ENTITY-LINKINGS: A Unified Library for Entity Linking

Yuya Sawada, Tsuyoshi Fujita, Yusuke Sakai, Taro Watanabe
Nara Institute of Science and Technology (NAIST)
{sawada.yuya.sr7, sakai.yusuke.sr9, taro}@is.naist.jp
fujita.tsuyoshi.fy4@naist.ac.jp

Abstract

Entity linking (EL) aims to disambiguate named entities in text by mapping them to the appropriate entities in a knowledge base. However, it is difficult to use some EL methods, as they sometimes have issues in reproducibility due to limited maintenance or the lack of official resources. To address this, we introduce ENTITY-LINKINGS, a unified library for using and developing entity linking systems through a unified interface. Our library flexibly integrates various candidate retrievers and re-ranking models, making it easy to compare and use any entity linking methods within a unified framework. In addition, it is designed with a strong emphasis on API usability, making it highly extensible, and it supports both command-line tools and APIs. Our code is available on GitHub[1] and is also distributed via PyPI[2] under the MIT-license. The video is available on YouTube[3].

1 Introduction

Entity linking (EL) is the task of mapping named entities in text to canonical entries in a knowledge base (KB). As shown in Figure 1, since Japan has had many emperors throughout its history, named entities, i.e., *emperor*, are inherently ambiguous. Therefore, it is necessary to identify the correct entity and link it to a canonical entry in the KBs. Most named entities are context-dependent and must be disambiguated by linking them to normalized identifiers in a KB for identification.

A typical EL pipeline consists of two steps: it first detects entity mentions in text (**Mention Detection; MD**), and then links each mention to a unique identifier in the KBs (**Entity Disambiguation; ED**). EL has broad applicability in automating the creation of hyperlinks for domain-specific

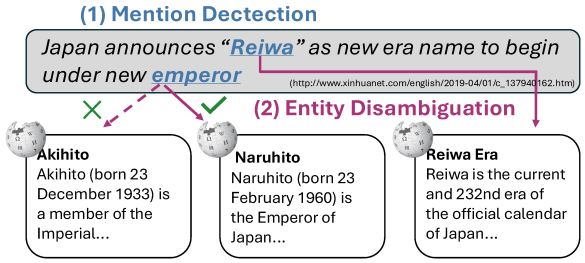

Figure 1: Overview of the entity linking task.

terms in resources such as Wikipedia, FAQs, and product manuals. In addition, it has been widely applied to a broad range of interdisciplinary and practical domains, including clinical records, financial documents, legal statutes, and contracts.

While MD can leverage ready-made named entity recognition (NER) tools such as spaCy (Honnibal et al., 2020) and Flair (Akbik et al., 2018), ED requires a dedicated implementation that involves constructing and maintaining the task-specific KBs and retrieval components. As a result, each method is implemented individually, which poses challenges for comparison under unified conditions. In fact, even when using a common knowledge base such as a Wikidata dump (Vrandečić and Krötzsch, 2014), many subtle but critical differences exist across methods, including the dump timing, candidate pruning strategies, and data formats, all of which hinder fair and consistent evaluation. Moreover, several official implementations are either unavailable, e.g., Févry et al. (2020); Wang et al. (2024a) or no longer maintained, e.g., Wu et al. (2020), leading to serious reproducibility issues[4]. Although some EL evaluation benchmarks (Milich and Akbik, 2023; Röder et al., 2018) have been proposed, many studies still report baseline perfor-

[1] : naist-nlp/entity-linkings
[2] : pip install entity-linkings
[3] : https://youtu.be/xFx05wBoz5E

[4] For example, BLINK's implementation on Github is reported an open issue that hard negatives sampling is not implemented: github.com/facebookresearch/BLINK/issues/31. Due to this issue, Rücker and Akbik (2025) used a BLINK model trained only on in-batch samples as a baseline.

mance by citing the originally reported scores from previous work. This practice increasingly undermines the scientific reproducibility of EL research. Therefore, it is highly desirable to establish a unified framework for utilizing and evaluating entity linking methods.

We introduce ENTITY-LINKINGS, a unified library that supports multiple modern EL systems and datasets. In particular, we primarily focus on ED, which has become the central component of modern EL methods. It facilitates easy reproduction, simplifies the implementation of custom models, and consolidates experimental setups. We carefully decompose EL systems into three modular components into a unified pipeline: *Mention Detector*, *Candidate Generator*, and *Candidate Reranker*, as illustrated in Figure 2. This design enables flexible combinations, thereby improving extensibility, maintainability, and transparency. ENTITY-LINKINGS supports comprehensive experimentation, provides well-documented interfaces, and supports both CLI- and Python API-based access. We hope that it will further accelerate EL research.

2 Background

2.1 Preliminaries on Entity Linking

Entity linking (EL) is the task of associating mentions in natural language text with entries in a knowledge base (KB). Formally, we represent the input text x as a sequence of tokens $x = x_1, x_2, \ldots, x_n$, where each x_i denotes the i-th token in the text after tokenization. A mention span m is defined as a contiguous subsequence of tokens $m = x_i \ldots x_j$, where $1 \leq i \leq j \leq n$, meaning that the span starts at token position i and ends at token position j. EL aims to produce an entry e in the KB $\mathcal{E}$ ($e \in \mathcal{E}$) corresponding to each mention m in x.

Broadly, existing EL methods decompose the task into two subtasks: *mention detection* (MD) and *entity disambiguation* (ED), as shown in Figure 2. In pipeline systems, mention spans in the input text are first identified in the MD stage and then linked to KB entries in the ED stage. In the MD stage, previous work[5] often employs off-the-shelf NER modules such as spaCy (Honnibal et al., 2020). In the ED stage, the identified mentions are

[5]Although a few approaches, namely end-to-end EL systems, jointly train the MD module with the linking component, in practice, the main difference is whether the MD module is trained or not. Hence, we adopt a simplified description here.

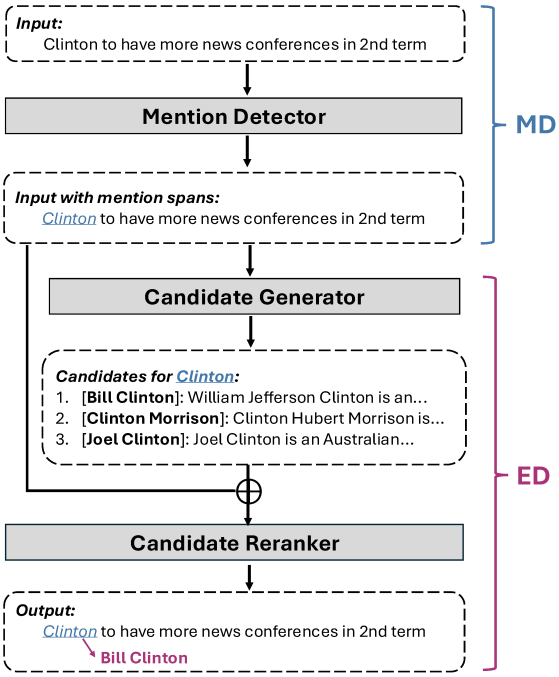

Figure 2: Main components of EL systems and their data processing flow.

linked to their corresponding entities. Specifically, EL systems calculate the probability $p(e|m)$, i.e., an entry e is the target for a given mention m, and output the entry with the highest probability as the target. Nevertheless, for large-scale KBs where $|\mathcal{E}|$ is large, exhaustive computation of $p(e|m)$ for all entries is expensive and complicates the identification of correct entries. To improve the accuracy and efficiency of the system, most ED studies introduce two submodules: *candidate generator* and *candidate reranker*. The candidate generator retrieves a small set $\hat{\mathcal{E}}$ of candidate entries for each mention from the KB $\mathcal{E}$, and the candidate reranker scores these candidates and selects the most plausible one.

For evaluation, the *InKB* micro-F1 score (Röder et al., 2018) is typically used as the standard metric for the EL system. This metric considers a prediction correct if both the extracted mention span and the target entry match the ground truth, and calculates F1 scores as the harmonic mean of precision and recall. On the other hand, since most ED studies focus on specific subtasks of candidate selection or reranking, metrics that assume oracle mention spans are also employed. Specifically, Recall@k is used for candidate selection, where a prediction is correct if the gold entry is included in the top-k predicted candidates. For reranking, Top-1 accuracy is used to measure whether the predicted entity from the candidates matches the gold entry.

2.2 Main Components of EL Systems

We carefully decompose a wide range of EL systems into three components: *Mention Detector*, *Candidate Generator* and *Candidate Reranker*, thereby enabling the unified description and implementation within a standardized framework.

Mention Detector The mention detector is typically used to extract mentions corresponding to specific entries in a KB and is executed before the ED step. Since wrong mention detection leads to error propagation within the pipeline system, recent studies have proposed methods that perform MD after the ED stage (Wang et al., 2024a; Zhang et al., 2022) or specifically enhance the mention detector for the ED step (Tedeschi et al., 2021). Recent studies on end-to-end EL have also introduced unified models that handle both MD and ED through multi-task learning (Ayoola et al., 2022), and that simultaneously execute MD and ED using constrained decoding (Cao et al., 2021). Nevertheless, even in end-to-end EL, the process can still be conceptually decomposed into MD and ED stages, and most systems explicitly maintain a mention detection step in practice. Furthermore, most EL studies focus on enhancing ED, since the MD component can rely on NER tools such as spaCy.

Candidate Generator It retrieves candidate entries from the KBs by taking as input the text with identified mention spans. Existing retrievers can be broadly categorized into textual-frequency-based methods and vector-similarity-based methods. Frequency-based retrievers rely on lexical overlap between the input text and KB entries, and hyperlink counts in Wikipedia. Notable examples include **BM25** (Lù, 2024) and entity-mention prior probabilities such as the Zelda Candidate List (Milich and Akbik, 2023). Similarity-based retrievers embed mentions and entities into a shared vector space and perform nearest-neighbor search to retrieve candidate entries. While **Dual-Encoder** is widely used as a similarity-based retriever (Wu et al., 2020; Gillick et al., 2019), recent studies also employ **Text Embedding Models** (Wang et al., 2024b). Formally, similarity-based retrievers compute the relevance between a mention-context representation and a candidate entity representation using the inner product of their vector embeddings:

$$\text{similarity}(m, e) = \boldsymbol{h}_m^\top \boldsymbol{h}_e, \tag{1}$$

where $\boldsymbol{h}_m$ and $\boldsymbol{h}_e$ denote the vector representations of the mention m and the candidate KB entry $e \in \mathcal{E}$, respectively. Each vector is obtained by encoding the corresponding token sequence:

$$\boldsymbol{h}_m = \text{red}(E_1(\tau_m)), \tag{2}$$

$$\boldsymbol{h}_e = \text{red}(E_2(\tau_e)), \tag{3}$$

where τ_m and τ_e are input representations of mention and entry, respectively. E_1 and E_2 are encoders, with text embedding models typically using a single shared encoder, i.e., $E_1 = E_2$, and dual-encoder employing two separate encoders. $\text{red}(\cdot)$ is a function that reduces the encoder output into a fixed-size vector by applying average pooling over the final-layer token embeddings in the case of text embedding models or by taking the final-layer `[CLS]` token representation in dual-encoder models. We follow the experiments of Wu et al. (2020) in the construction of τ_m and τ_e.

Candidate Reranker The reranker is used in ED to refine the candidate set returned by the candidate generator and select the most appropriate entity for each mention. Rerankers using encoder-only models are widely investigated. Encoder-only rerankers take the mention context and candidate entity descriptions as input and compute a relevance score using cross-encoders (Logeswaran et al., 2019), enabling the model to capture interactions between the mention context and the candidates, e.g., **BLINK** (Wu et al., 2020)[6], **FEVRY** (Févry et al., 2020), and **ExtEnD** (Barba et al., 2022). More recently, rerankers based on encoder-decoder or decoder-only models have also been explored. **FusionED** (Wang et al., 2024a) employs an encoder-decoder model to jointly encode the mention context together with all candidate entities and then decode over the fused representations to select the correct entity. **ChatEL** (Ding et al., 2024) prompts decoder-only large language models (LLMs) with the mention context and the candidates to perform reranking in a generative manner. **ReFinED** (Ayoola et al., 2022) formalize the end-to-end EL system, but they use the candidate lists to reduce training cost, and they can execute ED mode by inputting mentions.

2.3 Dataset

Since EL systems are often expected to generalize across domains sharing the same KBs, recent studies mostly evaluate a single trained model across

[6]BLINK comprises a dual-encoder and a cross-encoder, where the cross-encoder corresponds to the candidate reranker.

multiple evaluation datasets from diverse sources. For example, **GERBIL** (Röder et al., 2018) is a widely used EL benchmark that uses **AIDA-CoNLL** (Hoffart et al., 2011) dataset for training, and evaluates models on the AIDA-CoNLL test set, as well as eight out-of-domain datasets: **MSNBC** (Cucerzan, 2007), **AQUAINT** (Milne and Witten, 2008), **KORE50** (Hoffart et al., 2012), **N3-Reuters-128**, **N3-RSS-500** (Röder et al., 2014), **Derczynski** (Derczynski et al., 2015), **OKE-2015** (Nuzzolese et al., 2015), and **OKE-2016** (Nuzzolese et al., 2016).

Furthermore, recent mainstream evaluations of EL systems focus on improving individual components, and component-specific evaluation datasets are typically used. For candidate generators, **WikilinksNED Unseen Mentions** (Eshel et al., 2017; Onoe and Durrett, 2020), which comprises diverse ambiguous entities found in web-crawled text, and **ZeshEL** (Logeswaran et al., 2019), which features unique entities associated with specific domains such as fictional books and film series from Wikia[7], are often used. Both datasets provide a challenging evaluation setting by ensuring that all mention-entity pairs in the test set are unseen during training. For candidate rerankers, **ZELDA** (Milich and Akbik, 2023) is often used as a comprehensive ED benchmark. ZELDA contains a training dataset comprising 95k documents and a fixed entity vocabulary comprising 82.2k entries derived from the Kensho Derived Wikimedia Dataset[8]. ZELDA also provides aggregated mention link counts from Wikipedia, Wikidata, and Wikilinks to construct candidate lists. Furthermore, the evaluation employs a total of nine datasets: **TWEEKI** (Harandizadeh and Singh, 2020), **AIDA-B** (Hoffart et al., 2011), **REDDIT-POSTS**, **REDDIT-COMMENTS** (Botzer et al., 2021), **WNED-WIKI**, **WNED-CWEB** (Guo and Barbosa, 2018), and **SHADOWLINK-{TOP, SHADOW, TAIL}** (Provatorova et al., 2021).

For KBs, prior work often used independently acquired Wikipedia dumps because there was no standardized KB version. In contrast, recent studies often employ the 5.9M Wikipedia pages provided by the **KILT** benchmark as the target KB (Petroni et al., 2021; Zhang et al., 2022; Wang et al., 2024a). Although WikilinksNED Unseen Mentions does not specify the version of Wikipedia, KILT (Petroni et al., 2021) can be used as the target KB. For ZeshEL, it uses the 49.2k entries derived from Wikia dumps.

[7] https://www.fandom.com/

[8] http://datasets.kensho.com/datasets/wikimedia

3 Problems of Existing EL Systems

Inconsistent Implementations Although many EL systems have been proposed, their implementations are typically designed with system-specific interfaces and data flows, resulting in limited compatibility and reusability. In particular, as described in Section 2.2, although EL systems can be decomposed into three unified pipeline modules and each component can, in principle, be reused modularly, the lack of standardized implementations makes such reuse difficult in practice. As a result, reproducing prior work or replacing individual components, e.g., using a different candidate generator, becomes challenging, hindering unified evaluation.

Transparency and Fair Comparison In particular, previous studies often employ different candidate generators, such as frequency-based or similarity-based retrievers, which makes it challenging to conduct fair comparisons of other modules, e.g., candidate rerankers, across studies. Furthermore, although many evaluation datasets have been proposed and are publicly available, their unified use is hindered by differences in data formats, preprocessing pipelines, and even the snapshots of the knowledge bases. Additionally, methods that depend on such dataset-specific precompiled candidate sets, e.g., PPRforNED (Pershina et al., 2015), cannot be directly applied to other datasets without equivalent precompiled resources, thereby preventing comprehensive evaluation across diverse benchmarks. Moreover, most studies heavily rely on reported scores without re-implementation under consistent settings, which hinders the disentanglement of individual contributions and fair comparisons.

4 Our Library: ENTITY-LINKINGS

ENTITY-LINKINGS supports recent EL systems through a unified interface. As described in Section 2.2, we decompose an EL system into three main components: *Mention Detector*, *Candidate Generator* and *Candidate Reranker*, thereby achieving clear modularity, high extensibility, and ease of maintenance. Moreover, we provide multiple evaluation datasets in a consistent format to enhance usability. We also offer basic pretrained models and precompiled KBs, enabling the comparison of

custom models and datasets under a unified experimental environment with minimal effort.

4.1 Supported Methods and Datasets

We continuously update the supported systems and datasets. Please refer to our GitHub repository[9] for details on the latest supported methods.

Methods Currently, ENTITY-LINKINGS covers the major EL systems introduced in Section 2.2. Specifically, for the ED stage components, ENTITY-LINKINGS supports a wide range of representative systems. For candidate generators, it includes **BM25**, **Dual-Encoder**, and **Text Embedding Models**. For rerankers, it supports **BLINK**, **FEVRY**, **ExtEnD**, **FusionED**, and **ChatEL**.

Datasets ENTITY-LINKINGS can download and use the standard EL datasets introduced in Section 2.3, such as those included in **ZELDA** (Milich and Akbik, 2023) and **GERBIL** (Röder et al., 2018), directly through our HuggingFace Collection[10]. In addition, the library accepts arbitrary datasets in a simple unified JSONL format. A complete list of supported datasets, their licenses, and detailed information is provided in Appendix A.

4.2 Interfaces

ENTITY-LINKINGS has two interfaces: Python application programming interface (API) and command-line interface (CLI). Listing 1 illustrates an example workflow via Python API for training, evaluation, and prediction using the ZELDA dataset, a Dual-Encoder candidate generator, and the BLINK reranker. The KB, candidate generator, and reranker are instantiated by specifying their identifiers in the `load_dictionary()`, `get_retrievers()`, and `get_rerankers()` functions. Once these components are loaded, training and evaluation proceed through the unified `train()` and `evaluate()` methods. A trained model can be applied to any input text using the `predict()` method. Replacing the dataset, dictionary, or any EL component requires only changing the corresponding identifier passed to the API, without modifying the rest of the workflow. In addition, predefined datasets and KBs, as well as user-provided datasets and KBs, can be loaded by supplying their file paths to `load_dataset()`[11] and `load_dictionary()` in lines 5–9. This design allows ENTITY-LINKINGS to support both standard benchmarks and custom EL datasets uniformly.

```python
from datasets import load_dataset
from entity_linkings import get_retrievers, get_rerankers, ELPipeline, load_dictionary

# Existing Corpus
dataset = load_dataset('naist-nlp/zelda')
dictionary = load_dictionary('zelda')
# Custom Corpus
# dataset = load_dataset('json', data_files = {train': 'train.jsonl', 'validation': 'valid.jsonl', 'test': 'test.jsonl'})
# dictionary = load_dictionary('dict.jsonl')

# Model loading
retriever_cls=get_retrievers('dualencoder')
retriever=retriever_cls(dictionary, config=retriever_cls.Config())
reranker_cls=get_rerankers('crossencoder')
reranker=model_cls(retriever, config=model_cls.Config())

# Training
result = reranker.train(dataset['train'], dataset['validation'])

# Evaluation
metrics = retriever.evaluate(dataset['test'])
# Output (metrics): {'R@1': , 'R@10': , 'R@50':, 'R@100':, 'MRR':}
metrics = reranker.evaluate(dataset['test'])
# Output (metrics): {'Acc': }

# Prediction
sentence = 'Toyota makes cars.'
spans=[(0,6)]
predictions = reranker.predict(sentence, spans)
# Output: [{'start': 0, 'end': 6, 'id':'30984'}]
```

Listing 1: An implementation of training and evaluation using BLINK as a model and ZELDA as a dataset.

Extensibility All components are implemented by inheriting from abstract base classes. For instance, retrievers or rerankers are defined via `RetrieverBase` or `RerankerBase` abstract class, respectively. These base classes specify the minimum set of required methods that must be overridden in the derived classes. The components can be instantiated through unified factory functions such as `get_retrievers()` and `get_rerankers()`, which guarantee a consistent interface across different user implementations. In addition, we provide comprehensive test code for all components with CI/CD support, along with well-written documentation. This modular design

[9] https://github.com/naist-nlp/entity-linkings

[10] https://hf.co/collections/naist-nlp/entity-linkings

[11] https://hf.co/docs/datasets/en/loading

Model	R@1	R@10	R@50	R@100
BM25	0.207	0.440	0.556	0.598
Dual-Encoder	0.361	0.594	0.693	0.735
Text Embedding	0.429	0.796	0.796	0.834

Table 1: Candidate generation results in ZeshEL.

enables straightforward extensibility with minimal implementation effort.

Reproducibility We also support configuration via YAML input and export all settings to YAML files, enabling users to share the exact experimental configurations and ensuring high reproducibility as well as deterministic results across runs. Furthermore, we provide basic pretrained weights and precompiled KBs, allowing us to easily reproduce results and conduct fair comparisons.

5 Experiments

In this paper, we follow a standard EL evaluation protocol for simplicity and to ensure a controlled experiment. Specifically, we fix *Mention Detector* to spaCy and focus our evaluation on the ED stage. We compare the performance of two EL components, i.e., *Candidate Generator* and *Candidate Reranker*. We then construct multiple pipeline systems by combining each component and compare their performance across these full pipeline systems. Appendix B describes the detailed settings.

5.1 Candidate Generator

5.1.1 Setup

We evaluate three text retrievers: **BM25**, **Dual-Encoder** and **Text Embedding Model**. For similarity-based retrievers, we use `bert-base-uncased` (110M×2) and `e5-base` (110M) to ensure that the same core Transformer architecture is shared. Following Wu et al. (2020), we train these encoders using in-batch random negatives and top-10 hard negatives. We employ R@k as our evaluation metric, where k ranges from 1 to 100. We use the **ZeshEL** benchmark as introduced in Section 2.3.

5.1.2 Results

Table 1 shows the results. As the value of k increases, the R@k scores for all three retrievers consistently increase, and similarity-based retrievers outperform the frequency-based retriever by effectively leveraging contextual information and entity descriptions. Hence, we confirm that the retrievers are functioning as we expected.

5.2 Candidate Reranker

5.2.1 Setup

We evaluate the eight methods: **Most Frequent Sense** (**MFS**), **Dual-Encoder**, **Text Embedding Model**, **FEVRY**, **ExtEnD**, **ChatEL**, and **FusionED** on the ZELDA benchmark. MFS refers to the entity that is most frequently linked from each mention in the Kensho Wikimedia dataset, Wikilinks web corpus (Singh et al., 2012), and Wikidata, which is a general baseline for ZELDA. For each mention, we select the top 30 most frequent entries from the candidate list. During training, if the ground-truth entity is not among the top 30, we replace the 30th entry with it. For candidate lists with fewer than 30 entries, we fill the remaining slots by randomly sampling from the entire KB.

We use `bert-base-uncased` (110M), `longformer-base` (149M), and `flan-t5-small` (77M) to ensure that the total number of parameters is comparable across models.

5.2.2 Results

Table 2 shows the accuracy for each evaluation split. Although all methods were trained using a small-scale encoder with only the top 30 candidate entries, they consistently outperform the minimal baseline MFS. Furthermore, the results are comparable to those reported in previous work, which confirms each method is working correctly as intended. For the candidate rerankers, almost all the rerankers underperformed compared to the retriever. While rerankers are restricted to selecting from a fixed candidate list, the retriever is trained using hard negatives. This training regimen likely allows the retriever to leverage contextual cues more effectively. This observation is consistent with the findings reported by Rücker and Akbik (2025).

5.3 EL System Evaluation

5.3.1 Setup

We use the GERBIL benchmark, which is widely used for evaluating EL systems. We employ a Dual-Encoder model as the candidate generator and FEVRY, Cross-Encoder, ExtEnD, and ChatEL as the reranker, which achieves good performance in our ED evaluation. In addition, we report reproduction results obtained using the official implementation and pretrained weights of ReFinED (Ayoola et al., 2022) as a system baseline. For the evaluation metric, we employ *InKB* micro-F1 score. A predicted mention is regarded as correct only if

	backbone	AIDA-B	TWEEKI	REDDIT-POSTS	REDDIT-COMM	WNED-CWEB	WNED-WIKI	SLINKS-TAIL	SLINKS-SHADOW	SLINKS-TOP
MFS	–	0.629	0.700	0.825	0.794	0.605	0.648	0.991	0.146	0.402
Text Embedding	$E5_{Base}$	0.837	0.809	0.905	0.915	0.718	0.900	0.991	0.675	0.694
Dual-Encoder	$BERT_{Base}$	0.821	0.779	0.912	0.876	0.703	0.899	0.988	0.637	0.658
FEVRY	$BERT_{Base}$	0.762	0.748	0.888	0.842	0.697	0.849	0.805	0.322	0.439
Cross-Encoder	$BERT_{Base}$	0.793	0.792	0.918	0.911	0.722	0.857	0.996	0.442	0.591
ExtEnD	$Longformer_{Base}$	0.800	0.807	0.922	0.920	0.713	0.874	0.994	0.379	0.534
FusionED	$FlanT5_{small}$	0.638	0.657	0.801	0.760	0.614	0.761	0.989	0.351	0.468
ChatEL	$GPT\text{-}4o_{mini}$	0.756	0.758	0.851	0.814	0.686	0.716	0.980	0.380	0.730

Table 2: Entity Disambiguation results in ZELDA benchmark using our library.

	MSNBC	ACE2004	Derczynski	KORE50	R128	R500	OKE15	OKE16
Dual-Encoder	0.437	0.133	0.304	0.379	0.325	0.256	0.379	0.349
+ FEVRY	0.146	0.100	0.076	0.069	0.193	0.071	0.118	0.127
+ Cross-Encoder	0.474	0.156	0.324	0.336	0.354	0.307	0.416	0.375
+ ExtEnD	0.472	0.156	0.336	0.400	0.345	0.300	0.378	0.351
+ ChatEL	0.407	0.118	0.281	0.543	0.287	0.243	0.349	0.330
ReFinED*	0.561	0.197	0.465	0.55.7	0.474	0.348	0.599	0.573

Table 3: End-to-End Entity Linking results in GERBIL. * means the model that trains using external resources.

both the mention span and the target entry match the ground truth for entities present in the KB.

5.3.2 Results

As shown in Table 3, our straightforwardly constructed system achieves moderate performance compared with existing EL systems, indicating that our library can build competitive EL systems with minimal implementation effort. Among the rerankers, the Cross-Encoder and ExtEnd consistently improved the performance from the results by the Dual-Encoder, whereas FEVRY exhibited a significant decline in performance. The performance gap likely stems from the fact that FEVRY directly projects the representation of the span by concatenating the representation at the span start and end into the entity embedding space without referring to entity titles or descriptions. Unlike competing models, FEVRY fails to leverage explicit semantic information, such as entity titles or descriptions, which limits its ability to resolve entities in out-of-domain datasets.

6 Conclusion

We proposed ENTITY-LINKINGS, a unified library for EL systems based on a standardized three-component interface. In our experiments, we conducted replication studies using standardized EL benchmarks, such as ZELDA and GERBIL. We confirmed that the performance of our three candidate retrievers and five candidate rerankers is consistent with results reported in previous works. Moving forward, to consider the high rate of errors in mention detection by spaCy, we plan to explore the integration of End-to-End EL architectures (Cao et al., 2021; Shavarani and Sarkar, 2023; Ayoola et al., 2022) and Retriever-to-Reader models (Zhang et al., 2022). We will continue to actively maintain and expand it, and we hope that it will further advance both EL research and the community.

Ethics and Broader Impact Statement

Using ENTITY-LINKINGS enhances the reproducibility and transparency of experiments, which is essential from a research ethics perspective. The ACL Rolling Review checklist[12] explicitly emphasizes implementation and experimental settings, underscoring their importance to the community. Through continued maintenance and expansion of ENTITY-LINKINGS, we aim to further support these efforts.

While long-term community adoption ultimately depends on broader usage, we emphasize that we plan to actively use ENTITY-LINKINGS in our own future research and will therefore continue to maintain and update it on a regular basis. This library stems from the practical challenges we encountered while building and evaluating EL systems, and our goal is to share this solution with the broader community. Furthermore, this demonstra-

[12] https://aclrollingreview.org/responsibleNLPresearch/

tion paper presents only a subset of essential results and brief analyses for the purpose of system verification. More extensive experimental results are available in our GitHub repository. Nevertheless, the results reported in this paper are sufficient to validate our claims, and the paper can be read as a self-contained and complete study. By making our implementation details and validation results publicly available as much as possible, we aim to provide a broader impact to the research community. This transparency helps avoid redundant experimentation and allows researchers to focus their efforts more effectively. Accordingly, we plan to continuously add and update new results and perform regular maintenance. In this way, ENTITY-LINKINGS is intended to deliver sustained value beyond this paper alone.

We verified the licenses of all datasets used in this study, as summarized in Table 4 in Appendix A, and confirmed that their use complies with all applicable terms. In addition, this work does not involve the generation of harmful content. Accordingly, we ensure that our study is fully compliant with ethical guidelines such as the ACL Ethics Policy[13].

Acknowledgements

We thank the anonymous reviewers and the area chair for their valuable comments and suggestions.

The architecture design of ENTITY-LINKINGS is inspired by MBRS (Deguchi et al., 2024) and GEC-METRICS (Goto et al., 2025).

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

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.

Edoardo Barba, Luigi Procopio, and Roberto Navigli. 2022. ExtEnD: Extractive entity disambiguation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2478–2488, Dublin, Ireland. Association for Computational Linguistics.

Nicholas Botzer, Yifan Ding, and Tim Weninger. 2021. Reddit entity linking dataset. *Inf. Process. Manage.*, 58(3).

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 708–716, Prague, Czech Republic. Association for Computational Linguistics.

Hiroyuki Deguchi, Yusuke Sakai, Hidetaka Kamigaito, and Taro Watanabe. 2024. mbrs: A library for minimum Bayes risk decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 351–362, Miami, Florida, USA. Association for Computational Linguistics.

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. 2015. Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2):32–49.

Yifan Ding, Qingkai Zeng, and Tim Weninger. 2024. ChatEL: Entity linking with chatbots. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3086–3097, Torino, Italia. ELRA and ICCL.

Yotam Eshel, Noam Cohen, Kira Radinsky, Shaul Markovitch, Ikuya Yamada, and Omer Levy. 2017. Named entity disambiguation for noisy text. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 58–68, Vancouver, Canada. Association for Computational Linguistics.

Thibault Févry, Nicholas FitzGerald, Livio Baldini Soares, and Tom Kwiatkowski. 2020. Empirical evaluation of pretraining strategies for supervised entity linking. *Preprint*, arXiv:2005.14253.

Pierre-Yves Genest, Pierre-Edouard Portier, Elöd Egyed-Zsigmond, and Martino Lovisetto. 2023. Linked-DocRED – Enhancing DocRED with Entity-Linking to Evaluate End-To-End Document-Level Information Extraction Pipelines. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'23)*, page 11, Taipei, Taiwan. Association for Computing Machinery.

[13] https://www.aclweb.org/adminwiki/index.php/ACL_Policy_on_Publication_Ethics

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. 2019. Learning dense representations for entity retrieval. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 528–537, Hong Kong, China. Association for Computational Linguistics.

Takumi Goto, Yusuke Sakai, and Taro Watanabe. 2025. gec-metrics: A unified library for grammatical error correction evaluation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 524–534, Vienna, Austria. Association for Computational Linguistics.

Zhaochen Guo and Denilson Barbosa. 2014. Robust entity linking via random walks. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, pages 499–508.

Zhaochen Guo and Denilson Barbosa. 2018. Robust named entity disambiguation with random walks. *Semant. 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.

Johannes Hoffart, Stephan Seufert, Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2012. Kore: keyphrase overlap relatedness for entity disambiguation. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, page 545–554, New York, NY, USA. Association for Computing Machinery.

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.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength Natural Language Processing in Python.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. 2019. Zero-shot entity linking by reading entity descriptions. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3449–3460, Florence, Italy. Association for Computational Linguistics.

Xing Han Lù. 2024. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring. *Preprint*, arXiv:2407.03618.

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.

David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, page 509–518, New York, NY, USA. Association for Computing Machinery.

Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, Aldo Gangemi, Darío Garigliotti, and Roberto Navigli. 2015. Open knowledge extraction challenge. In *Semantic Web Evaluation Challenges*, pages 3–15. Springer.

Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, Aldo Gangemi, Robert Meusel, and Heiko Paulheim. 2016. The second open knowledge extraction challenge. In *Semantic Web Challenges*, pages 3–16, Cham. Springer International Publishing.

Yasumasa Onoe and Greg Durrett. 2020. Fine-grained entity typing for domain independent entity linking. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8576–8583.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–243, Denver, Colorado. Association for Computational Linguistics.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

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.

Lev Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1375–1384, Portland, Oregon, USA. Association for Computational Linguistics.

Michael Röder, Ricardo Usbeck, Sebastian Hellmann, Daniel Gerber, and Andreas Both. 2014. N^3 - a collection of datasets for named entity recognition and disambiguation in the NLP interchange format. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 3529–3533, Reykjavik, Iceland. European Language Resources Association (ELRA).

Susanna Rücker and Alan Akbik. 2025. Evaluating design decisions for dual encoder-based entity disambiguation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15685–15701, Vienna, Austria. Association for Computational Linguistics.

Michael Röder, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. 2018. GERBIL - Benchmarking Named Entity Recognition and Linking consistently. *Semantic Web*, 9(5):605–625.

Hassan Shavarani and Anoop Sarkar. 2023. SpEL: Structured prediction for entity linking. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11123–11137, Singapore. Association for Computational Linguistics.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical report, University of Massachusetts, Amherst, Tech. Rep. UM-CS-2012.

Simone Tedeschi, Simone Conia, Francesco Cecconi, and Roberto Navigli. 2021. Named Entity Recognition for Entity Linking: What works and what's next. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2584–2596, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Junxiong Wang, Ali Mousavi, Omar Attia, Ronak Pradeep, Saloni Potdar, Alexander Rush, Umar Farooq Minhas, and Yunyao Li. 2024a. Entity disambiguation via fusion entity decoding. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6524–6536, Mexico City, Mexico. Association for Computational Linguistics.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024b. Text embeddings by weakly-supervised contrastive pre-training. *Preprint*, arXiv:2212.03533.

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.

Wenzheng Zhang, Wenyue Hua, and Karl Stratos. 2022. EntQA: Entity linking as question answering. In *International Conference on Learning Representations*.

A Dataset Details

```
{
  "id": "doc-001-P1",
  "text": "Toyota makes cars.",
  "entities": [
    {
      "start": 0,
      "end": 6,
      "label": ["000011"],
    }
  ]
}
```

Listing 2: Dataset format

```
{
  "id": "000011",
  "name": "Toyota",
  "description": "Toyota is a Japanese car
    manufacturer."
}
```

Listing 3: Ontology format

Table 4 shows a complete list of datasets supported by ENTITY-LINKINGS. Additionally, arbitrary datasets and ontologies can be used with the ENTITY-LINKINGS library if they are provided in a JSONL format shown in Listing 2 and 3. Note that training and evaluation splits of AIDA-CoNLL dataset (Hoffart et al., 2011) are not publicly available, and users need to download them after obtaining approval for using the Reuters Corpora[14]. Once downloaded, users can use a preprocessing script we provide to convert AIDA-CoNLL dataset to the format supported in ENTITY-LINKINGS.

B Details of Experimental Setup

Table 5,6 list the hyperparameter settings used in our experiments. All models are trained and evaluated based on these settings. ENTITY-LINKINGS allows these parameters to be saved and loaded in YAML format, facilitating easy reproducibility.

[14] trec.nist.gov/data/reuters/reuters.html

data_id	Dataset	Domain	Lang.	Ontology	Train	Licence
msnbc	MSNBC (Cucerzan, 2007)	News	English	Wikipedia		Unknown*
aquaint	AQUAINT (Milne and Witten, 2008)	News	English	Wikipedia		Unknown*
ace2004	ACE2004 (Ratinov et al., 2011)	News	English	Wikipedia		Unknown*
kore50	KORE50 (Hoffart et al., 2012)	News	English	Wikipedia		CC BY-SA 3.0
n3-r128	N3-Reuters-128 (Röder et al., 2014)	News	English	Wikipedia		GNU AGPL-3.0
n3-r500	N3-RSS-500 (Röder et al., 2014)	RSS	English	Wikipedia		GNU AGPL-3.0
derczynski	Derczynski (Derczynski et al., 2015)	Twitter	English	Wikipedia		CC-BY 4.0
oke-2015	OKE-2015 (Nuzzolese et al., 2015)	News	English	Wikipedia	Yes	Unknown*
oke-2016	OKE-2016 (Nuzzolese et al., 2016)	News	English	Wikipedia	Yes	Unknown*
wned-wiki	WNED-WIKI (Guo and Barbosa, 2014)	Wikipedia	English	Wikipedia		Unknown
wned-cweb	WNED-CWEB (Guo and Barbosa, 2014)	Web	English	Wikipedia		Apache License 2.0
unseen	WikilinksNED Unseen-Mentions (Onoe and Durrett, 2020)	Web	English	Wikipedia	Yes	CC-BY 3.0*
tweeki	Tweeki EL (Harandizadeh and Singh, 2020)	Twitter	English	Wikipedia	Yes	Apache License 2.0
reddit-comments	Reddit EL (Botzer et al., 2021)	Reddit	English	Wikipedia		CC-BY 4.0
reddit-posts	Reddit EL (Botzer et al., 2021)	Reddit	English	Wikipedia		CC-BY 4.0
shadowlink-shadow	ShadowLink (Provatorova et al., 2021)	Wikipedia	English	Wikipedia		Unknown*
shadowlink-top	ShadowLink (Provatorova et al., 2021)	Wikipedia	English	Wikipedia		Unknown*
shadowlink-tail	ShadowLink (Provatorova et al., 2021)	Wikipedia	English	Wikipedia		Unknown*
zeshel	Zeshel (Logeswaran et al., 2019)	Wikia	English	Wikia	Yes	CC-BY-SA
docred	Linked-DocRED (Genest et al., 2023)	News	English	Wikipedia	Yes	CC-BY 4.0

Table 4: Public Entity Linking Datasets. *Unknown licence information as provided in the original sources.

	Parameters
Seeds	42
Training Epochs	2
Hard negatives	10
Batch size (train)	32
Batch size (eval)	256
Max token length (context)	128
Max token length (candidate)	50
Context window	500
Learning rate	1e-5
Gradient accumulation steps	4
Scheduler	linear
Optimizer	AdamW
Warmup	0.06
Weight decay	0.01
Max grad norm	0.0
Adam beta	[0.9, 0.98]
Adam epsilon	1e-6

Table 5: Hyperparameters for Candidate Generator

	ZELDA	AIDA-CoNLL
Seeds	42	42
Training Epochs	1 (10)	5 (30)
Candidates	30	30
Batch size (train)	8 (32)	8 (32)
Batch size (eval)	32 (256)	32 (256)
Max token length (context)	128	128
Max token length (candidate)	50	50
Learning rate	2e-5 (5e-5)	2e-5 (5e-5)
Gradient accumulation steps	4	4
Scheduler	linear	linear
Optimizer	AdamW	AdamW
Warmup	0.06	0.06
Weight decay	0.01	0.01
Max grad norm	0.0	0.0
Adam beta	[0.9, 0.98]	[0.9, 0.98]
Adam epsilon	1e-6	1e-6

Table 6: Hyperparameters for Candidate Reranker. The values in parentheses represent the parameters for FEVRY.

C Details of Command-Line Interface

Listing 4 illustrates the examples via CLI for training with ZELDA dataset and Dual-Encoder candidate generator. The training and evaluation processes for both the candidate retriever and reranker are instantiated through dedicated commands: `train-retriever`, `eval-retriever`, `train-reranker`, and `eval-reranker`. Additionally, `eval-pipeline` enables the end-to-end evaluation of the pipeline system, incorporating spaCy for mention detection. To facilitate rigorous model comparison, all training and evaluation workflows are integrated with Weights & Biases.

```
# Model Training
entitylinkings-train-retrieval \
    --retriever_id  dualencoder \
    --dataset_id zelda \
    --dictionary_id_or_path zelda \
    # Custom Corpus
    # --train_file train.jsonl \
    # --validation_file validation.jsonl \
    # --dictionary_id_or_path
      dictionary.jsonl \
    --output_dir save_model/ \
    --num_hard_negatives 10 \
    --num_train_epochs 2 \
    --train_batch_size 8 \
    --validation_batch_size 32 \
    --config configs/dualencoder.yaml \
    --gpus 0,1 \
    --wandb
```

Listing 4: An CLI example of training, using Dual-Encoder as candidate generator and ZELDA as a dataset.