RAG-ED: Retrieval-Augmented Generation for Entity Disambiguation

Anonymous ACL submission

Abstract

Entity Disambiguation (ED) resolves ambiguous mentions in text by linking them to entities in a knowledge base. A key challenge in ED is entity overshadowing, where dominant entities obscure the correct choice. We propose RAG-ED (Retrieval-Augmented Generation for Entity Disambiguation), a data efficient three-stage pipeline consisting of a lightweight retriever, reranker, and a strong large language model based selector. RAG-ED achieves state-of-the-art performance on entity overshadowing cases, outperforming prior methods by 17 points. Additionally, the pipeline can also maintain competitive performance across standard ED benchmarks, demonstrating its broad applicability. A key advantage of RAG-ED is its ability to identify instances where disambiguation should not be performed, which is particularly useful in settings relying on lightweight retrievers. We conduct extensive analyses and ablation studies on diverse ED datasets further highlighting the effectiveness of our approach.

1 Introduction

Entity Disambiguation (ED) is a fundamental challenge in Natural Language Processing (NLP), where ambiguous mentions must be correctly linked to entities in a knowledge base (Hoffart et al., 2011). Traditional ED methods rely on fixed candidate sets and static representations, often struggling with *entity overshadowing* (Provatorova et al., 2021), a scenario where a well-known entity dominates over a lesser-known but contextually correct one. This phenomenon leads to systematic biases in entity disambiguation, particularly when handling overshadowed entities.

Retrieval-Augmented Generation (RAG) has demonstrated promising results across various NLP tasks (Lewis et al., 2020b) and holds significant potential for ED (Cao et al., 2021; Mrini et al., 2022). By dynamically retrieving relevant contextual information, RAG enables more flexible and context-aware entity selection. However, integrating retrieval with instruction-following large language models (LLMs) poses unique challenges, including managing retrieval efficiency, reranking noisy candidate sets, and optimizing final selection within constrained context windows.

In this study, we introduce RAG-ED, Retrieval Augmented Generation for Entity Disambiguation, a novel three-stage ED pipeline that leverages: 1. An efficient retriever (e.g., Wikipedia API, or a BLINK-like encoder (Wu et al., 2020a)) to ensure broad coverage. 2. A computationally lightweight reranker to prioritize the most relevant candidates. 3. A strong selector to make the final entity selection which takes advantage of the text understanding and instruction-following capabilities of LLMs such as GPT-4 (OpenAI, 2023) or Llama-3 (AI@Meta, 2024).

Previous ED solutions leverage machine learning and deep learning models (Shen et al., 2021; Sevgili et al., 2022). While these methods demonstrate strong performance in many scenarios, they often struggle with disambiguating complex mentions, scaling to large knowledge bases, and handling overshadowing. Unlike these approaches that rely on fixed ranking mechanisms, our method allows for context-dependent candidate refinement, maximizing accuracy while minimizing computational costs.

We evaluate RAG-ED using the ZELDA benchmark (Milich and Akbik, 2023), focusing on the entity overshadowing ShadowLinks datasets (Provatorova et al., 2021). Evaluation is performed using both closed-source and open-source LLMs as selectors. While closed-source models have demonstrated strong performance, open-source alternatives offer greater adaptability and accessibility. To further enhance performance of the open-source LLM, we apply lightweight fine-tuning to it, demonstrating the feasibility of adapting such models to ED.

Additionally, we investigate an under-explored aspect of ED: handling cases when the retriever/reranker fails. Most ED models assume the correct entity is always present in the candidate set, leading to forced and often incorrect predictions. Our approach allows LLMs to abstain from incorrect predictions, explicitly stating when none of the given candidates are correct. This capability is crucial because erroneous entity linking can provide incorrect information to downstream tasks (Zhu et al., 2023).

Through this work, we aim to establish an efficient and scalable pipeline for ED that balances retrieval cost, reranking efficiency, and LLM-based selection. Our contributions are as follows:

1. a retriever-reranker-selector pipeline for ED;
2. experiments showing the effectiveness of our approach in handling entity overshadowing scenarios which acheives state-of-the art results showing a 17 point improvement in this scenario;
3. an investigation into candidate selection in RAG for ED; and
4. a fine tuned open-source LLM for candidate selection.

The source code for our methods is provided as supplementary material and released publicly [1].

The rest of the paper is organized as follows: Section 2 reviews related work. Section 3 describes our proposed approach. Section 4 describes the experiments performed. Section 5 discusses the results and our findings, and Section 6 concludes the paper.

2 Related Work

Entity Disambiguation: Traditional entity disambiguation methods typically involve candidate selection using techniques like TF-IDF (Aizawa, 2003) and word2vec (Mikolov et al., 2013), followed by reranking with models such as LSTMs (Hochreiter and Schmidhuber, 1997) or BERT (Devlin et al., 2019). More advanced architectures, such as CNNs (Francis-Landau et al., 2016) and cross-encoders like BLINK (Wu et al., 2020a), capture richer contextual relationships, improving entity ranking accuracy. Additionally, autoregressive models like BART (Lewis et al., 2020a) have been explored for entity disambiguation (Cao et al., 2021), leveraging generative capabilities to refine entity selection. However, these methods still face challenges in high-ambiguity scenarios, particularly when dealing with short or contextually coarse-grained texts (Shi et al., 2024; Liu et al., 2024a).

While end-to-end entity disambiguation models like ReFinED (Ayoola et al., 2022) streamline processing by integrating mention detection, entity typing, and disambiguation into a single forward pass, they can be constrained by fixed retrieval mechanisms and lack flexibility in ambiguous contexts. In contrast, our approach leverages RAG and instruction-following LLMs to refine disambiguation by dynamically retrieving and selecting entities.

Retrieval-Augmented Generation: The RAG paradigm combines retrieval mechanisms with generative models, achieving success across tasks like question answering and summarization (Lewis et al., 2020b; Karpukhin et al., 2020; Vig et al., 2022). In entity disambiguation, RAG enables retrieval of relevant candidates and their refinement via generative models (Xiao et al., 2023; Orlando et al., 2024). Although effective, existing methods often struggle to distinguish between retrieval errors and limitations of generative models. Because of the retrieval portion of RAG approaches, there is a potential for the correct candidate entity not to be retrieved. We test the impact of this later in the paper. This problem is somewhat similar to the problem of NIL entities (Zhu et al., 2023) where the goal is to determine when there is no entity match for a mention. Our setting is different in that we do not assume that the retrieval is correct.

LLMs in Entity Disambiguation: LLMs have transformed NLP with their advanced contextual understanding, instruction-following behavior (Ouyang et al., 2022), and in-context learning capabilities (Brown et al., 2020; Touvron et al., 2023). In entity disambiguation, LLMs have been employed to enrich candidate entity descriptions (Xin et al., 2024) or refine predictions in multi-step reasoning frameworks (Liu et al., 2024b). These methods often introduce additional computational overhead and complexity due to the need for open-ended text generation for entity explanations and iterative reasoning steps.

[1] https://anonymous.4open.science/r/RAG-ED-33B8

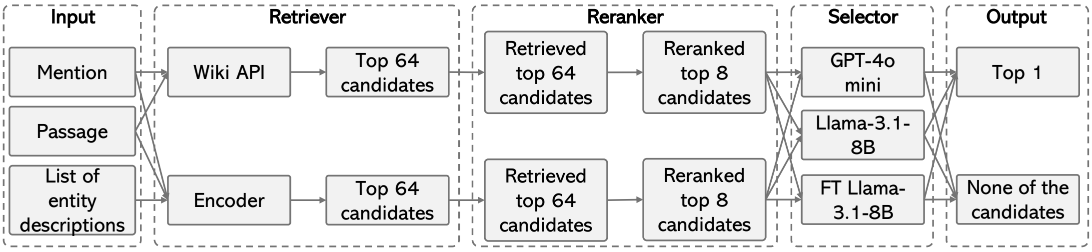


Figure 1: Architecture of RAG-ED, illustrating the input, the Retriever, Reranker, Selector modules, and the output.

3 Proposed approach

3.1 Problem statement

ED is defined as follows: given a textual passage as a sequence of words $d = (w_1, \ldots, w_t)$, together with a set of mentions $m = (m_1, \ldots, m_n)$ where each m_i is a selection of one or several words in d that span mentions of entities. We are additionally given a knowledge base (such as Wikipedia) consisting of a fixed list of entities $\mathcal{E}$. The goal of the ED task is to find a function $f(d, m_i) = e_i \in \mathcal{E}$ that maps each of the mentions in the passage to the correct entity in the knowledge base.

3.2 Approach Overview

We propose to solve the ED problem by making use of three stages, aimed at fast retrieval of candidates for mentions with high recall in the initial stage, followed by improved precision over smaller sets of candidates to control computational cost. These three stages consist of a **retriever**, a **reranker**, and a **selector**.

Retriever. Given a passage-mention pair (d, m_i), the retriever is tasked with ranking all entities in $\mathcal{E}$ as candidates for the mention. A crucial requirement for the retriever is computational efficiency, as in practical applications the set $\mathcal{E}$ can grow to millions of entities[2]. For the majority of texts, only a small subset of $\mathcal{E}$ is likely to be relevant, which motivates prioritizing recall over precision at this stage.

Reranker. At this stage, we select a subset of candidates by taking the top-k_r entities retrieved by the retriever. For each of these entities, we form an input consisting of their description together with the input passage, which we pass to a cross-encoder model that computes a score for the relevance of each candidate entity. Given that this model needs to jointly process the candidate and the passage, it requires more resources than the retriever, though over a smaller set of entities of size $k_r \ll |\mathcal{E}|$.

Selector. In the last stage, we select a smaller set of candidates by taking the top $k_s < k_r$ entities according to the scores computed by the reranker. This final stage is aimed at high precision, thus we rely on directly prompting an LLM by providing it with the descriptions of the k_s candidate entities, the textual passage, and the task of deciding which candidate entity corresponds to the mention query.

We illustrate an overview of the architecture of RAG-ED in Fig. 1. In the following sections we describe the specific details of each of the modules in our implementation of the RAG-ED pipeline.

3.3 Retriever

We consider two approaches for fast retrieval of entities over a large knowledge base: an embedding-based Encoder method, and the Wikipedia API, which relies on methods such as fuzzy search and string matching.

Encoder. We employ the bi-encoder architecture proposed in BLINK by Wu et al. (2020b)[3] as one of our retrievers. BLINK ranks a list of candidate entities based on their likelihood of being the correct match for a given entity. The bi-encoder module uses two independent BERT models (Devlin et al., 2019) to encode the mention context and entity description into dense vectors $y_m, y_e \in \mathbb{R}^d$. The similarity between a mention and an entity candidate is computed using the dot product of these vectors, which scales well to retrieval over large sets of entities (Malkov and Yashunin, 2018).

Wikipedia API. The Wikipedia API serves as a lightweight and efficient retrieval mechanism to generate potential entity candidates for ED tasks.

[2]For reference, Wikipedia has over 50 million pages as of 2025.

[3]We use the implementation available at https://github.com/facebookresearch/BLINK.

This method is based on string matching and keyword search to identify relevant entities from the vast knowledge base of Wikipedia. To retrieve entity candidates, we query the Wikipedia API's `search` endpoint, which identifies articles matching a provided string or keyword. For each identified candidate, we then query the `extracts` endpoint of the Wikipedia API using the page ID. This retrieves the introductory section of the article, typically providing a concise and informative summary.

3.4 Reranker

Given the list of entities retrieved by the previous module, we take the top k_r entities together with their titles, unique identifiers, and a short textual description. For each mention m_i in the input passage d, we construct the following sentence:

```
What does the entity <mention> refer to in
the following sentence?
<input passage>
```

We pass this sentence, together with the description of each of the k_r candidates from the retriever, to a cross-encoder model that outputs a score indicating the relevance of the candidate. We employ a Sentence-BERT (SBERT) Cross-Encoder (Reimers and Gurevych, 2019)[4]. This model helps improve the ranking of candidates, ensuring that the most contextually relevant entities are prioritized before being passed to the last stage.

3.5 Selector

The goal of the last stage is to improve precision by prompting an instruction-tuned LLM with the task of selecting the right entity from a the top k_s candidates obtained from the reranker module. From the previous stage, we prepare a prompt describing the ED task, the input passage, and the list of candidates and their descriptions (a detailed description and examples of our prompt are shown in Appendix A). We use this to prompt the LLM to generate an output indicating which is the right candidate entity for a given mention. In comparison with the retriever and reranker modules, the selector requires a slight increase in computation, which is offset by the small set of $k_s < k_r \ll |\mathcal{E}|$ candidates that have been filtered out by the initial stages and allows us to focus on increasing precision at the final stage.

Importantly, at this stage we enable the LLM to **reject the list of candidates**. This can be useful to detect and address recall problems in the first two stages, as an alternative to forcing the selector to choose a potentially incorrect entity.

We experiment with three LLMs for the selector:

GPT-4o-mini. A compact variant of GPT-4 (OpenAI, 2023), optimized for high performance in resource-constrained environments. This closed-source model offers state-of-the-art generalization across various NLP tasks, serving as a robust baseline for evaluating our prompt design and retrieval and reranking mechanisms.

Llama-3.1-8B-Instruct. An open-source model from Meta's Llama 3 family (AI@Meta, 2024), designed for generative and instruction-following tasks. Its instruction-tuning enables effective performance in structured tasks like entity disambiguation, making it suitable for our RAG-ED pipeline.

Llama-3.1-8B-QLoRA. We fine-tuned the previous Llama-3.1-8B-Instruct model using QLoRA (Quantized Low-Rank Adapter) (Dettmers et al., 2024) with a subset of training datasets available for the ED task. Fine-tuning was conducted using the Python `unsloth` library. We used a small subset of training examples containing diverse, task-specific examples, including mentions, context, and labeled candidates, to better align the model with the requirements of ED.

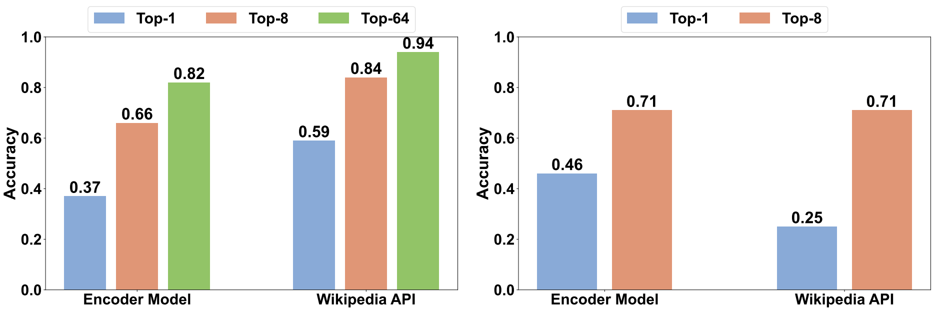
4 Experiments

Our experiments are aimed at answering the following questions:

1. What is the entity disambiguation performance of each of the stages in RAG-ED?
2. How does the performance of RAG-ED compare with respect to state-of-the-art methods for ED?
3. What can RAG-ED do when early stages fail at retrieving correct entities?
4. What is the impact of allowing LLMs in the selector phase to reject lists of candidates on their disambiguation capacity?

We measure ED performance by computing accuracy, which for each mention we define as 1 if the final entity selected by RAG-ED is correct, and 0 otherwise.

[4] We use the implementation available at https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2.



(a) Retriever (b) Reranker

Figure 2: Retriever (a) and Reranker (b) performance on the ZELDA Benchmark: accuracy scores.

4.1 Datasets

The ZELDA benchmark (Milich and Akbik, 2023) provides a unified training dataset, a standardized entity vocabulary, predefined candidate lists, and evaluation splits across multiple domains, enabling rigorous assessment of ED methods. The benchmark consists of the following datasets:

AIDA-B: A test split from the widely used AIDA (Yosef et al., 2011) dataset, featuring 231 manually annotated Reuters news articles.

TWEEKI: A collection of 500 hand-annotated tweets, representing short and highly ambiguous contexts (Harandizadeh and Singh, 2020).

REDDIT-POSTS and REDDIT-COMMENTS: Two datasets of top-ranking posts and comments from Reddit, focusing on informal and conversational text. Only annotations with full agreement among annotators are included (Botzer et al., 2021).

WNED-WIKI and WNED-CWEB: Wikipedia articles and web pages, annotated with difficulty levels for more granular analysis (Guo and Barbosa, 2018).

ShadowLink: This is a dataset aimed at investigating the problem of *entity overshadowing* (Provatorova et al., 2021). Entity overshadowing is a phenomenon in entity linking where a prominent or frequently mentioned entity overshadows less common but contextually appropriate entities in the candidate selection or linking process. It contains instances of the following cases:

SLINKS-TOP: High ambiguity, but the correct answer is the most frequent entity.

SLINKS-SHADOW: High level of ambiguity, and the correct answer is overshadowed by a more popular entity.

SLINKS-TAIL: Control group, low level of ambiguity but contains only "long tail" entities that are very rare in Wikipedia.

4.2 Methods

RAG-ED. We experiment with different variants of RAG-ED, investigating the effect of each module. In one variant, we employ an Encoder→Selector pipeline, in which we keep the top $k_r = 8$ retrieved entities which we then pass directly to the Selector. In a second variant, we employ a pipeline Encoder→Reranker→Selector where we keep the top $k_r = 64$ entities from the Retriever, which we pass to the Reranker to keep the top $k_s = 8$ that we pass to the last Selector stage. For these two variants, we experiment with two settings: using a BLINK-based Encoder, or the Wikipedia API for the Retriever, which leads to a total of four variants.

Baselines We compare our method with FEVRY (Févry et al., 2020), LUKE (Yamada et al., 2022), and GENRE (Cao et al., 2021) - the current state-of-the-art in ED. We refer to their performance as reported in Milich and Akbik (2023).

5 Results

5.1 Retriever Performance

Retrievers play a crucial role in the RAG-ED pipeline, as LLMs can only disambiguate entities when the correct candidate is included in the retrieved set. To assess retrieval effectiveness, we compare our Encoder as retriever with the Wikipedia API retriever, measuring their accuracy in retrieving the correct entity within the top 1, 8, and 64 candidates (Figure 2a).

The Encoder as retriever, described in Section 3.3, achieves an accuracy of 0.37 at the top-1 candi-

	AIDA-B	TWEEKI	REDDIT-POSTS	REDDIT-COMM.	WNED-CWEB	WNED-WIKI	SLINKS-TAIL	SLINKS-SHADOW	SLINKS-TOP	∅
Pipeline 1: *Encoder-Top 8 (Retriever) => LLM (Selector)*										
RAG-ED$_{GPT}$	0.68	0.57	0.62	0.60	0.57	0.67	0.73	0.51	0.53	0.61
RAG-ED$_{Llama}$	0.42	0.43	0.40	0.37	0.35	0.46	0.71	0.42	0.43	0.44
RAG-ED$_{FT}$	0.66	0.57	0.61	0.57	0.52	0.65	0.76	0.47	0.51	0.59
Pipeline 2: *API-Top 8 (Retriever) => LLM (Selector)*										
RAG-ED$_{GPT}$	0.76	0.79	**0.92**	**0.92**	0.65	0.65	0.98	0.36	0.62	0.74
RAG-ED$_{Llama}$	0.52	0.49	0.43	0.43	0.38	0.41	0.80	0.28	0.48	0.47
RAG-ED$_{FT}$	0.72	0.75	0.87	0.82	0.58	0.61	0.96	0.23	0.51	0.67
Pipeline 3: *Encoder-Top 64 (Retriever) => SBERT-Top8 (Reranker) => LLM (Selector)*										
RAG-ED$_{GPT}$	0.68	0.65	0.72	0.73	0.48	0.64	0.87	**0.60**	**0.64**	0.66
RAG-ED$_{Llama}$	0.38	0.42	0.44	0.47	0.30	0.43	0.80	0.49	0.52	0.47
RAG-ED$_{FT}$	0.56	0.64	0.70	0.69	0.45	0.62	0.86	0.57	0.61	0.63
Pipeline 4: *API-Top 64 (Retriever) => SBERT-Top 8 (Reranker) => LLM (Selector)*										
RAG-ED$_{GPT}$	0.54	0.72	0.77	0.73	0.34	0.51	0.97	0.55	0.56	0.63
RAG-ED$_{Llama}$	0.38	0.50	0.44	0.47	0.22	0.34	0.81	0.43	0.42	0.45
RAG-ED$_{FT}$	0.51	0.66	0.71	0.61	0.31	0.47	0.95	0.43	0.46	0.58
Other approaches *(Milich and Akbik, 2023)*										
FEVRY$_{ALL}$	0.79	0.71	0.88	0.84	0.68	0.84	0.63	0.43	0.53	0.70
FEVRY$_{CL}$	0.79	0.76	0.89	0.86	0.70	0.84	0.87	0.31	0.47	0.72
LUKE$_{PRE}$	0.79	0.73	0.76	0.69	0.66	0.68	0.97	0.20	0.50	0.67
LUKE$_{FT}$	**0.81**	0.77	0.81	0.78	0.70	0.76	0.98	0.22	0.51	0.71
GENRE$_{ALL}$	0.72	0.75	0.88	0.83	0.66	0.85	0.95	0.38	0.43	0.72
GENRE$_{CL}$	0.78	**0.80**	**0.92**	0.91	**0.73**	**0.88**	**0.99**	0.37	0.52	**0.77**

Table 1: Performance of the RAG-ED pipelines on the ZELDA Benchmark across different candidate generation and selection methods. In this evaluation, the selector must choose an answer from the given candidate list and is not permitted to reject all options. The highest scores are **bolded**, and the second-highest scores are underlined.

date, improving to 0.66 at top-8 and 0.82 at top-64. In comparison, the Wikipedia API retriever outperforms it, achieving 0.59, 0.84, and 0.94 under the same conditions. These results highlight the strength of the Wikipedia API, particularly in low-context settings, where retrieval is based solely on the entity mention.

5.2 Reranking Performance

Using all 64 retrieved candidates in a single prompt is impractical due to context length limitations. It also makes the task of selector more difficult. To refine the candidate set, we employ an SBERT CrossEncoder as a reranker, aiming to improve ranking quality by pushing the correct entity into the top-8 candidates. As shown in Figure 2b, reranking significantly benefits the Encoder retriever, boosting top-1 accuracy from 0.37 to 0.46 and top-8 accuracy from 0.66 to 0.71. However, its effect on the Wikipedia API retriever is negative, with top-1 accuracy dropping from 0.59 to 0.25 and top-8 accuracy decreasing from 0.84 to 0.71.

This disparity suggests that while reranking enhances less-structured retrieval outputs, it may disrupt the ranking of already well-ordered candidates. One possible explanation lies in the difference in candidate descriptions: the Encoder retriever uses descriptions from the ZELDA candidate list, while the API retriever relies on Wikipedia API descriptions, which can vary in length and detail. The reranker, trained on well-defined entity descriptions, may struggle to interpret these pages correctly, leading to misplacement of relevant candidates and ultimately reducing retrieval accuracy.

5.3 RAG-ED Performance

Table 1 compares the performance of RAG-ED pipelines against other approaches across ZELDA test sets. The top-performing RAG-ED model achieves an overall accuracy of 0.74, closely approaching the state-of-the-art 0.77 achieved by GENRE$_{CL}$. In this evaluation setup, the selector must choose an answer from the provided candidate list, as previous work does not assess whether models can reject incorrect candidate sets. This setting allows for a direct comparison between RAG-ED and prior methods, highlighting the competitiveness and effectiveness of our approach in entity disambiguation.

5.3.1 Performance on Overshadowing Cases

RAG-ED$_{GPT}$ in Pipeline 3 (Table 1) sets a new state-of-the-art in mitigating entity overshadowing. It outperforms the previous state-of-the-art model, FEVRY$_{ALL}$, with a substantial +17-point gain on the SLINKS-Shadow dataset and an +11-

	AIDA-B	TWEEKI	REDDIT-POSTS	REDDIT-COMM.	WNED-CWEB	WNED-WIKI	SLINKS-TAIL	SLINKS-SHADOW	SLINKS-TOP	∅
Pipeline 1: *Encoder-Top 8 (Retriever) => LLM (Selector)*										
RAG-ED$_{GPT}$	**0.83**	**0.83**	0.86	0.86	**0.79**	**0.80**	**0.99**	**0.77**	**0.82**	**0.84**
RAG-ED$_{Llama}$	0.39	0.39	0.32	0.32	0.41	0.45	0.66	0.39	0.40	0.41
RAG-ED$_{FT}$	0.70	0.65	0.68	0.68	0.64	0.69	0.84	0.55	0.59	0.67
Pipeline 2: *API-Top 8 (Retriever) => LLM (Selector)*										
RAG-ED$_{GPT}$	<u>0.79</u>	<u>0.82</u>	**0.92**	**0.93**	0.72	0.73	<u>0.98</u>	0.50	0.69	0.79
RAG-ED$_{Llama}$	0.46	0.48	0.41	0.35	0.41	0.39	0.79	0.22	0.41	0.44
RAG-ED$_{FT}$	0.73	0.76	0.87	0.83	0.61	0.64	0.96	0.23	0.51	0.68
Pipeline 3: *Encoder-Top 64 (Retriever) => SBERT-Top 8 (Reranker) => LLM (Selector)*										
RAG-ED$_{GPT}$	**0.83**	<u>0.82</u>	<u>0.89</u>	<u>0.87</u>	<u>0.77</u>	<u>0.79</u>	**0.99**	<u>0.73</u>	<u>0.81</u>	<u>0.83</u>
RAG-ED$_{Llama}$	0.34	0.38	0.29	0.35	0.40	0.42	0.72	0.39	0.45	0.42
RAG-ED$_{FT}$	0.62	0.71	0.74	0.75	0.62	0.67	0.91	0.61	0.66	0.70
Pipeline 4: *API-Top 64 (Retriever) => SBERT-Top 8 (Reranker) => LLM (Selector)*										
RAG-ED$_{GPT}$	0.67	0.76	0.80	0.77	0.58	0.66	<u>0.98</u>	0.61	0.63	0.72
RAG-ED$_{Llama}$	0.35	0.49	0.38	0.39	0.32	0.34	0.80	0.35	0.39	0.42
RAG-ED$_{FT}$	0.52	0.68	0.72	0.69	0.44	0.52	0.94	0.42	0.47	0.60

Table 2: Performance of the RAG-ED pipelines on the ZELDA Benchmark with Various Candidate Generation and Selection Methods. The table reports accuracy scores calculated by taking true negatives into account, i.e., when models can respond with "None of the given candidates is the correct entity." The highest scores are **bolded**, and the second-highest scores are <u>underlined</u>.

point improvement on SLINKS-Top. As the best-performing solution on these challenging benchmarks, Pipeline 3 demonstrates effectiveness in addressing entity overshadowing. We do note that there is a trade-off lowering performance on less challenging datasets.

5.3.2 Impact of Task-specific Adaptation

Task-specific adaptation is essential for enhancing the performance of open-source models like Llama, which are not inherently optimized for Entity Disambiguation. Our results demonstrate that fine-tuning with the QLoRA method leads to significant improvement in performance, as observed when comparing the accuracy scores achieved by RAG-ED$_{FT}$ and the off-the-shelf RAG-ED$_{Llama}$ (Table 1).

Across all datasets, RAG-ED$_{FT}$ consistently outperforms RAG-ED$_{Llama}$ and comes close to RAG-ED$_{GPT}$. Concretely, in the Pipeline 3, the performance gap between the two is just 3 points (0.66 vs 0.63). This improvement underscores the effectiveness of task adaptation in aligning the model with the specific challenges of Entity Disambiguation. Notably, RAG-ED$_{FT}$ achieves the second-highest accuracy on most challenging datasets such as Slinks-Shadow, demonstrating its enhanced ability to handle complex cases. These findings highlight the added value of lightweight fine-tuning in refining model performance without requiring extensive retraining or substantial computational resources.

5.3.3 What can Selectors do when candidate selection fails?

One of the key advantages of using LLMs as selectors is their ability to assess whether the correct entity is present among the given candidates. Concretely, when the correct entity is absent (i.e., candidate selection fails), the model may reject all options and respond with "None of the candidates given is the correct entity." This capability is crucial for reducing the number of false positives and improving overall reliability.

Table 2 presents accuracy scores in a setting where the model must either select the correct candidate or reject all options when none are correct. In this evaluation, the RAG-ED model achieves an accuracy of 0.84, revealing key insights into the performance dynamics of RAG-ED pipelines. Notably, allowing the model to reject incorrect candidates leads to a 10-point accuracy increase, reaching 0.84. This highlights the LLM's ability not only to select the correct entity but also to recognize when no correct option is present.

Interestingly, while the Encoder Model underperforms compared to the Wikipedia API as a standalone retriever, Table 2 reveals a different trend when selectors are allowed to reject disambiguation. In this setting, Encoder-based pipelines surpass those using the Wikipedia API, suggesting a synergistic effect between the retriever, reranker, and LLM selector. The broader candidate sets retrieved by the Encoder Model appear to benefit from the reranker's filtering and the LLM's refined

	AIDA-B	TWEEKI	REDDIT-POSTS	REDDIT-COMM.	WNED-CWEB	WNED-WIKI	SLINKS-TAIL	SLINKS-SHADOW	SLINKS-TOP	∅
Pipeline 1: *Encoder-Top 8 (Retriever) => LLM (Selector)*										
RAG-ED$_{GPT}$	<u>0.67</u>	0.57	0.62	0.60	0.56	**0.67**	0.76	0.50	0.52	0.61
RAG-ED$_{Llama}$	0.37	0.36	0.31	0.29	0.31	0.44	0.63	0.37	0.38	0.38
RAG-ED$_{FT}$	0.67	0.57	0.61	0.58	0.51	<u>0.65</u>	0.75	0.47	0.51	0.59
Pipeline 2: *API-Top 8 (Retriever) => LLM (Selector)*										
RAG-ED$_{GPT}$	**0.76**	**0.78**	**0.91**	**0.91**	**0.64**	<u>0.65</u>	**0.98**	0.36	<u>0.62</u>	**0.73**
RAG-ED$_{Llama}$	0.46	0.48	0.41	0.35	0.38	0.37	0.79	0.22	0.41	0.43
RAG-ED$_{FT}$	0.72	0.75	<u>0.87</u>	<u>0.83</u>	<u>0.58</u>	0.61	0.96	0.23	0.51	<u>0.67</u>
Pipeline 3: *Encoder-Top 64 (Retriever) => SBERT-Top 8 (Reranker) => LLM (Selector)*										
RAG-ED$_{GPT}$	0.58	0.65	0.72	0.73	0.48	0.63	0.87	**0.59**	**0.63**	0.65
RAG-ED$_{Llama}$	0.32	0.36	0.28	0.33	0.28	0.41	0.71	0.37	0.44	0.39
RAG-ED$_{FT}$	0.56	0.64	0.70	0.69	0.44	0.62	0.86	<u>0.56</u>	0.61	0.63
Pipeline 4: *API-Top 64 (Retriever) => SBERT-Top 8 (Reranker) => LLM (Selector)*										
RAG-ED$_{GPT}$	0.54	<u>0.71</u>	0.76	0.73	0.34	0.51	<u>0.97</u>	0.54	0.55	0.63
RAG-ED$_{Llama}$	0.34	0.49	0.38	0.38	0.21	0.32	0.80	0.35	0.39	0.41
RAG-ED$_{FT}$	0.51	0.67	0.71	0.68	0.31	0.47	0.94	0.41	0.47	0.57

Table 3: Ablation study on the impact of rejecting candidate sets. Accuracy is calculated by treating such responses as incorrect, effectively penalizing the model for rejecting all candidates. The highest scores are **bolded**, and the second-highest scores are <u>underlined</u>.

selection process, ultimately leading to more accurate disambiguation.

The comparison of LLMs as selectors follows a similar trend to Table 1, with RAG-ED$_{GPT}$ consistently achieving the highest performance, followed by RAG-ED$_{FT}$, while RAG-ED$_{Llama}$ lags behind. RAG-ED$_{FT}$ remains competitive when forced to select a candidate, it struggles compared to the GPT-based model in correctly rejecting incorrect candidates.

5.3.4 Does allowing LLMs to reject candidate sets decrease their disambiguation capacity?

We perform an ablation study in which rejecting a candidate set is treated as incorrect. This allows us to assess whether rejecting candidates negatively affects the selector's ability to disambiguate. The results are presented in Table 3.

Comparing this evaluation setup to the results in Table 1, where the LLM is required to select from the given candidate set, we observe that RAG-ED$_{GPT}$ and RAG-ED$_{FT}$ experience only a modest decline in performance (1-2%) when permitted to reject incorrect candidates. This suggests that their disambiguation capabilities remain strong, even with the added task of identifying when no correct entity is present. In contrast, RAG-ED$_{Llama}$ shows a more pronounced performance drop, reaching up to 6% (in both Pipeline 1 and Pipeline 3). This suggests that the Llama model faces greater difficulty when tasked with rejecting incorrect candidates, underscoring the need for task-specific adaptation to improve its performance in such scenarios.

These findings show that RAG-ED$_{GPT}$ and RAG-ED$_{FT}$ retain strong disambiguation performance even when tasked with rejecting candidate sets, indicating that their ability to detect the absence of the correct entity does not undermine their overall disambiguation capabilities. In contrast, the substantial performance drop observed in RAG-ED$_{Llama}$ highlights the need for task-specific fine-tuning.

6 Conclusion

In this work, we introduced RAG-ED (Retrieval-Augmented Generation for Entity Disambiguation), a pipeline that addresses the challenging issue of entity overshadowing. By combining a lightweight retriever, a reranker, and a robust large language model selector, RAG-ED significantly outperforms prior methods in entity overshadowing cases. Our experiments show that RAG-ED excels in complex disambiguation scenarios but can also maintain competitive performance across a range of standard ED benchmarks depending on its configuration. Notably, RAG-ED's ability to reject candidate sets when the correct entity is absent further enhances its reliability, especially in settings with lightweight retrievers. We believe that the ability to use lightweight retrievers like the Wikipedia API makes the implementation of entity disambiguation pipelines easier and more efficient in practice. Going forward, we aim to investigate additional ensembles of selectors and integration the entity disambiguation into a full entity linking pipeline.

Limitations

While RAGED demonstrates strong performance in entity disambiguation, several challenges remain. One limitation is its reliance on lightweight retrievers, such as Wikipedia API and the encoder model as retriever. Although efficient, these retrievers may not always surface the most relevant candidates, particularly for less common entities. A more advanced retrieval strategy could further improve performance.

Another constraint lies in the reranking stage. The reranker employed in our pipeline, SBERT CrossEncoder, is not specifically fine-tuned for entity disambiguation. Using a reranker designed for this task could lead to better candidate ranking and more precise selections.

A key unknown is the extent to which LLMs have encountered these entities during pretraining. It is difficult to determine whether the models are reasoning from context or relying on memorized knowledge.

Generalization across domains is another consideration. While RAG-ED performs well on standard benchmarks, its effectiveness in specialized fields such as medicine or law remains uncertain. Adapting the pipeline to domain-specific datasets could enhance its applicability in those areas.

Finally, the use of LLMs introduces potential biases inherited from their training data. These biases may impact disambiguation decisions, particularly for underrepresented entities or contexts. Addressing these biases is an important step toward ensuring fairness and reliability in real-world applications.

References

AI@Meta. 2024. Llama 3 model card.

Akiko Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing Management*, 39(1):45–65.

Tom Ayoola, Shubhi Tyagi, Joseph Fisher, Christos Christodoulopoulos, and Andrea Pierleoni. 2022. ReFinED: An efficient zero-shot-capable approach to end-to-end entity linking. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 209–220, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Nicholas Botzer, Yifan Ding, and Tim Weninger. 2021. Reddit entity linking dataset. *Information Processing & Management*, 58(3):102479.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Thibault Févry, Nicholas FitzGerald, Livio Baldini Soares, and Tom Kwiatkowski. 2020. Empirical evaluation of pretraining strategies for supervised entity linking. In *Automated Knowledge Base Construction*.

Matthew Francis-Landau, Greg Durrett, and Dan Klein. 2016. Capturing semantic similarity for entity linking with convolutional neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1256–1261, San Diego, California. Association for Computational Linguistics.

Zhaochen Guo and Denilson Barbosa. 2018. Robust named entity disambiguation with random walks. *Semantic Web*, 9(4):459–479.

Bahareh Harandizadeh and Sameer Singh. 2020. Tweeki: Linking named entities on Twitter to a knowledge graph. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 222–231, Online. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 782–792, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Qi Liu, Yongyi He, Tong Xu, Defu Lian, Che Liu, Zhi Zheng, and Enhong Chen. 2024a. Unimel: A unified framework for multimodal entity linking with large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 1909–1919.

Xukai Liu, Ye Liu, Kai Zhang, Kehang Wang, Qi Liu, and Enhong Chen. 2024b. OneNet: A fine-tuning free framework for few-shot entity linking via large language model prompting. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13634–13651, Miami, Florida, USA. Association for Computational Linguistics.

Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*.

Marcel Milich and Alan Akbik. 2023. ZELDA: A comprehensive benchmark for supervised entity disambiguation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2061–2072, Dubrovnik, Croatia. Association for Computational Linguistics.

Khalil Mrini, Shaoliang Nie, Jiatao Gu, Sinong Wang, Maziar Sanjabi, and Hamed Firooz. 2022. Detection, disambiguation, re-ranking: Autoregressive entity linking as a multi-task problem. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1972–1983, Dublin, Ireland. Association for Computational Linguistics.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Riccardo Orlando, Pere-Lluís Huguet Cabot, Edoardo Barba, and Roberto Navigli. 2024. ReLiK: Retrieve and LinK, fast and accurate entity linking and relation extraction on an academic budget. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 14114–14132, Bangkok, Thailand. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Vera Provatorova, Samarth Bhargav, Svitlana Vakulenko, and Evangelos Kanoulas. 2021. Robustness evaluation of entity disambiguation using prior probes: the case of entity overshadowing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10501–10510, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Özge Sevgili, Artem Shelmanov, Mikhail Arkhipov, Alexander Panchenko, and Chris Biemann. 2022. Neural entity linking: A survey of models based on deep learning. *Semantic Web*, 13(3):527–570.

Wei Shen, Yuhan Li, Yinan Liu, Jiawei Han, Jianyong Wang, and Xiaojie Yuan. 2021. Entity linking meets deep learning: Techniques and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 35(3):2556–2578.

Senbao Shi, Zhenran Xu, Baotian Hu, and Min Zhang. 2024. Generative multimodal entity linking. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources* and Evaluation (LREC-COLING 2024)*, pages 7654–7665, Torino, Italia. ELRA and ICCL.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Jesse Vig, Alexander Fabbri, Wojciech Kryscinski, Chien-Sheng Wu, and Wenhao Liu. 2022. Exploring neural models for query-focused summarization. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1455–1468, Seattle, United States. Association for Computational Linguistics.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020a. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. Association for Computational Linguistics.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020b. Scalable zero-shot entity linking with dense entity retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6397–6407, Online. Association for Computational Linguistics.

Zilin Xiao, Ming Gong, Jie Wu, Xingyao Zhang, Linjun Shou, and Daxin Jiang. 2023. Instructed language models with retrievers are powerful entity linkers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2267–2282, Singapore. Association for Computational Linguistics.

Amy Xin, Yunjia Qi, Zijun Yao, Fangwei Zhu, Kaisheng Zeng, Xu Bin, Lei Hou, and Juanzi Li. 2024. Llmael: Large language models are good context augmenters for entity linking. *Preprint*, arXiv:2407.04020.

Ikuya Yamada, Koki Washio, Hiroyuki Shindo, and Yuji Matsumoto. 2022. Global entity disambiguation with BERT. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3264–3271, Seattle, United States. Association for Computational Linguistics.

Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, and Gerhard Weikum. 2011. Aida: an online tool for accurate disambiguation of named entities in text and tables. *Proc. VLDB Endow.*, 4(12):1450–1453.

Fangwei Zhu, Jifan Yu, Hailong Jin, Lei Hou, Juanzi Li, and Zhifang Sui. 2023. Learn to not link: Exploring NIL prediction in entity linking. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10846–10860, Toronto, Canada. Association for Computational Linguistics.

A Prompt Example

A.1 System Role

System role establishes the overarching context, defines the task, and sets the behavior of the LLM. We intentionally keep the system role brief, delegating the task of providing detailed instructions and examples to the user role.

```
You are a helpful AI assistant specializing in Entity Disambiguation. You will be given a mention and some context. Your task is to select the correct entity from the given candidates.
```

A.2 User Role

User role is responsible for delivering the detailed task description and providing an illustrative example to guide the LLM. Our approach focuses on user role to leverage the in-context learning capabilities of the model by suppling contextual information about the input, and it instructs the LLM the specific task while outlining the expected output format.

This role ensures that the LLM is equipped with all necessary information to complete the task while adhering to the specified format. It also emphasizes minimalistic responses to prevent ambiguity in the output or any post processing errors.

```
Entity linking is the process of determining the true identity of an entity mentioned in text by linking it to the correct entry in a knowledge base. Given a piece of input text where the mention is marked as follows: #mention#, your goal is to select the candidate that most accurately represents the given mention based on the contextual information provided in the text.

Here is an example:

Text: #Amazon# is one of the largest e-commerce platforms in the world, founded by Jeff Bezos.

Entity Mention: Amazon

Candidates: [list of entity candidates with descriptions]
```

```
Correct Answer: [correct candidate ID]

Now it is time to perform entity linking with the following inputs:

Text: #mention#...

Entity Mention: [mention]

Candidates: [list of candidates]

Answer only with the candidate number that you think is the correct answer. Answer with 'None of the candidates' if none of the candidates can be selected. Please do not include any additional information or explanation in your answer.
```

A.3 Assistant Role

The assistant role represents the LLM's response to the task outlined by the system and user roles. In this role, the LLM is expected to process the provided text, mention, and candidate entities, and then identify the most appropriate entity from the given list. The assistant's output is intentionally designed to be concise and unambiguous, focusing solely on delivering the correct answer without additional commentary or explanation.

B Implementation Details

B.1 Context and Description Length

The ZELDA Benchmark contains instances with varying context lengths, and entity descriptions also differ in size. Descriptions retrieved via the Wikipedia API exhibit even greater length variation. To standardize context and description lengths, we used the `nltk` library. Specifically, we extract context from the passage by locating the mention and selecting up to two sentences on either side using a sentence tokenizer. For entity descriptions, we enforce brevity by limiting them to a maximum of three sentences.

B.2 Encoder as Retriever

For our encoder model as a retriever, we randomly sample 100,000 instances from the large ZELDA training set (8M datapoints). We split this data 80:20 for training and validation. Training is conducted on an A100 GPU (40GB), taking approximately 2 hours, with testing requiring an additional 30 minutes.

B.3 QLoRA: ED Adaptation

For ED task adaptation, we adopt a data- and compute-efficient approach using the Python `unsloth` library for QLoRA fine-tuning. We randomly sample 4,000 instances from the ZELDA training set, splitting it 90:10 for training and validation. We fine-tune Llama-3.1-8B-Instruct for one epoch on an A100 GPU (40GB), which takes approximately one hour. Testing requires around 2.5 hours due to the large test set of 27,277 instances.

For more details, please refer to the code repository: https://anonymous.4open.science/r/RAG-ED-33B8.