NER Retriever: Zero-Shot Named Entity Retrieval with Type-Aware Embeddings

Or Shachar **Uri Katz** **Yoav Goldberg** **Oren Glickman**
Computer Science Department, Bar-Ilan University
{shachao8, uri.katz, yoav.goldberg, oren.glickman}@biu.ac.il

Abstract

We present **NER Retriever**, a zero-shot retrieval framework for ad-hoc Named Entity Retrieval, a variant of Named Entity Recognition (NER), where the types of interest are not provided in advance, and a user-defined type description is used to retrieve documents mentioning entities of that type. Instead of relying on fixed schemas or fine-tuned models, our method builds on internal representations of large language models (LLMs) to embed both entity mentions and user-provided open-ended type descriptions into a shared semantic space. We show that internal representations, specifically the value vectors from mid-layer transformer blocks, encode fine-grained type information more effectively than commonly used top-layer embeddings. To refine these representations, we train a lightweight contrastive projection network that aligns type-compatible entities while separating unrelated types. The resulting entity embeddings are compact, type-aware, and well-suited for nearest-neighbor search. Evaluated on three benchmarks, NER Retriever significantly outperforms both lexical and dense sentence-level retrieval baselines. Our findings provide empirical support for representation selection within LLMs and demonstrate a practical solution for scalable, schema-free entity retrieval. The NER Retriever Codebase is publicly available at: https://github.com/ShacharOr100/ner_retriever

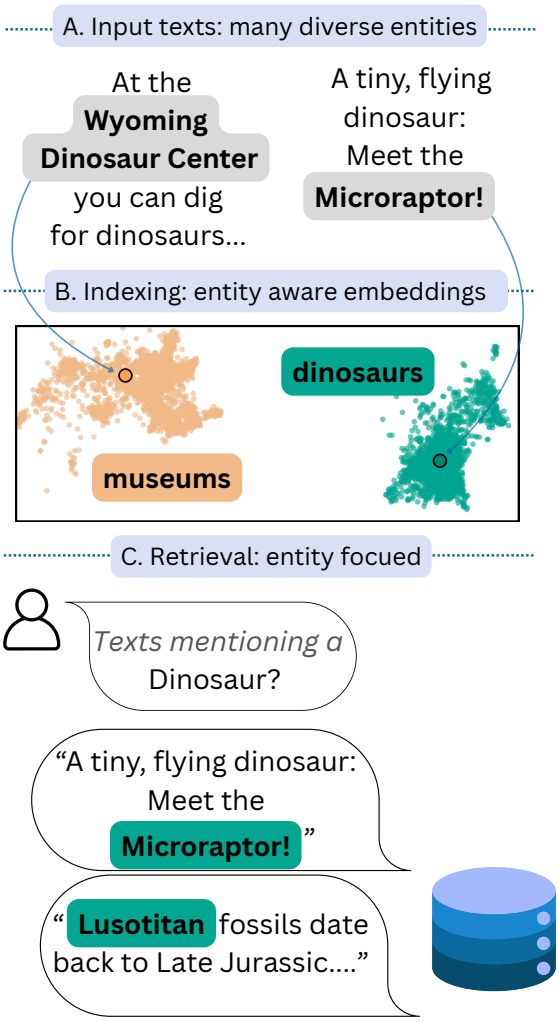

Figure 1: Example use case for ad-hoc Named Entity Retrieval given a query type (“dinosaur”)

1 Introduction

Named Entity Recognition (NER) is a foundational task in natural language processing (NLP), aimed at identifying and classifying entity mentions in text into predefined categories. While traditional NER systems have made substantial progress, they typically rely on extensive labeled data and are constrained to a fixed set of entity types such as *person*, *organization* or *location*. This limits their applicability in real-world settings, where the types of entities of interest can be diverse, domain-specific, and defined in an ad-hoc manner. To address this limitation, fine-grained entity typing (Choi et al., 2018) assigns open-ended fine-grained entity types to entities. Recently, it was proposed to extend named-entity recognition to named entity *retrieval* (Katz et al., 2023). In the retrieval task rather than predicting entity types from a closed schema, the goal is to store documents in a way that enables

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 11175–11186
November 4-9, 2025

retrieving all text segments that mention entities of a type defined at query time (e.g., "retrieve all texts mentioning a *dinosaur* or a *politician*"). This setup reflects practical needs in information retrieval, question answering, and knowledge base construction. However, it poses unique challenges: entity types may be unseen during training, and representations must generalize across fine-grained, open-ended categories.

In this work, we propose **NER Retriever**, a zero-shot retrieval framework for ad-hoc Named Entity Retrieval. Our method leverages the internal structure of large language models (LLMs) to obtain type-aware representations of entity mentions. Specifically, we extract contextual embeddings from an intermediate transformer layer (e.g., layer 17 of LLaMA 3.1 8B) and distill them into a compact, discriminative embedding space using a contrastive learning objective. This allows the system to generalize to novel entity types without requiring task-specific LLM finetuning.

Figure 1 illustrates a typical use case. During offline indexing, all entity mentions in the corpus (e.g., "Microraptor", "Lusotitan") are projected into an embedding space where they cluster near their type description (e.g., "dinosaur") and remain distinct from unrelated concepts (e.g., "The Wyoming Dinosaur Museum"). At query time, user-specified type queries are mapped into the same space and used to retrieve relevant documents via nearest-neighbor search.

We evaluate NER Retriever across three benchmarks: Few-NERD (Ding et al., 2021), MultiCoNER 2 (Fetahu et al., 2023), and NERetrieve (Katz et al., 2023), and find that it outperforms both lexical (BM25) and dense retrieval baselines in most cases. Our results demonstrate that the system enables zero-shot named entity retrieval of highly fine-grained entity types across diverse domains, as represented in the evaluation benchmarks. Our results highlight the value of mid-layer LLM representations and contrastive tuning for zero-shot entity retrieval, and offer insights into the internal structure of type-sensitive knowledge within pretrained language models.

2 Task Formulation and Problem Definition

We follow the task formulation proposed by Katz et al. (2023), which defines *Named Entity Retrieval* as retrieving texts that mention entities belonging to a given entity type. Unlike traditional NER, which detects and classifies entity spans in text, this task frames the entity type itself as an open-ended user-provided query (e.g., "mountain", "politician", "dinosaurs") and aims to return all documents or sentences containing mentions of entities of that type. Formally, given a query q_{type} representing a specific named entity type, and a corpus $\mathcal{C} = \{d_1, d_2, \ldots, d_N\}$ of documents, the task is to return a ranked list $\mathcal{L} = [d_i, d_j, \ldots, d_k]$ of documents from $\mathcal{C}$. Each document in $\mathcal{L}$ must contain at least one named entity mention whose type corresponds to the query q_{type}. The task is evaluated under an *exhaustive* retrieval setting, where systems are expected to return all relevant documents for each query type from a large corpus. The task is designed to support zero-shot retrieval, a setup that reflects real-world scenarios in which users may search for novel or emerging entity categories, and systems must generalize beyond memorized types to retrieve relevant content.

3 Method Overview

We adopt an embedding-based approach in which both entity mentions and type queries are embedded into a shared semantic space, enabling similarity-based search over their vector representations.

A central design choice is *what* to embed and *how* to embed it effectively. While previous approaches have relied on sentence-level representations (Rassin et al., 2024) or prompted classification (Ashok and Lipton, 2023), we argue that directly embedding individual entity mentions offers greater precision and flexibility. This approach allows the system to focus on localized signals, rather than diluting entity-type cues across entire documents. Although embedding every entity increases the number of stored vectors, we offset this by using low-dimensional representations tailored for named entity retrieval.

To obtain high-quality embeddings, we build on the contextual representations generated by LLMs. However, unlike most prior work that uses top-layer outputs, we systematically investigate which internal layer and subcomponent best captures entity-type information. We find that the value (V) vectors from a mid-layer (e.g., block 17 in LLaMA 3.1 8B) outperform other representations in distinguishing fine-grained types.

Using this insight, we construct an end-to-end

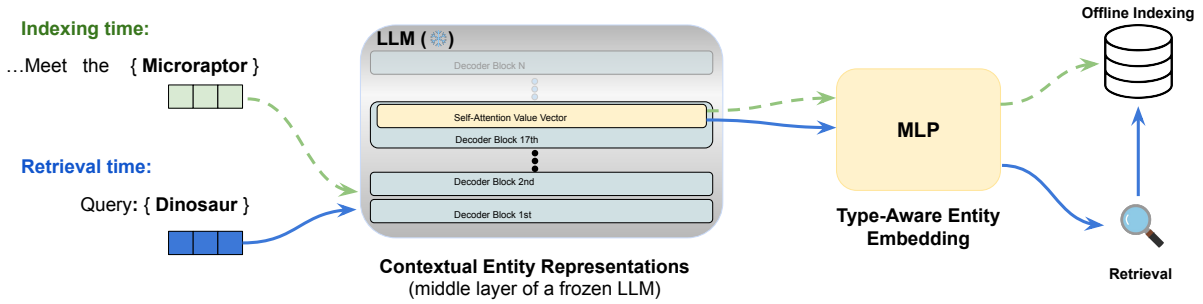

Figure 2: NER Retriever high-level architecture. During indexing (green, dashed), entity spans are embedded and stored using LLM mid-layer contextual representation and a type-aware encoder. At query time (blue, solid), the type description is embedded using the same pipeline and matched to relevant entities via nearest-neighbor search.

system (Figure 2) where both entity mentions and type queries are embedded using the selected LLM representation, followed by a lightweight multi-layer perceptron (MLP) that maps these embeddings into a compact, type-aware vector space. The MLP is trained with a contrastive objective to align entities of the same type and separate those of different types. At indexing time, we compute and store the transformed embeddings for all entity mentions. At query time, we embed the user-specified type and retrieve the most similar entity vectors via nearest-neighbor search.

The next section provides a detailed analysis of LLM layer selection and motivates our choice of representation through a targeted evaluation of type-discriminative power.

4 Contextual Entity Representations

Our approach relies on extracting entity mention embeddings from intermediate layers of a pre-trained LLM. In this section, we investigate which layer and component yield the most type-sensitive representations, that is, those that best distinguish entities based on their semantic class (e.g., *drug* vs. *disease*). While LLMs encode rich contextual information, not all layers capture type semantics equally well. Rather than using final-layer outputs, as is common in sentence embedding models, we empirically search for internal representations that maximize type separability.

Motivation: Embedding choice matters. Prior work has shown that sentence embeddings, even those trained for retrieval, often underperform on entity-type discrimination tasks (Katz et al., 2023). This occurs despite LLMs containing detailed entity knowledge (Heinzerling and Inui, 2021), suggesting that naive pooling or top-layer representations fail to transfer type-specific information. We hypothesize that hidden representations from earlier layers may better isolate entity type information, and that these representations can be exploited for retrieval if appropriately selected.

Evaluation setup. We use the Few-NERD dataset to evaluate type sensitivity across different LLM layers and components. For each of 20 sampled fine-grained entity types, we collect 20 sentences containing an entity mention of that type. We then form balanced pairs: *positive* pairs consist of two mentions of the same type, while *negative* pairs consist of mentions of different types. For each mention, we extract its final-token representation and compute the cosine similarity between mention pairs. We then calculate AUC scores for a binary classification task (same-type or not).

Layer and component sweep. We apply this evaluation procedure to several LLMs, including LLaMA 3.1 8B, T5 11B, Mistral 7B, and Gemma 2 7B. For each model, we extract hidden representations from all transformer blocks and subcomponents (e.g., self-attention keys, values, output projections). In total, we analyze 416 distinct representation sources from LLaMA 3.1 alone. We repeat the full evaluation five times with different random seeds and report average AUC scores.

Findings. Figure 3 shows a heatmap of AUC scores across LLaMA 3.1 layers and subcomponents. We observe substantial variation in performance, with mid-to-late layers outperforming both early and final layers. The highest AUC is achieved by the **value (V)** vectors in the self-attention module of block 17, suggesting that this component encodes particularly strong type-level signals.

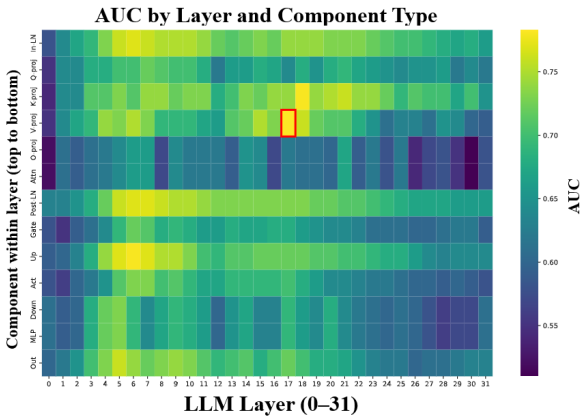

Figure 3: Entity-type discrimination AUC scores for 13 subcomponents across all 32 transformer blocks of LLaMA 3.1 8B.

Table 1 compares top-performing layers across models. LLaMA 3.1 consistently yields better type sensitivity than other architectures, and the performance gap highlights the importance of both model choice and layer selection. Interestingly, the highest overall performance is specifically attributed to representations extracted from the value (V) vectors within the self-attention sub-layer of the 17th transformer block of Llama 3.1 8B, underscoring the significance of the self-attention mechanism in encoding detailed entity-type distinctions. Moreover, it is clear that mid and lower layers encode entity type distinctions much better than top layers.

Model	Block	Layer	Size	AUC
Llama 3.1 8B	17	V	1024	0.78
T5 11B Encoder Part	14	Input Norm	1024	0.74
Gemma 2 7B	6	V	4096	0.56
Mistral 7B v0.3	9	K	1024	0.60

Table 1: Top-performing layer and subcomponent for each tested LLM.

Figure 4 extends this analysis across LLMs by comparing value (V) vectors representations across all transformer blocks. To enable comparison across models with different depths, we normalize the block index from 0 (first block) to 1 (last block). This reveals that the trend persists across models: mid-to-late layers tend to better encode entity-type information, offering practical guidance for representation selection in entity-aware tasks.

Based on these findings, we use the value (V) vectors from block 17 of LLaMA 3.1 8B as the basis for entity embeddings in our retrieval system, described in the next section.

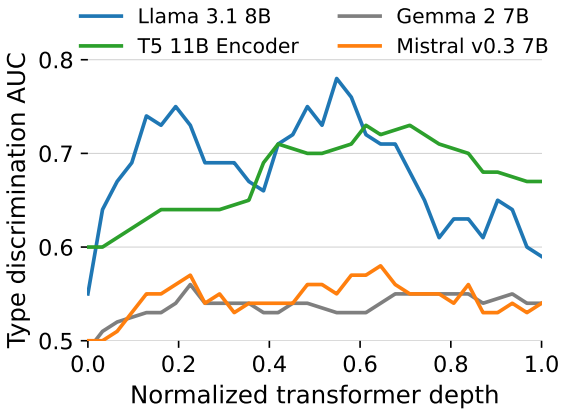

Figure 4: Type discrimination AUC for self-attention value (V) vectors across normalized transformer depth.

5 Type-Aware Entity Embeddings

LLM-derived contextual embeddings are high-dimensional and not inherently optimized for similarity-based named entity retrieval (Reimers and Gurevych, 2019). To address this, we introduce a lightweight, supervised contrastive projection module that transforms entity representations into a compact, type-aware embedding space suitable for nearest-neighbor search (Type-Aware Entity Embeddings block in Figure 2).

We use a two-layer multilayer perceptron (MLP) trained with a triplet contrastive loss (Schroff et al., 2015). Each training triplet includes three components: (1) an *anchor*, which is a natural language type description (e.g., *politician*); (2) a set of *positives*, which are entity mentions of the same type (e.g., *Angela Merkel*, *George W. Bush*); and (3) a set of *negatives*, which are mentions from different types (e.g., *Danube River*, *Mount Everest*).

Positive examples are sampled uniformly from entities sharing the same type, while negatives include both randomly selected and hard-negative entities from different types that exhibit lexical similarity to the anchor (e.g., sharing similar surface forms or contextual phrases). This encourages the model to distinguish semantically relevant types from superficially similar distractors.

The contrastive objective encourages embeddings of same-type entities to be close together and dissimilar types to be well separated. The resulting representations are low-dimensional, discriminative, and aligned with cosine similarity metrics used during retrieval (Weng, 2021).

5.1 Training Data

We train our projection module using the NERetrieve dataset (Katz et al., 2023), which includes 500 fine-grained entity types from diverse domains, each associated with thousands of weakly labeled paragraphs. We use the official training split, containing 400 entity types leaving the test entity types as queries for the end-to-end system evaluation. For each type, we generate 5,000 triplets comprising a query type, a positive paragraph (mentioning a relevant entity), and a negative paragraph (mentioning an unrelated entity). Negative paragraphs are sampled using the hard negative mining procedure described in §5.2. The full training set consists of 2 million triplets.

5.2 Implementation details

We use a frozen LLM (LLaMA 3.1 8B) as the base encoder and extract the final token's value (V) vector from block 17 as the entity representation (see §4). This representation is passed through a two-layer MLP. Input layer's size is derived from the LLM output (in our case 1024). The output layer dimension was set to 500 to reduce the input size while preserving sufficient capacity for capturing task-specific information. The hidden layer was chosen to match this output dimension. We employ the SiLU activation function, which forms the core of SwiGLU used in many modern transformer LLMs, but without its additional parameters. This allows the projection network to learn effectively while remaining lightweight. We apply dropout with a small rate of 0.1 to mitigate overfitting while distilling the entity representations. We adopt Triplet Loss, where the anchor is the embedding of an entity type, positives are embeddings of entity spans of that type within their sentence context, and negatives are embeddings of entity spans belonging to other types. The objective is to align entity types with their instances while separating them from instances of other types.

For hard negative mining, 10% of negative examples in each batch are selected using BM25 similarity to the anchor. This increases training difficulty and encourages robustness to lexical distractors (Karpukhin et al., 2020). All numerical hyperparameters were chosen by independently evaluating a range of values and selecting those yielding the best performance on the entity type discrimination task (Described in Section 4).

Figure 5 illustrates the effectiveness of our learned embedding space. Using UMAP (McInnes et al., 2018), we project type-aware embeddings of entity mentions from the top twenty five Few-NERD fine-grained types into two dimensions. The resulting layout reveals both fine and coarse-grained semantic structure. Entity mentions of the same fine type such as *politician* or *mountain* form dense, coherent clusters, indicating that the model successfully aligns entities of the same category. Moreover, clusters corresponding to different fine types that share a common coarse type such as *geographic entities* (mountain, road and body of water) or *people* (politician, artist and actor) tend to appear adjacent in the space. This suggests that the embedding space not only distinguishes between specific types but also preserves higher-level semantic relationships, supporting both fine-grained retrieval and type generalization.

6 Experimental setting

6.1 NER Retriever: End-to-End System

To evaluate the effectiveness of our type-aware entity embeddings in a retrieval setting, we implemented a complete NER Retriever system consisting of three main stages: **entity detection**, **indexing**, and **retrieval**, following the design described in Section 3 and illustrated in Figure 2.

Entity detection phase. In our approach, it is essential to identify the spans of all entities in a text, independent of their semantic category, so that they can be indexed in the subsequent phase. To this end, we employ an LLM trained on general NER corpora to mark entity spans, following the approach of Awasthy et al. (2020). The identifier operates in a category-agnostic manner, aiming to capture all entities present in the document. None of the benchmarks used in our evaluation were used in training this model.

Indexing phase. For each document, all entity mentions are identified and the document is passed through a frozen LLM encoder (LLaMA 3.1 8B). The contextual representations corresponding to the end span of each entity are extracted from the selected internal component, the value (V) vector from transformer block 17 (see Section 4). These embeddings are then passed through a trained projection module, a lightweight two-layer MLP that maps the high-dimensional LLM embeddings into a compact, type-sensitive space (Section 5). The resulting entity vectors are stored in a vector in-

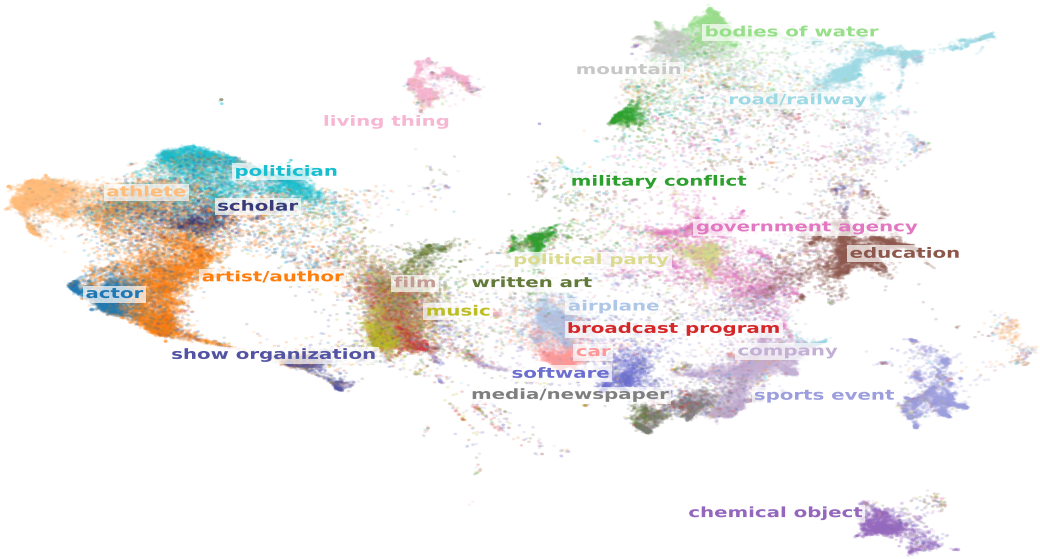

Figure 5: 2D UMAP projection of type-aware entity embeddings produced by our model, visualized for the top 25 Few-NERD types. Entity mentions of the same type cluster tightly, while different types form distinct groups.

dex linked to their corresponding document and optimized for cosine similarity search.

Retrieval phase. At query time, the user provides a type description in natural language (e.g., "dinosaur" or "airline"). This query is embedded using the same LLM and projection pipeline, producing a vector in the same embedding space as the indexed entities. We then retrieve the k most similar entity embeddings based on cosine similarity, and return the corresponding documents as retrieval results. If a document contains multiple entities (and thus multiple embeddings), it is retrieved once as long as at least one of its embeddings meets the similarity criterion.

This setup allows us to evaluate how well our learned embedding space aligns entity mentions with arbitrary type queries in a zero-shot setting. The next subsections describe the datasets, evaluation protocol, and baselines used in our experiments.

6.2 Baselines

We evaluate our proposed NER Retriever framework against a set of strong baselines commonly used in representation-based retrieval tasks.

BM25. A lexical retrieval baseline (Robertson and Zaragoza, 2009), BM25 remains competitive in many entity-centric retrieval settings (Sciavolino et al., 2021). We apply BM25 at the sentence level using the entity type label as a query and retrieve sentences containing matching entities.

E5-Mistral. A decoder-only sentence embedding model that uses last-token pooling to produce fixed-size representations (Wang et al., 2023). E5-Mistral is trained for general-purpose retrieval and serves as a strong dense baseline in both document and entity-level retrieval settings.

NV-Embed v2. A recent decoder-only model with the same backbone as E5-Mistral but using a latent attention-based pooling mechanism to improve sentence-level representation quality (Lee et al., 2024).

Both E5-Mistral and NV-Embed v2 have comparable architecture and parameter count to LLaMA 3.1, providing a fair basis for comparison with our approach.

All models are evaluated in the same zero-shot setting, without task-specific fine-tuning or instructions.

6.3 Evaluation Datasets

We evaluate our system in a zero-shot entity retrieval setting using multiple fine-grained NER benchmarks adapted as retrieval tasks. In each benchmark, every entity type is treated as a distinct query, and the goal is to retrieve all text segments (paragraphs or sentences) containing entity mentions of that type.

We conduct experiments on three widely used

Model	Few-NERD	MultiCoNER 2	NERetrieve Test
BM25	0.22	0.08	0.27
NV-Embed v2	0.04	0.07	**0.29**
E5-Mistral	0.08	0.09	0.22
NER Retriever (Ours)	**0.34**†	**0.32**†	0.28
NER Retriever w/ *entity span oracle*	0.37	0.35	0.34

Table 2: Zero-shot **R-Precision** scores across three datasets. NER Retriever yields up to four times more performance compared to lexical (BM25) and dense (E5-Mistral, NV-Embed v2) baselines. † indicates that the score is significantly higher than the strongest competing baseline (Wilcoxon, $p < 0.05$)

NER benchmarks, repurposed for entity-centric retrieval:

NERetrieve (Katz et al., 2023): The NERetrieve test set is a large, silver-annotated benchmark comprising 100 held-out fine-grained entity types paired with approximately 2.4 M paragraphs. We use the test split described in §5.1, which is disjoint from the contrastive MLP training data. Due to the substantial size of this split, evaluation is performed on a randomly sampled 5% subset, corresponding to about 120 K documents, to ensure computational feasibility while maintaining representativeness.

Few-NERD (supervised) (Ding et al., 2021): A manually annotated dataset covering 66 fine-grained entity types across 188K Wikipedia sentences. For fairness in comparison, we omit instances with fewer than five words, as such very short sentences were found to harm the performance of sentence-encoder baselines while not impacting our method.

MultiCoNER 2 (Fetahu et al., 2023): A silver-annotated dataset containing 12 languages and 33 entity types. We used the English section in evaluation, consisting of ∼268K sentences. Extracted from multiple sources, it emphasizes short texts and low-context scenarios, making it a challenging benchmark for retrieval-based systems.

These benchmarks include long-tail and domain-specific types, making them well-suited to evaluate zero-shot generalization. Crucially, all test-time entity types and associated documents are disjoint from those used during training our system.

6.4 Evaluation Metrics

We report **Average R-Precision** (Manning et al., 2008) to assess retrieval performance. For each query (i.e., entity type), R-Precision is defined as Precision@K, where K is the total number of relevant documents for that query. We compute the R-Precision for each entity type and report the mean across all queries in a dataset. This metric rewards systems that retrieve all relevant mentions while penalizing retrieval of unrelated entities.

7 Results and Analysis

Table 2 reports R-Precision for NER Retriever and baseline models across three benchmarks (Few-NERD, MultiCoNER 2, and NERetrieve). NER Retriever substantially outperforms all baselines on Few-NERD and MultiCoNER 2, while achieving comparable performance on the NERetrieve test set, demonstrating the effectiveness of type-aware embeddings for ad-hoc entity retrieval.

MultiCoNER 2. This dataset presents the most challenging retrieval scenario due to its short and low-context text segments. NER Retriever achieves an R-Precision of 0.32—more than three times higher than E5-Mistral (0.09) and four times higher than BM25 (0.08). These results indicate that sentence-level embedding models and lexical methods struggle in limited-context settings, while our type-aware embeddings remain effective.

Few-NERD. NER Retriever also leads on Few-NERD with an R-Precision of 0.34, substantially outperforming both NV-Embed v2 (0.04) and E5-Mistral (0.08). This confirms our system's ability to generalize across manually annotated, diverse fine-grained entity types.

NERetrieve Test. On the large and diverse NERetrieve Test set, NER Retriever achieves results comparable to NV-Embed v2 (0.29 vs. 0.28, difference not statistically significant) and BM25 (0.28). This is the only dataset where our method does not clearly outperform all baselines. We attribute this to the dataset's descriptive nature, as its Wikipedia-based source often provides explicit cues about entity types. For example, when querying for "French singer," the term singer is fre-

quently stated directly in the text. The fact that BM25 also performs competitively supports this interpretation.

Entity detection. Table 2 highlights the critical role of entity span detection in our framework. When we replace the automatic span detector with an oracle that uses the gold annotations provided by each dataset, NER Retriever's performance improves by roughly 11% on average. This gap illustrates how missed or incorrectly detected spans can directly limit retrieval coverage. Importantly, under the oracle setting, NER Retriever outperforms all baselines, suggesting that our retrieval approach is highly effective when provided with accurate spans. These results underscore that advances in entity detection can further enhance NER Retriever's robustness and overall effectiveness.

We report additional metrics (Precision@50 and Precision@200) in Table 3, which demonstrate NER Retriever's effectiveness even in non-exhaustive retrieval scenarios, maintaining a strong lead in top-k results.

Statistical significance. We tested significance using the Wilcoxon signed-rank test. For each dataset, we compared our method against the strongest alternative (i.e., the highest-scoring competing model) to assess whether any other system significantly outperforms ours. All significance claims are based on a p-value threshold of 0.05.

8 Ablation Studies and Analysis

We conduct a series of ablation experiments to isolate the contribution of key design choices in our pipeline: (i) selection of the hidden layer used for representation extraction; (ii) method for selecting the token representation of an entity; and (iii) use of a task-specific projection MLP. All ablation experiments were evaluated on Few-NERD.

8.1 Hidden Layer Selection

Our approach uses the V-projection output from transformer block 17 of the LLaMA 3.1 8B model to extract contextual representations. Exhaustive evaluation of all possible layers is computationally infeasible. Instead, we adopted a layer estimation method (see Section 4) to identify a layer likely to contain task-relevant information. To validate this choice, we compared retrieval performance when using representations from the selected hidden layer versus the final model output. Results show a significant improvement in average R-Precision from 0.09 (final layer) to 0.19 (block 17), demonstrating the effectiveness of our layer selection strategy.

8.2 Token Representation

We compare two strategies for selecting the token embedding used to represent each entity. The first uses the representation of the end-of-sequence (EOS) token, hypothesizing that it may capture a global summary of the sentence. The second uses the final token in the span corresponding to the entity, motivated by the decoder's autoregressive property, only the final token can attend to all previous tokens in the sequence. As shown in Table 4, span-based representations substantially outperform EOS-based ones (0.19 vs. 0.03 R-Precision), supporting the use of direct entity-span tokens for retrieval tasks.

8.3 MLP Projection

We assess the effect of applying a learned MLP on top of the contextual embedding. The MLP serves to both distill the high-dimensional LLM representation into a lower-dimensional, task-specific embedding space and improve retrieval performance. We also experiment with feeding both the EOS token and the entity token as a dual input to the MLP, but this configuration yields no performance gain over using only the entity token, as shown in Table 4.

8.4 Efficiency and Storage Considerations

NER Retriever introduces a shift in retrieval design by storing one embedding per entity instance rather than per sentence. While this increases the number of stored vectors, each vector is only 500 dimensions, substantially smaller than modern sentence embedders like NV-Embed v2 (4096 dimensions). Moreover, NER Retriever does not generate embeddings for text segments without detected entities, further reducing storage requirements and potentially improving retrieval efficiency compared to sentence-level models that embed every document regardless of content. On MultiCoNER 2, the dense vector index for NV-Embed v2 occupies 9.2 GB, whereas NER Retriever requires only 2 GB, a reduction of nearly 79%.

Model	Few-NERD	MultiCoNER 2	NERetrieve Test
BM25	0.42/0.38	0.29/0.27	0.67/0.52
E5-Mistral	0.33/0.10	0.34/0.27	0.42/0.42
Nvidia NV-Embed	0.03/0.02	0.28/0.18	0.35/0.33
NER Retriever	0.49/0.48	0.39/0.42	0.49/0.41
NER Retriever w/ *entity span oracle*	0.61/0.58	0.50/0.49	0.61/0.52

Table 3: **Precision@50 / Precision@200** for different models across datasets

Model Variant	R-Precision
Full (17V + Entity MLP)	**0.34**
17V + Entity & EOS MLP	0.33
17V + Entity (-No MLP)	0.16
Final Output Layer + Entity (-No MLP)	0.08
17V + EOS (-No MLP)	0.02

Table 4: Ablation results on the Few-NERD dataset, measuring the effect of different components in the NER Retriever architecture.

9 Related Work

Named Entity Recognition (NER) has evolved from identifying coarse-grained types such as PERSON, ORGANIZATION, and LOCATION (Tjong Kim Sang and De Meulder, 2003; Settles, 2004; Sekine, 2008) toward fine-grained and domain-specific categories. Fine-grained entity typing seeks to assign open-ended, context-sensitive type labels, with Choi et al. (2018) introducing ultra-fine typing using natural language phrases.

Large language models (LLMs) have recently been shown to perform zero-shot NER by leveraging their parametric world knowledge (Ashok and Lipton, 2023; Xie et al., 2023). Our approach builds on this intrinsic capability, extracting entity representations from LLMs to enable zero-shot retrieval of named entities.

Named Entity Retrieval, introduced by Katz et al. (2023), frames the task of retrieving all documents containing mentions of entities of a given type. Existing retrieval methods perform poorly on this task, and entity-focused benchmarks such as QUEST (Malaviya et al., 2023), DBpedia Entity v2 List-Search (Hasibi et al., 2017), QAMPARI (Amouyal et al., 2023), D-MERIT (Rassin et al., 2024), and MoNaCo (Wolfson et al., 2025) similarly show that off-the-shelf retrieval achieves limited effectiveness. Prior work has focused mainly on linking entity mentions to knowledge bases (De Cao et al., 2020; Wu et al., 2020) or supervised entity retrieval in QA (Sciavolino et al., 2021; Shavarani and Sarkar, 2025), but none explicitly address ad-hoc retrieval of free-form entity categories in text—a gap our work targets.

As shown in the WideSearch benchmark (Wong et al., 2025), agents are limited in handling broad queries that demand exhaustive coverage; NER Retriever advances this goal by offering fine-grained named entity retrieval that could equip RAG-based agents with the granular evidence needed for reliable broad-query search.

Our method employs contrastive learning, aligning entity-category queries with entity-mention spans. While contrastive learning has been explored for traditional NER (Das et al., 2022; Zhang et al., 2023), these works focus on token classification rather than retrieval-based scenarios.

10 Conclusion

We introduced NER Retriever, a zero-shot framework for ad-hoc named entity retrieval based on type-aware embeddings derived from LLMs. By selecting and refining mid-layer representations with contrastive learning, our method enables accurate and efficient retrieval across diverse, unseen entity types. Our results highlight the effectiveness of representation selection and suggest new directions for embedding-based NER systems.

Limitations

Our system relies on the parametric knowledge encoded in LLMs, which may limit its effectiveness in specialized domains such as law, medicine, or finance. In zero-shot settings, performance in these areas may degrade due to the lack of domain-specific coverage. This limitation could be addressed by integrating domain-adapted LLMs or fine-tuning with targeted data.

References

Samuel Amouyal, Tomer Wolfson, Ohad Rubin, Ori Yoran, Jonathan Herzig, and Jonathan Berant. 2023. QAMPARI: A benchmark for open-domain questions with many answers. In *Proceedings of the Third Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 97–110, Singapore. Association for Computational Linguistics.

Dhananjay Ashok and Zachary C Lipton. 2023. Promptner: Prompting for named entity recognition. *arXiv preprint arXiv:2305.15444*.

Parul Awasthy, Taesun Moon, Jian Ni, and Radu Florian. 2020. Cascaded models for better fine-grained named entity recognition. *CoRR*, abs/2009.07317.

Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. 2018. Ultra-fine entity typing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 87–96, Melbourne, Australia. Association for Computational Linguistics.

Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca Passonneau, and Rui Zhang. 2022. CONTaiNER: Few-shot named entity recognition via contrastive learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6338–6353, Dublin, Ireland. Association for Computational Linguistics.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*.

Ning Ding, Guangwei Xu, Yulin Chen, Xiaobin Wang, Xu Han, Pengjun Xie, Haitao Zheng, and Zhiyuan Liu. 2021. Few-NERD: A few-shot named entity recognition dataset. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3198–3213, Online. Association for Computational Linguistics.

Besnik Fetahu, Zhiyu Chen, Sudipta Kar, Oleg Rokhlenko, and Shervin Malmasi. 2023. MultiCoNER v2: a large multilingual dataset for fine-grained and noisy named entity recognition. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2027–2051, Singapore. Association for Computational Linguistics.

Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. Dbpedia-entity v2: A test collection for entity search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1265–1268. ACM.

Benjamin Heinzerling and Kentaro Inui. 2021. Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1772–1791, Online. 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.

Uri Katz, Matan Vetzler, Amir Cohen, and Yoav Goldberg. 2023. NERetrieve: Dataset for next generation named entity recognition and retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3340–3354, Singapore. Association for Computational Linguistics.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Chaitanya Malaviya, Peter Shaw, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2023. QUEST: A retrieval dataset of entity-seeking queries with implicit set operations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14032–14047, Toronto, Canada. Association for Computational Linguistics.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. 2018. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861.

Royi Rassin, Yaron Fairstein, Oren Kalinsky, Guy Kushilevitz, Nachshon Cohen, Alexander Libov, and Yoav Goldberg. 2024. Evaluating D-MERIT of partial-annotation on information retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2913–2932, Miami, Florida, USA. 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.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Christopher Sciavolino, Zexuan Zhong, Jinhyuk Lee, and Danqi Chen. 2021. Simple entity-centric questions challenge dense retrievers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6138–6148.

Satoshi Sekine. 2008. Extended named entity ontology with attribute information. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco. European Language Resources Association (ELRA).

Burr Settles. 2004. Biomedical named entity recognition using conditional random fields and rich feature sets. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications (NLPBA/BioNLP)*, pages 107–110.

Hassan Shavarani and Anoop Sarkar. 2025. Entity retrieval for answering entity-centric questions. In *Proceedings of the 4th International Workshop on Knowledge-Augmented Methods for Natural Language Processing*, pages 1–17, Albuquerque, New Mexico, USA. Association for Computational Linguistics.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Lilian Weng. 2021. Contrastive representation learning. *lilianweng.github.io*.

Tomer Wolfson, Harsh Trivedi, Mor Geva, Yoav Goldberg, Dan Roth, Tushar Khot, Ashish Sabharwal, and Reut Tsarfaty. 2025. Monaco: More natural and complex questions for reasoning across dozens of documents. *Transactions of the Association for Computational Linguistics*.

Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, and 1 others. 2025. Widesearch: Benchmarking agentic broad info-seeking. *arXiv preprint arXiv:2508.07999*.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. 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.

Tingyu Xie, Qi Li, Jian Zhang, Yan Zhang, Zuozhu Liu, and Hongwei Wang. 2023. Empirical study of zero-shot NER with ChatGPT. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7935–7956, Singapore. Association for Computational Linguistics.

Sheng Zhang, Hao Cheng, Jianfeng Gao, and Hoifung Poon. 2023. Optimizing bi-encoder for named entity recognition via contrastive learning. In *The Eleventh International Conference on Learning Representations*.

A System Implementation

A.1 Entity Detection

In order to identify entities in text, we utilized the extractor component of CascadeNER (Awasthy et al., 2020)[1], a Qwen 2.5 model fine-tuned for entity span detection. The model was trained on the DynamicNER dataset, which provides multilingual and fine-grained entity annotations; importantly, none of the evaluation datasets used in this work were included in its fine-tuning. For each input text, the extractor outputs a marked version in which entities are encompassed by ## symbols. For example:

`model output: "##Claremore Lake## is a reservoir in ##Rogers County##"`

We evaluate the entity detection phase using the original span annotations provided in each dataset. A match between two spans is considered valid if their character-level Jaccard similarity exceeds 0.8. We report the coverage of annotated entities for each evaluation dataset, while noting that the extractor also identifies many additional entities that fall outside the original dataset's annotation scope. As a result, the extractor enables a more exhaustive identification process than what is captured by the evaluation datasets. Overall, the results are very strong, though there remains room for improvement.

Table 5: Entity detection coverage using the entity extractor model

Dataset	Coverage
NERetrieve	0.89
MultiCoNER 2	0.90
Few-NERD	0.94

[1] https://huggingface.co/CascadeNER/models_for_CascadeNER