL: **S**imple **To**ken-**I**tem **Ca**librated **L**oss

$$\mathcal{L}(q, d_r, r; \theta) = \boldsymbol{\lambda(r)} \sum_{t=1}^{|\mathcal{T}(d_r)|} \left(\mathrm{CE}\left(\boldsymbol{y(r,t)}, p_\theta\left(\mathcal{T}(d_r)[t] \mid \mathcal{T}(\bar{q}), \mathcal{T}(d_r)_{<t}\right)\right)\right)$$

model-generated per-timestep probabilities

STOICAL: **S**imple **To**ken-**I**tem **Ca**librated **L**oss

$$\mathcal{L}(q, d_r, r; \theta) = \boldsymbol{\lambda(r)} \sum_{t=1}^{|\mathcal{T}(d_r)|} \left(\mathrm{CE}\left(\boldsymbol{y(r,t)}, p_\theta\left(\mathcal{T}(d_r)[t] \mid \mathcal{T}(\bar{q}), \mathcal{T}(d_r)_{<t} \right) \right) \right)$$

(rank-aware) token-level supervision

model-generated per-timestep probabilities

STOICAL: **S**imple **To**ken-**I**tem **Ca**librated **L**oss

$$\mathcal{L}(q, d_r, r; \theta) = \boldsymbol{\lambda(r)} \sum_{t=1}^{|\mathcal{T}(d_r)|} \left(\mathrm{CE}\left(\boldsymbol{y(r,t)}, p_\theta\left(\mathcal{T}(d_r)[t] \mid \mathcal{T}(\bar{q}), \mathcal{T}(d_r)_{<t} \right) \right) \right)$$

(rank-aware) token-level supervision

(simplest is one-hots)

model-generated per-timestep probabilities

STOICAL: **S**imple **To**ken-**I**tem **Ca**librated **L**oss

$$\mathcal{L}(q, d_r, r; \theta) = \boldsymbol{\lambda(r)} \sum_{t=1}^{|\mathcal{T}(d_r)|} \left(\mathrm{CE}\left(\boldsymbol{y(r,t)}, p_\theta\left(\mathcal{T}(d_r)[t] \mid \mathcal{T}(\bar{q}), \mathcal{T}(d_r)_{<t} \right) \right) \right)$$

(rank-aware) item-level reweighting

(rank-aware) token-level supervision

(simplest is one-hots)

model-generated per-timestep probabilities

STOICAL: **S**imple **To**ken-**I**tem **Ca**librated **L**oss

$$\mathcal{L}(q, d_r, r; \theta) = \boldsymbol{\lambda(r)} \sum_{t=1}^{|\mathcal{T}(d_r)|} \left(\mathrm{CE}\left(\boldsymbol{y(r,t)}, p_\theta\left(\mathcal{T}(d_r)[t] \mid \mathcal{T}(\bar{q}), \mathcal{T}(d_r)_{<t} \right) \right) \right)$$

(rank-aware) item-level reweighting
(simplest is $\mathbb{I}_{r=1}$)

(rank-aware) token-level supervision
(simplest is one-hots)

model-generated per-timestep probabilities

Item-Level Reweighting

Item-Level Reweighting

- $\lambda(r) = \frac{1}{r^{\alpha}}$ *fractional (with temperature)*

Item-Level Reweighting

- $\lambda(r) = \frac{1}{r^{\alpha}}$ *fractional (with temperature)*
- $\lambda(r) = \frac{n_q - r + 1}{n_q}$ *stepwise (from 1 to 0)*

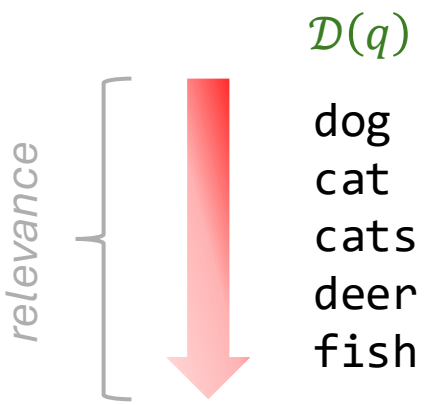
Prefix Tree (Trie) Construction

Prefix Tree (Trie) Construction

q: “a common pet”

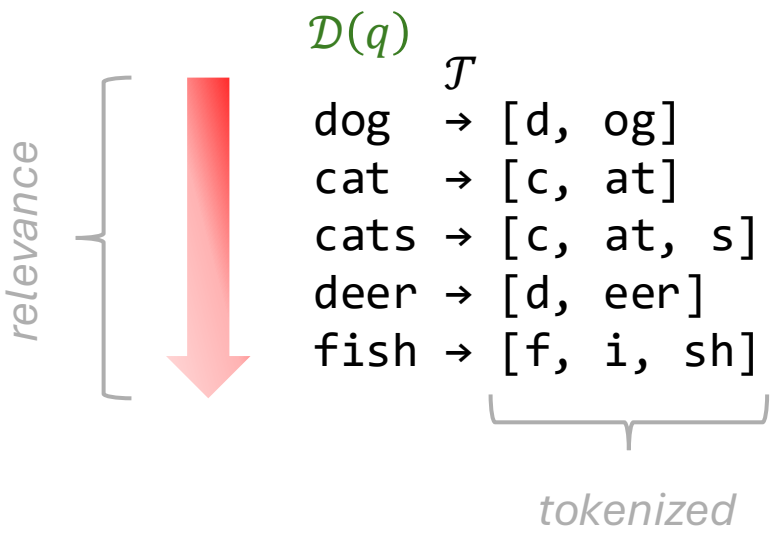
Prefix Tree (Trie) Construction

q: “a common pet”



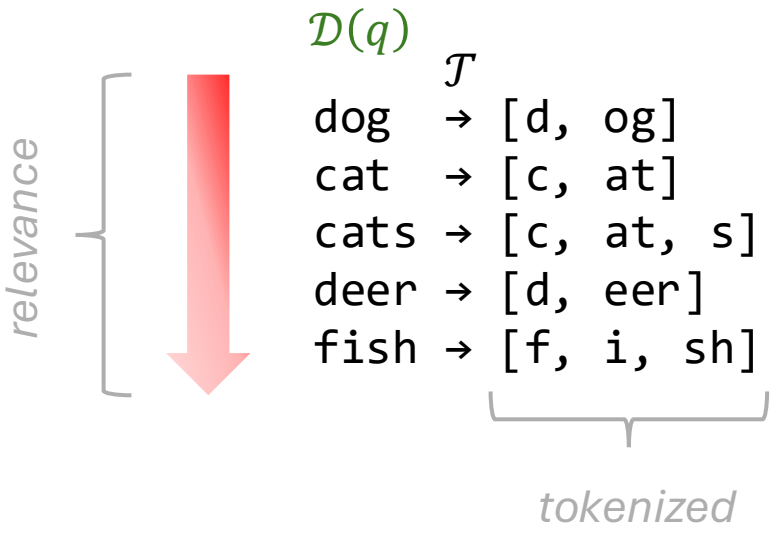
Prefix Tree (Trie) Construction

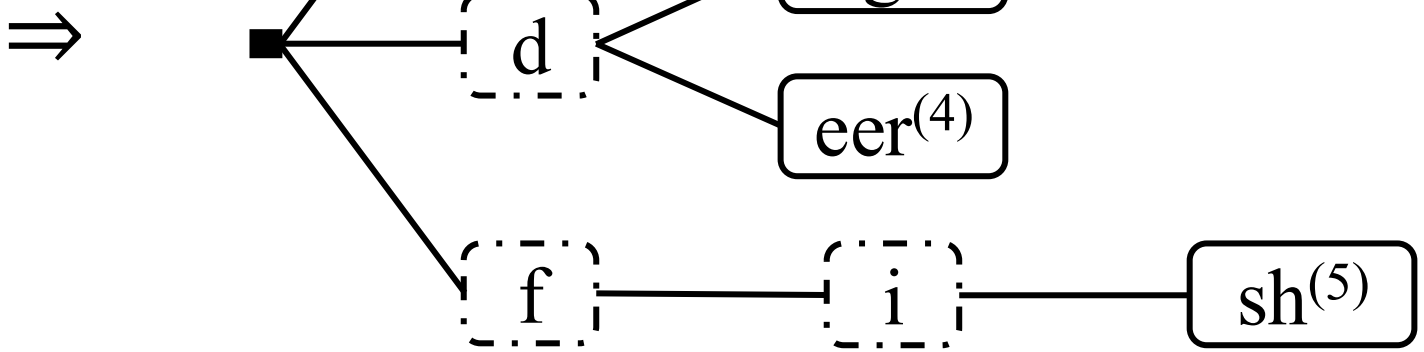
q: “a common pet”



Prefix Tree (Trie) Construction

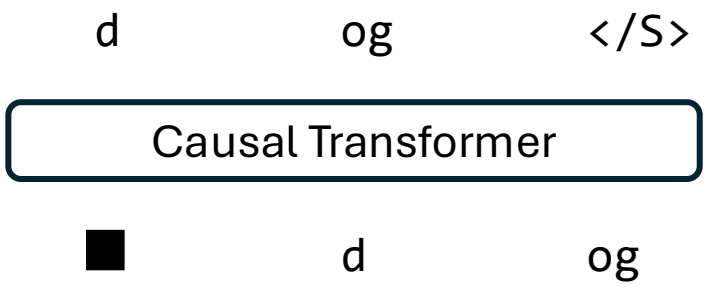
q: “a common pet”



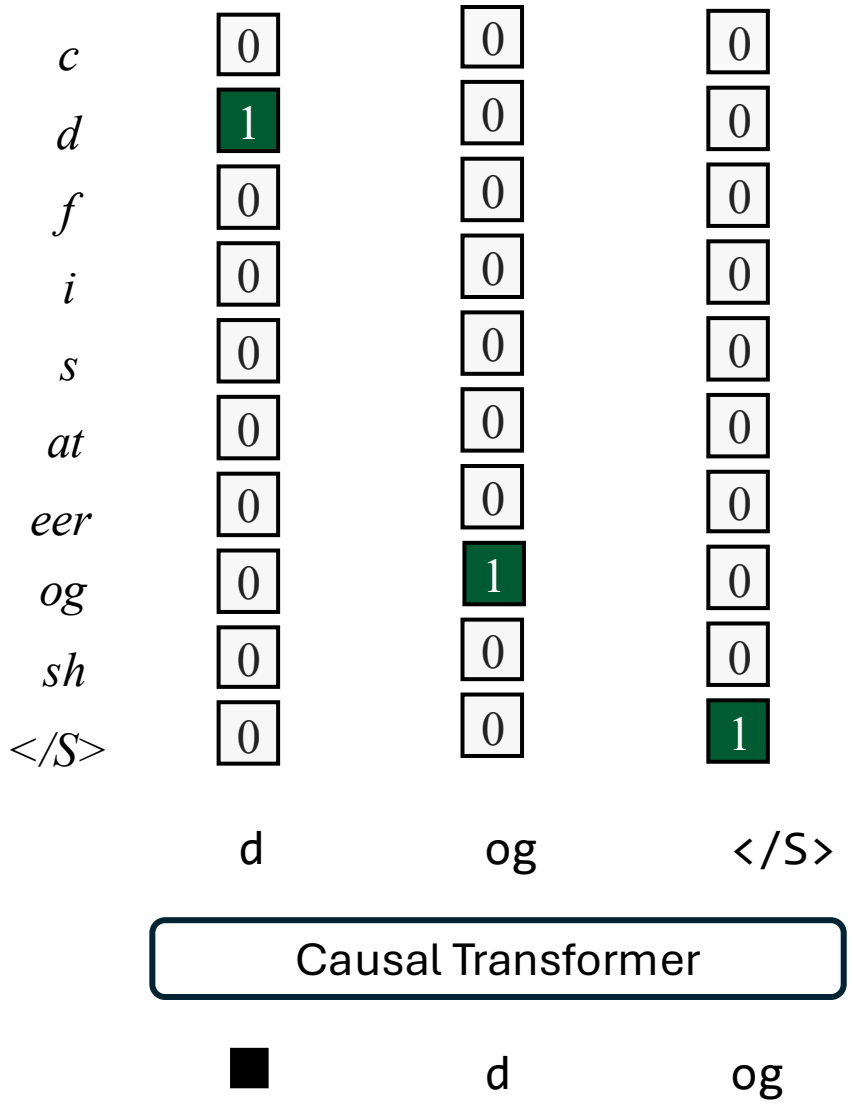


Teacher Forcing with One Hot Supervision

Teacher Forcing with One Hot Supervision

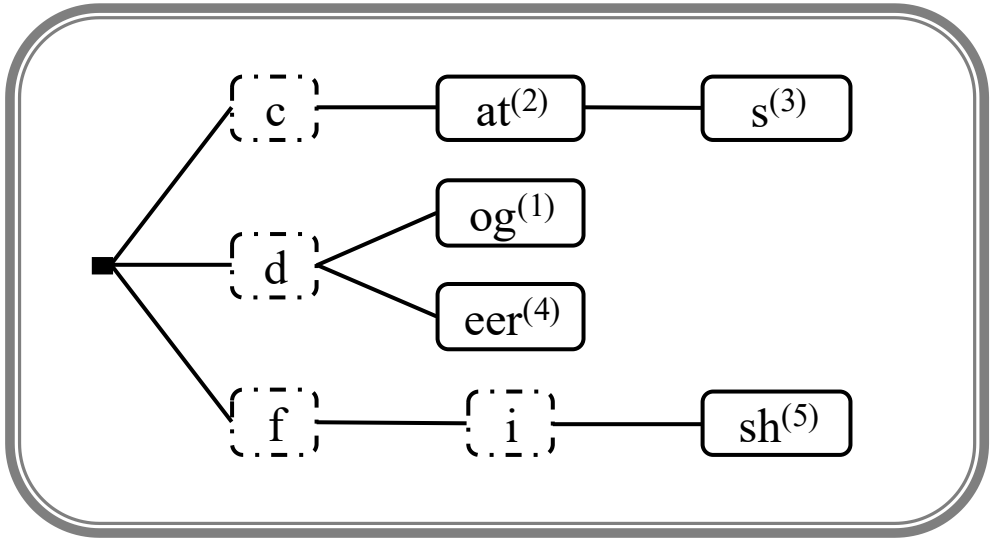


Teacher Forcing with One Hot Supervision



Teacher Forcing with Rank-Aware Supervision

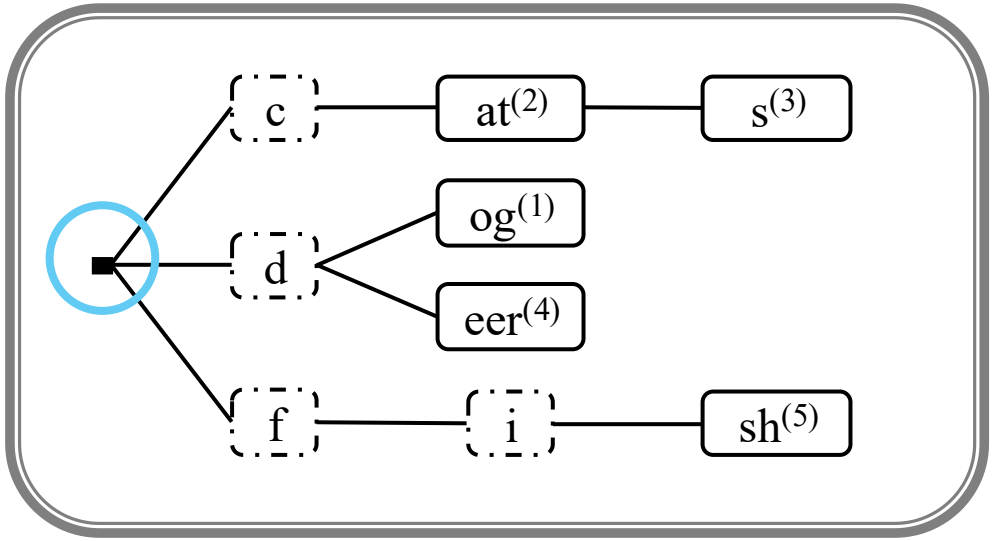
Teacher Forcing with Rank-Aware Supervision



Causal Transformer

■ d og

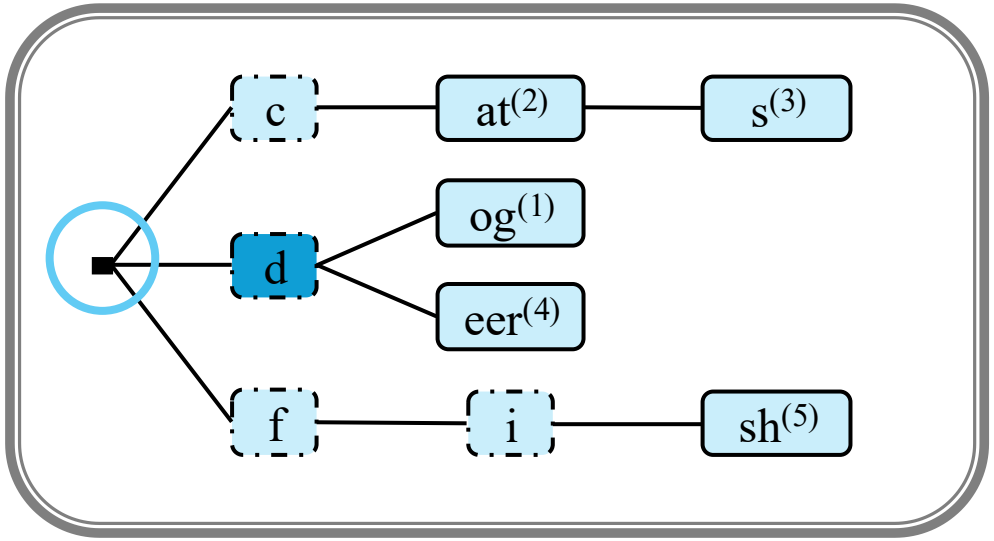
Teacher Forcing with Rank-Aware Supervision



Causal Transformer

■ d og

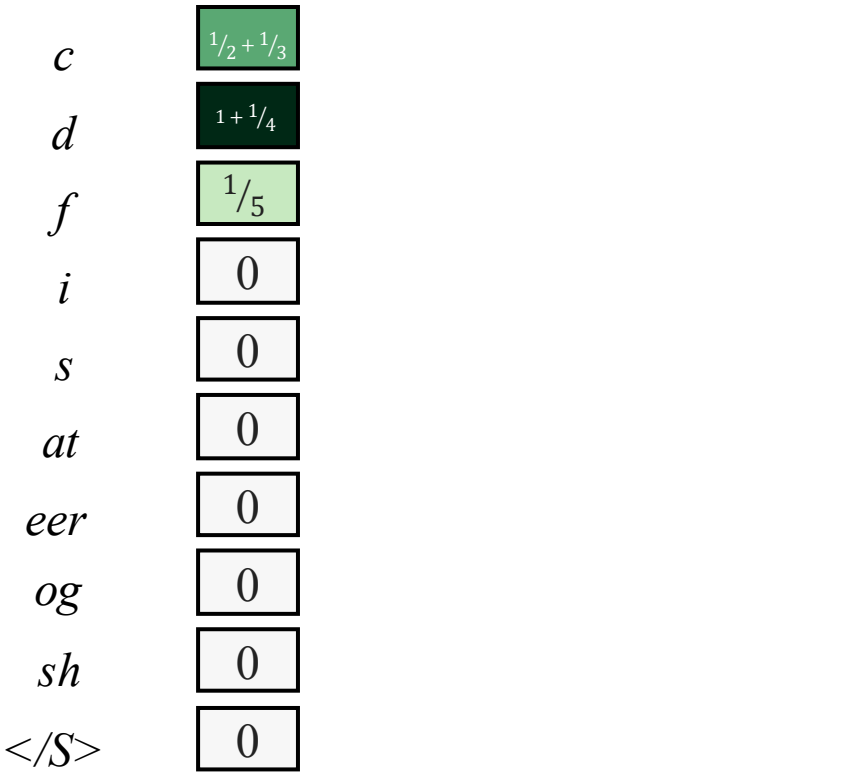
Teacher Forcing with Rank-Aware Supervision

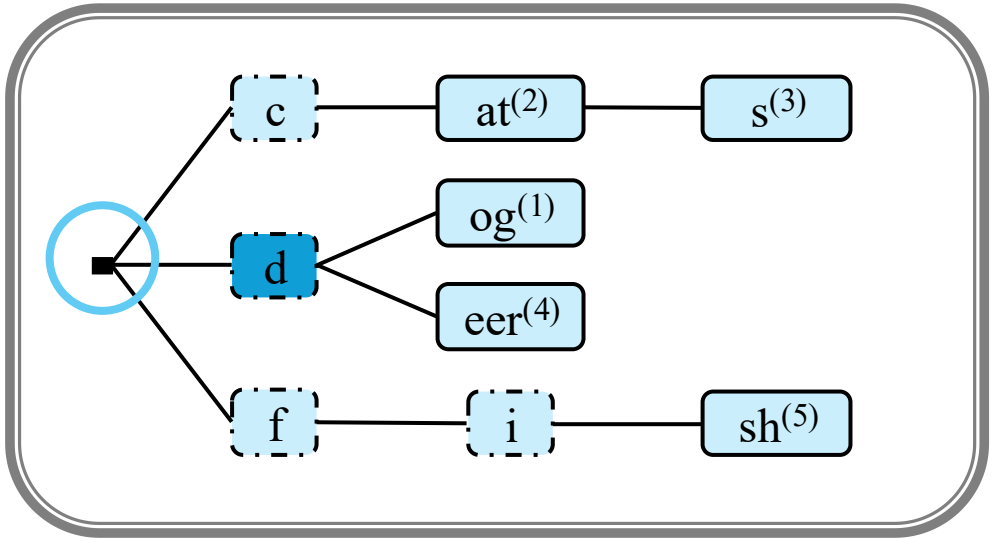


Causal Transformer

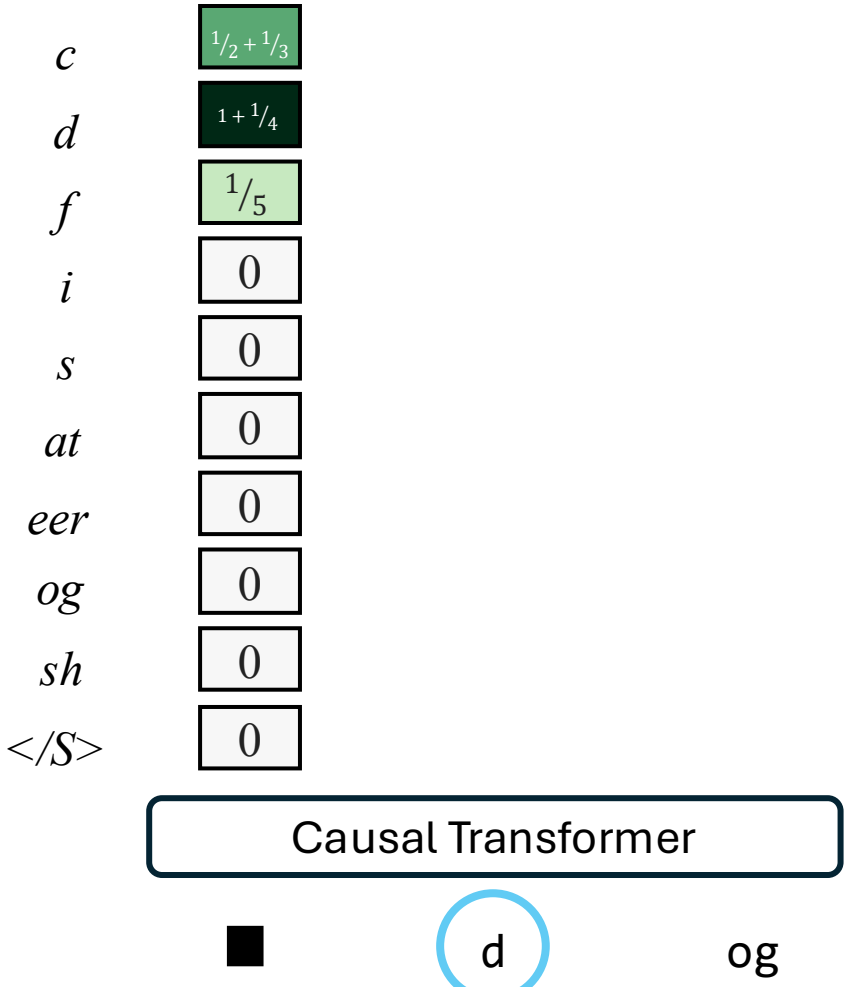
■ d og

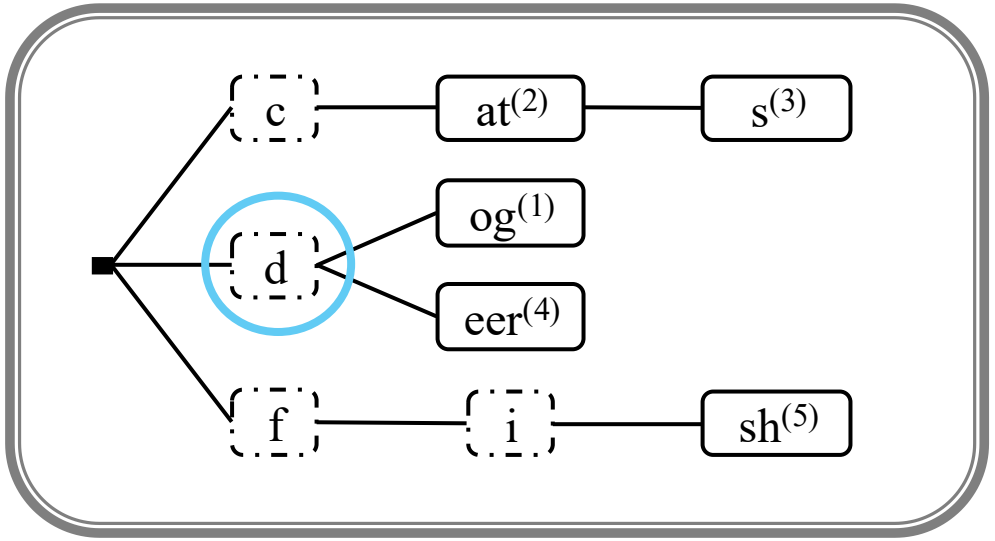
Teacher Forcing with Rank-Aware Supervision



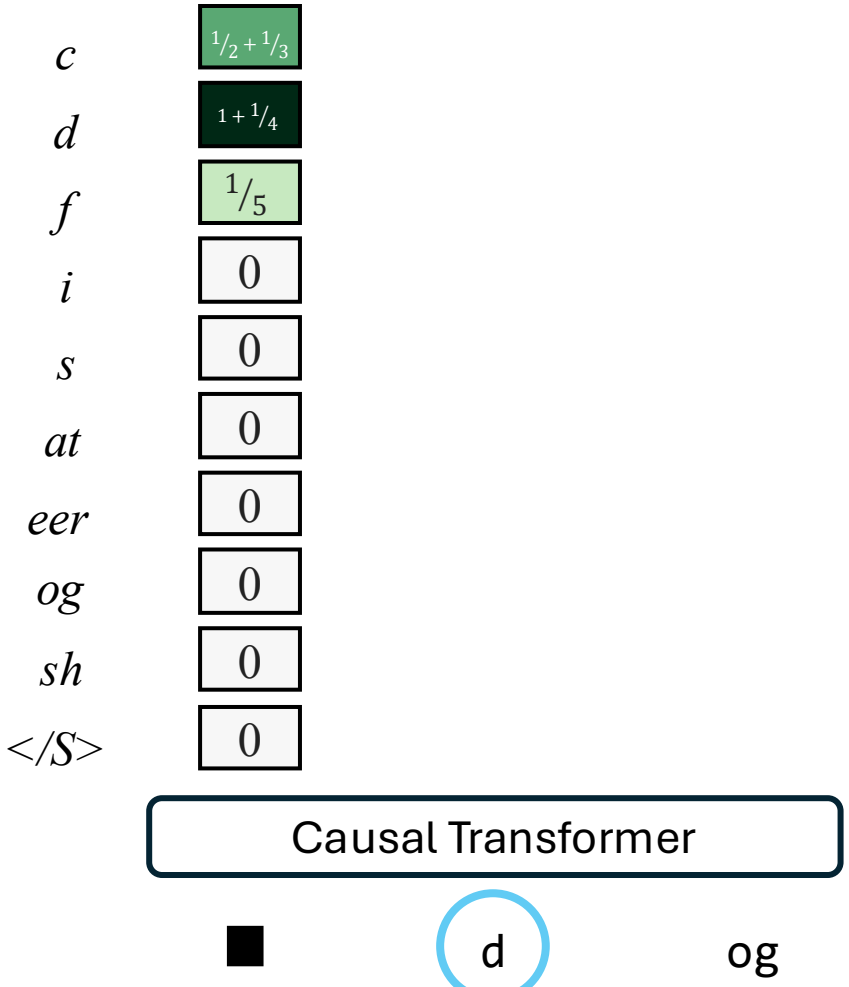


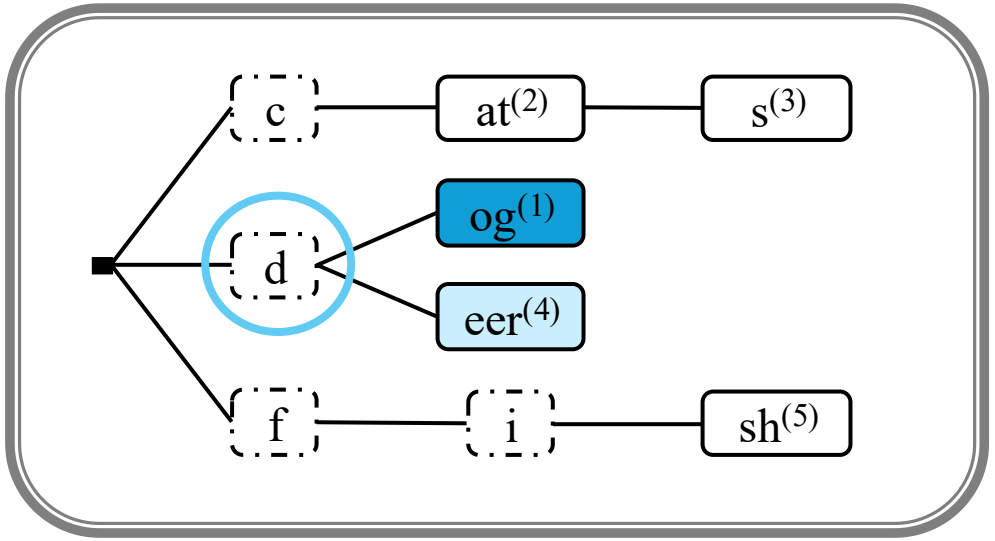
Teacher Forcing with Rank-Aware Supervision



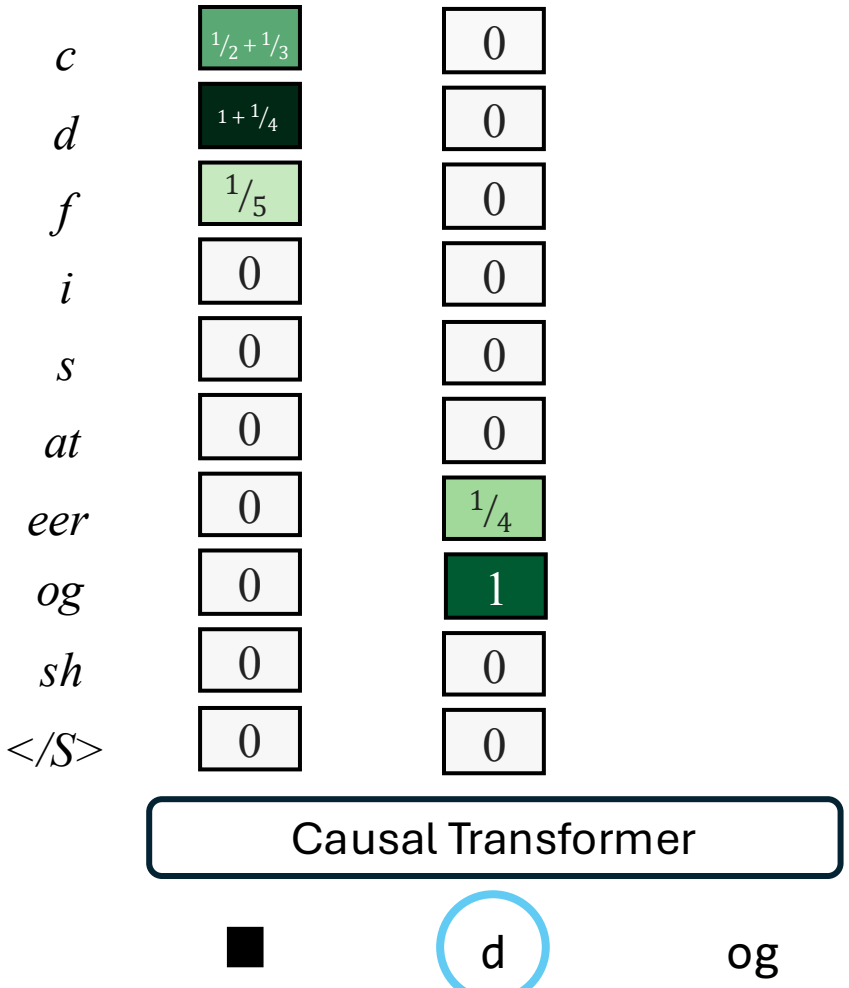


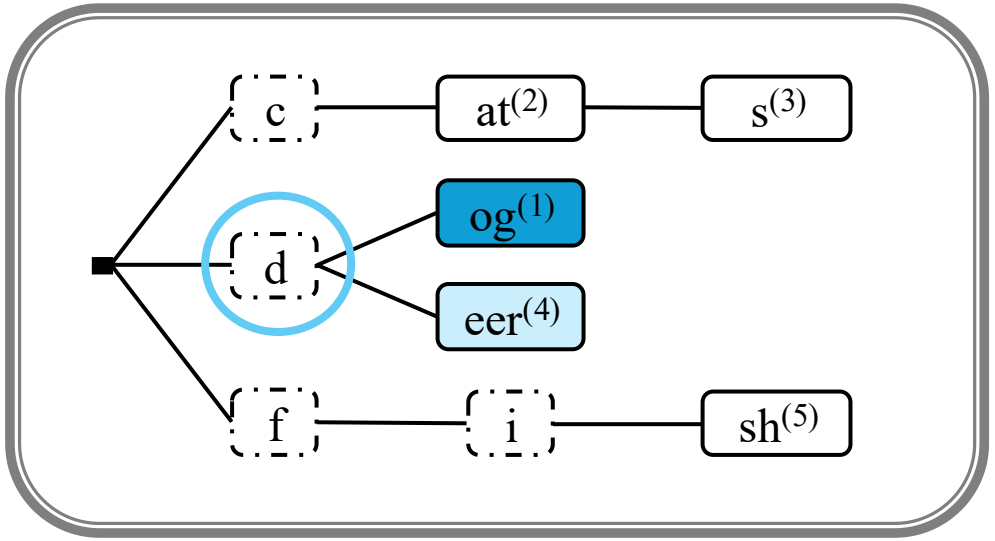
Teacher Forcing with Rank-Aware Supervision



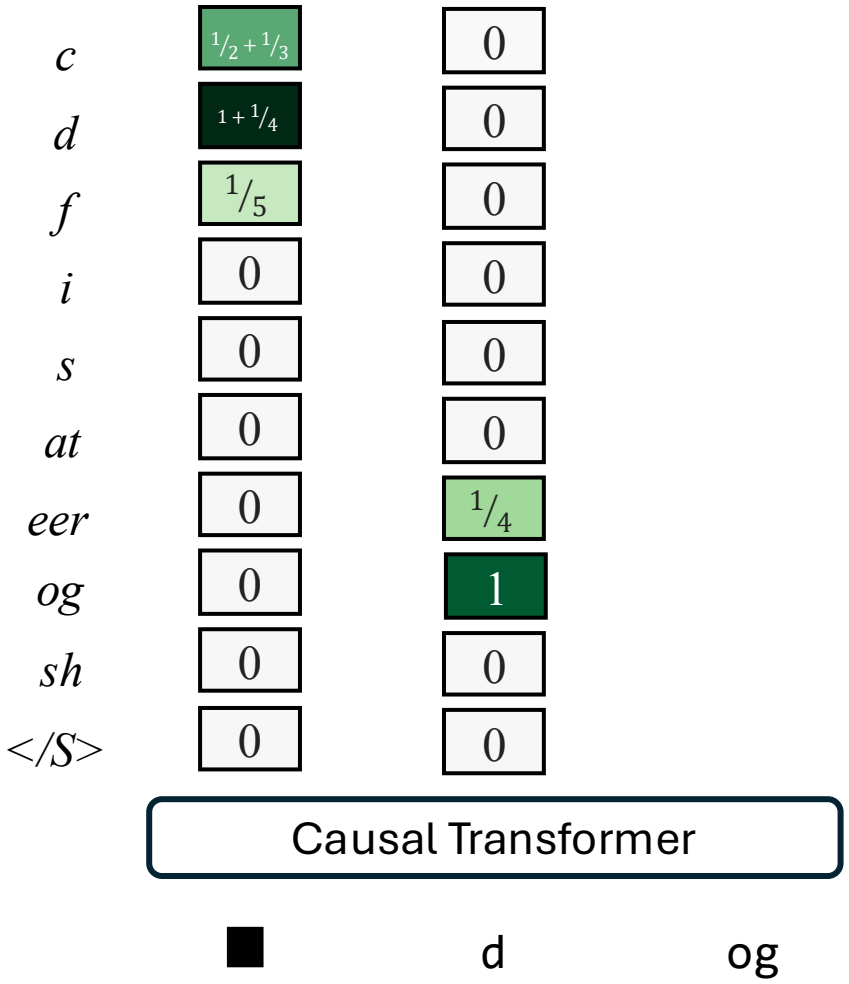


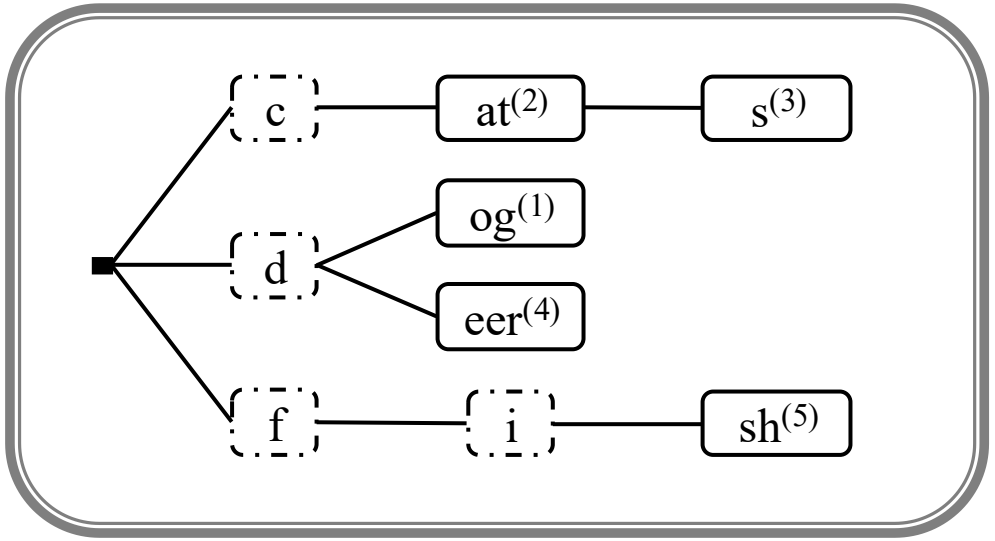
Teacher Forcing with Rank-Aware Supervision



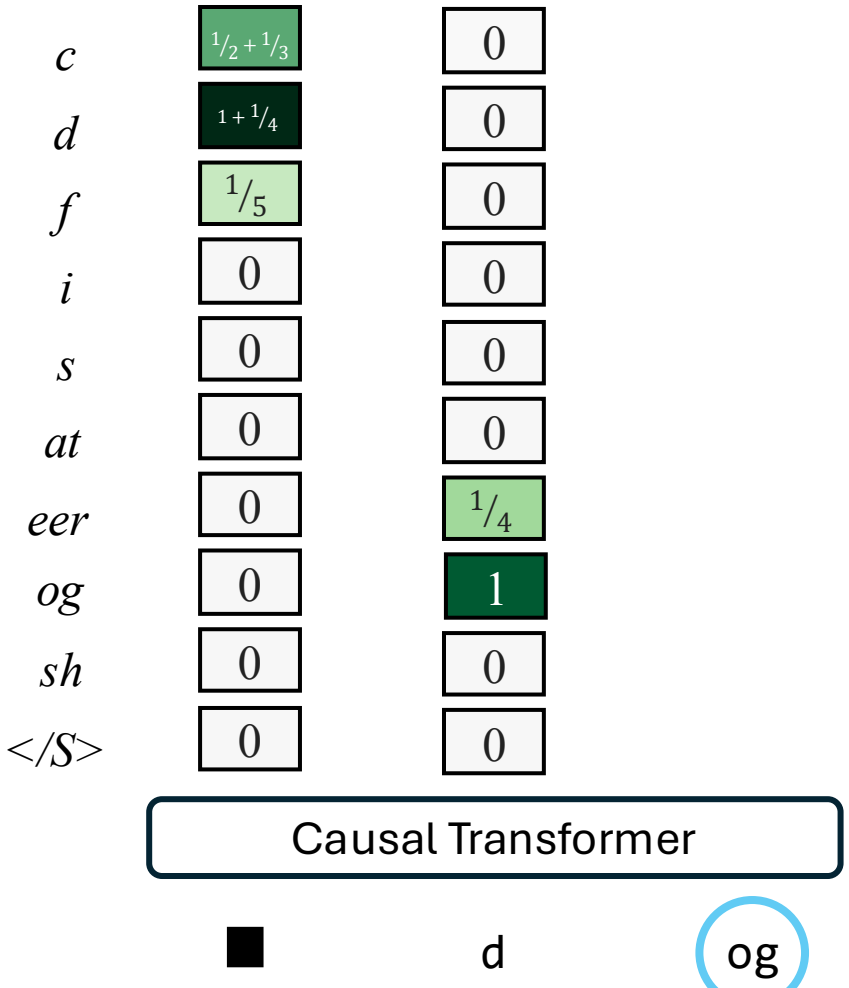


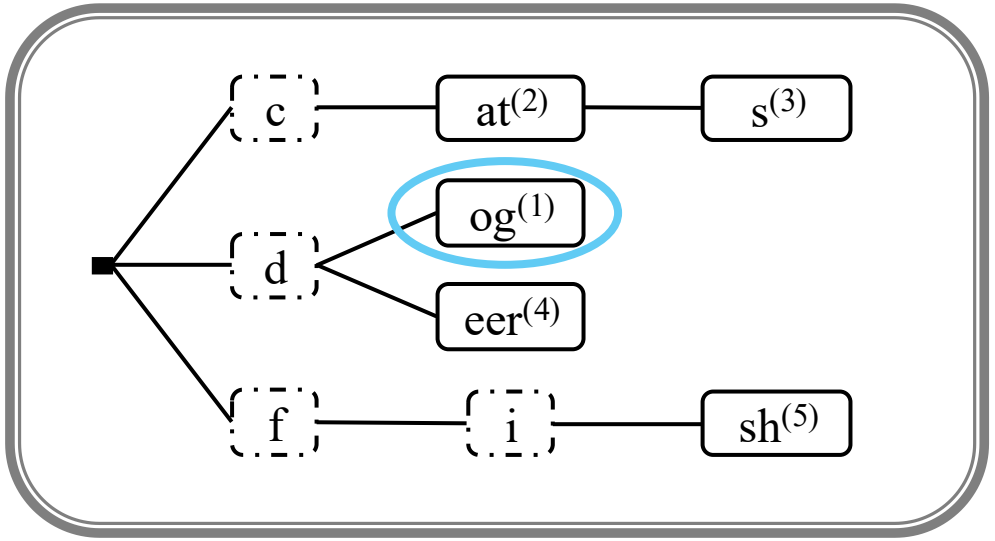
Teacher Forcing with Rank-Aware Supervision



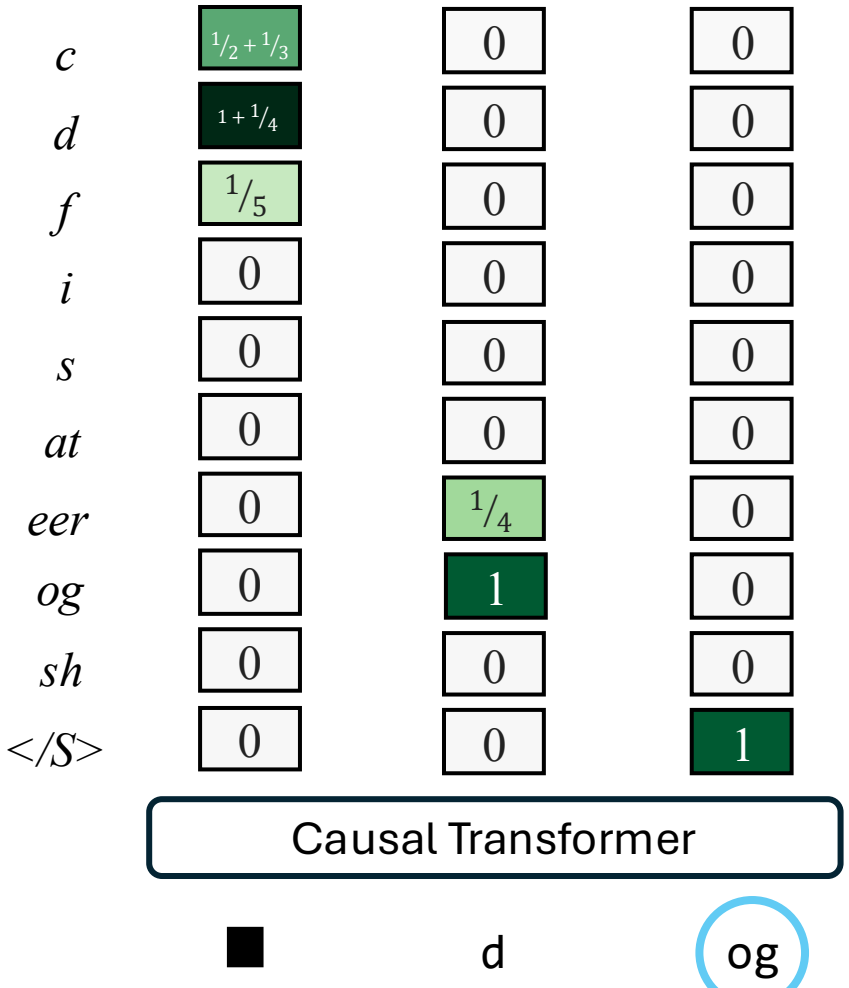


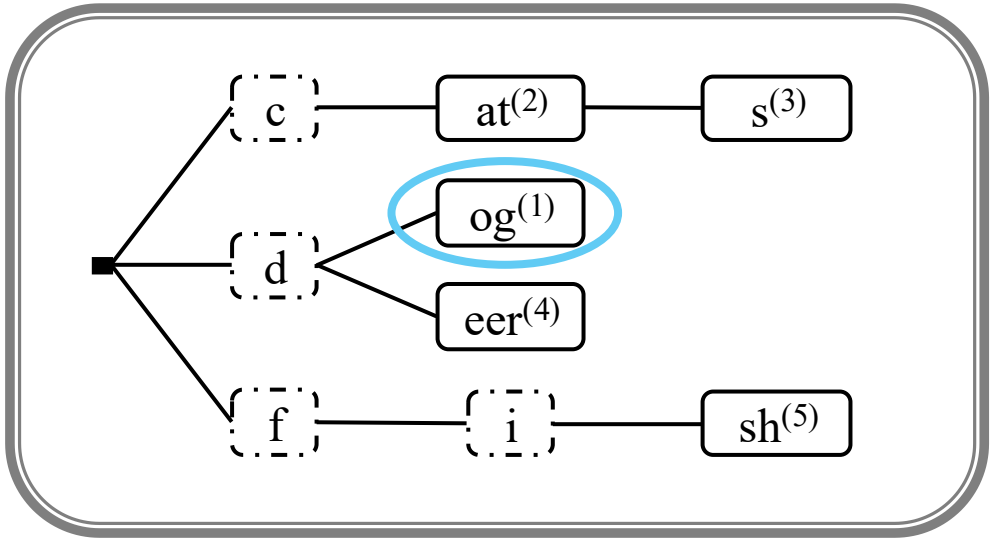
Teacher Forcing with Rank-Aware Supervision



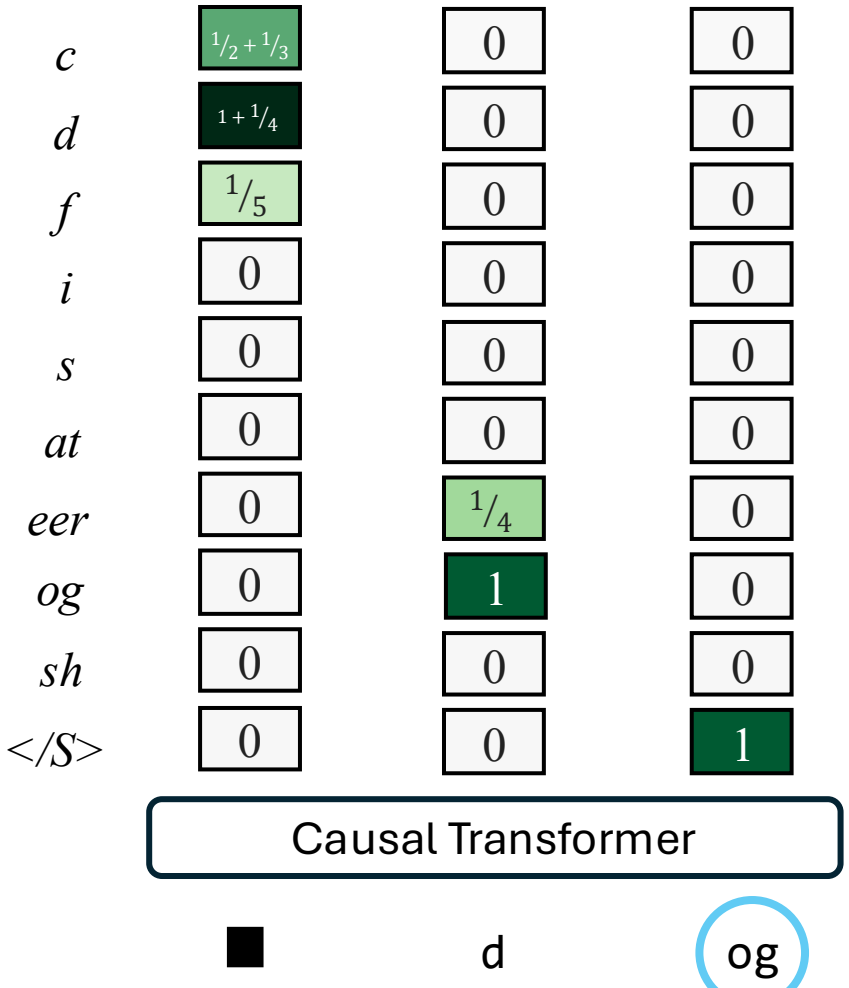


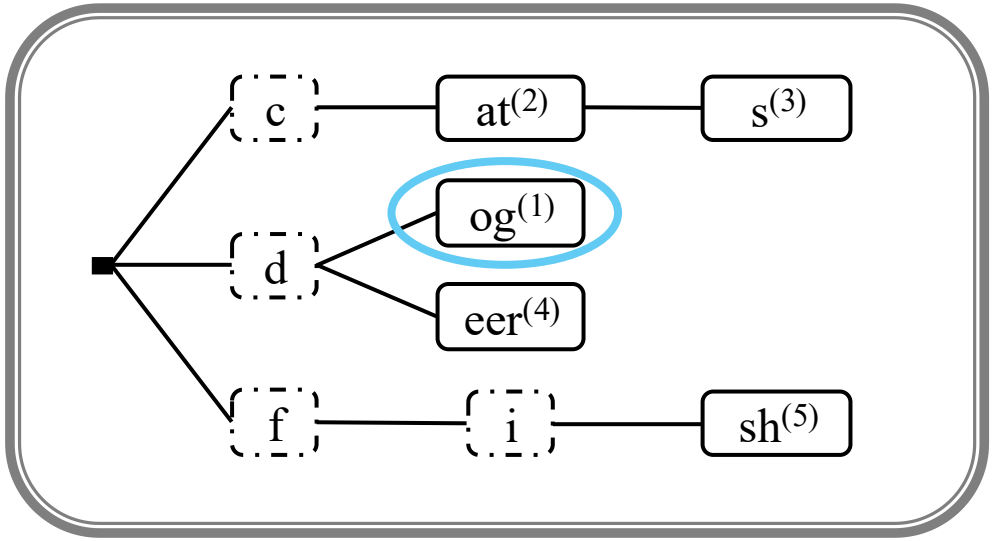
Teacher Forcing with Rank-Aware Supervision





Teacher Forcing with Rank-Aware Supervision





** Leaf weights can be controlled by temperature β*

Datasets & DocID Creation

Datasets & DocID Creation

- WordNet
  - Query: *deer*
  - Ranked docIDs: *ruminant ≻ even-toed ungulate ≻ ungulate ≻ placental ≻ mammal ≻ vertebrate ≻ chordate ≻ animal ≻ organism ≻ living thing ≻ whole ≻ object ≻ physical entity ≻ entity*
  - Negative docID: *solarization*

Datasets & DocID Creation

- WordNet
  - Query: *deer*
  - Ranked docIDs: *ruminant* ≻ *even-toed ungulate* ≻ *ungulate* ≻ *placental* ≻ *mammal* ≻ *vertebrate* ≻ *chordate* ≻ *animal* ≻ *organism* ≻ *living thing* ≻ *whole* ≻ *object* ≻ *physical entity* ≻ *entity*
  - Negative docID: *solarization*

- ESCI Shopping Queries
  - Embed queries and product titles with `gecko-1b-en`
  - For each query, take 1st, 100th, …, 10,000th closest product titles
  - Sparse dictionary learning → vectors in $\mathbb{R}^{100}$ with 3 nonzero entries
  - Indices sorted in descending order by absval of entries, e.g., “25, 36, 39”

Metrics

Metrics

- **“Constraint Violation Rate” (CVR)** — *WordNet only*
  - +1 if model assigns higher log-prob to irrelevant doc than to any of the relevant docs

Metrics

- **“Constraint Violation Rate” (CVR)** — *WordNet only*
  - +1 if model assigns higher log-prob to irrelevant doc than to any of the relevant docs

- **nDCG**
  - Between gold scores (below) and predicted log-probs
  - $\text{score}(d_1^+) = \log(n_q + 1), \text{score}(d_2^+) = \log(n_q) \ldots, \text{score}\left(d_{n_q-1}^+\right) = \log(3), \text{score}\left(d_{n_q}^+\right) = \log(2), \text{score}(d_q^-) = \log(1) = 0$

Metrics

- **“Constraint Violation Rate” (CVR)** — *WordNet only*
  - +1 if model assigns higher log-prob to irrelevant doc than to any of the relevant docs

- **nDCG**
  - Between gold scores (below) and predicted log-probs
  - $\text{score}(d_1^+) = \log(n_q + 1), \text{score}(d_2^+) = \log(n_q) \ldots, \text{score}\left(d_{n_q-1}^+\right) = \log(3), \text{score}\left(d_{n_q}^+\right) = \log(2), \text{score}(d_q^-) = \log(1) = 0$

- **Recall @ k**
  - $\text{R@k} = \frac{|\pi[:k] \cap \hat{\pi}[:k]|}{k}$

Metrics

- **“Constraint Violation Rate” (CVR)** — *WordNet only*
  - +1 if model assigns higher log-prob to irrelevant doc than to any of the relevant docs

- **nDCG**
  - Between gold scores (below) and predicted log-probs
  - $\text{score}(d_1^+) = \log(n_q + 1), \text{score}(d_2^+) = \log(n_q) \ldots, \text{score}\left(d_{n_q-1}^+\right) = \log(3), \text{score}\left(d_{n_q}^+\right) = \log(2), \text{score}(d_q^-) = \log(1) = 0$

- **Recall @ k**
  - $\text{R@k} = \frac{|\pi[:k] \cap \hat{\pi}[:k]|}{k}$

- Scores from greedy decoding, in absence of of constrained decoding with beam search:
  - $\text{score}(d_j) = \frac{1}{|\mathcal{T}(d_j)|} \sum_{k=1}^{|\mathcal{T}(d_j)|} \log p_\theta \left(\mathcal{T}(d_j)[k] \mid \mathcal{T}(\bar{q}), \mathcal{T}(d_j)_{<k} \right)$

Results: WordNet

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

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

Results: WordNet

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

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

Results: WordNet

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

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

Results: WordNet

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

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

Results: ESCI

- *ESCI too big to train with item-level reweighting for each item*

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

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

Results: ESCI

- *ESCI too big to train with item-level reweighting for each item*

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

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

- *Trie-based marginalization economical alternative*

Results: ESCI

- *ESCI too big to train with item-level reweighting for each item*

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

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

- *Trie-based marginalization economical alternative*
- *Synergy between token-level loss and trie-friendly docIDs*

CE and DE settings

CE and DE settings

- *Lookup table DEs with weighted batch softmax loss:*

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

CE and DE settings

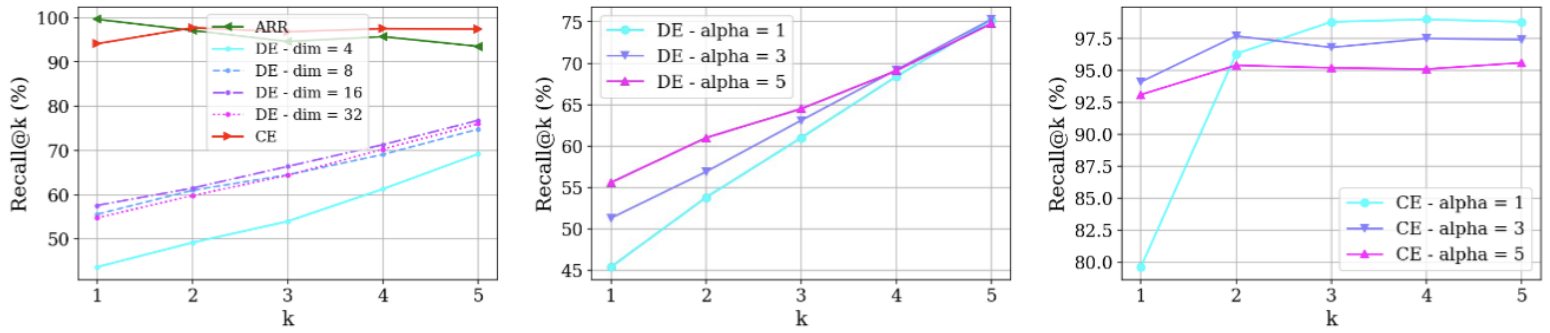
- *Lookup table DEs with weighted batch softmax loss:*

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

- *CEs with randomly chosen permutation for negative set:*

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

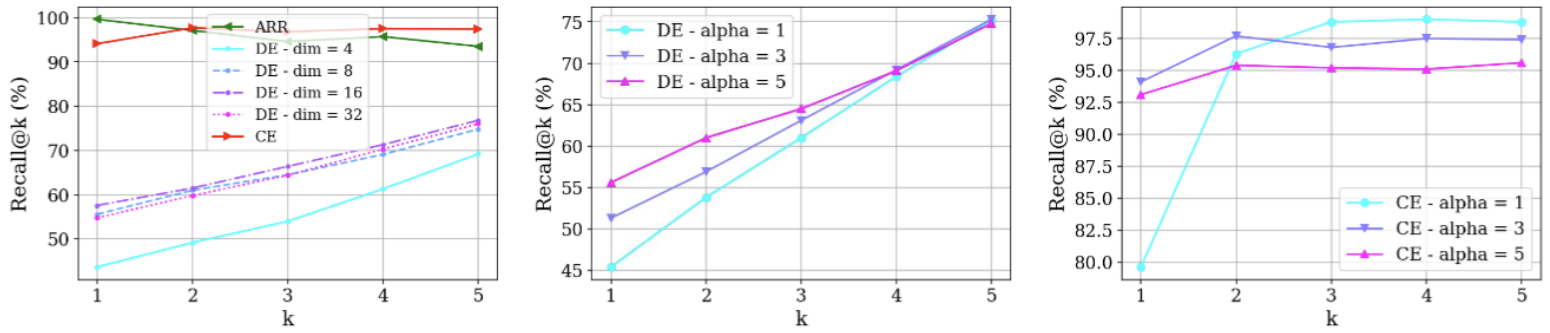
Comparison with DEs and CEs



(a) Comparison with DE/CE (b) Effect of α for DE (b) Effect of α for CE

Figure 3 | Results on WordNet with Dual Encoders (DEs) and Cross Encoders (CEs). (a) Comparison of autoregressive ranking with DEs ($n \in \{4, 8, 16, 32\}, \alpha = 5$) and CE ($n = 32$). (b, c) Effect of α in the reweighting function for DEs ($n = 8$) and CEs ($n = 32$), respectively.

Comparison with DEs and CEs



(a) Comparison with DE/CE (b) Effect of α for DE (b) Effect of α for CE

Figure 3 | Results on WordNet with Dual Encoders (DEs) and Cross Encoders (CEs). (a) Comparison of autoregressive ranking with DEs ($n \in \{4, 8, 16, 32\}, \alpha = 5$) and CE ($n = 32$). (b, c) Effect of α in the reweighting function for DEs ($n = 8$) and CEs ($n = 32$), respectively.

- *ARRs and CEs able to saturate dataset whereas DEs cannot*

Thank you!