EFFICIENT ENTITY DISAMBIGUATION WITH ADAPTIVE CANDIDATE PRUNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Entity disambiguation (ED) systems often use a two-stage pipeline: a candidate generator produces a fixed-size candidate set, and a more expensive candidate reranker selects the final prediction. However, because mention ambiguity varies substantially, reranking the same number of candidates for every mention wastes computation on easy cases. To address this issue, we propose a lightweight adaptive candidate pruning method that requires no additional model training. Inspired by conformal prediction, candidates are retained when their generator-score difference from the top candidate is below a threshold, yielding smaller sets for confident mentions and larger sets for ambiguous ones. We tune the threshold on development data for end-to-end accuracy, prioritizing practical utility over a formal coverage guarantee. Experiments on a standard ED benchmark show that our method reduces the average number of candidates passed to the reranker by 94% and achieves a 4.56× speedup, while improving accuracy. Further analyses demonstrate robustness to hyperparameter selection and show that the method places pruning boundaries close to the candidate depth required for each mention.

1 INTRODUCTION

Entity disambiguation (ED) links a mention in an input text to its corresponding entity in a knowledge base (KB). ED is often treated as a core component of entity linking (EL), where the system first identifies mention spans and then conducts ED. A widely used approach to ED is a two-stage pipeline comprising a candidate generator and a candidate reranker (Sevgili et al., 2022; Sawada et al., 2026). The candidate generator selects a small set of plausible entities from a large KB using scalable, low-latency methods such as dual-encoder (Gillick et al., 2019; Wu et al., 2020). The candidate reranker then predicts the most appropriate entity from the candidate set using a more accurate but computationally expensive model, including a cross-encoder (Logeswaran et al., 2019).



Figure 1: "Barack Obama" is relatively unambiguous, allowing aggressive pruning, whereas "Georgia" may refer to the country, the U.S. state, or the musician and therefore requires a larger candidate set.

In typical ED systems, the candidate generator produces a fixed number of candidates for every mention, and the entire set is passed to the reranker regardless of input difficulty (Wu et al., 2020; Xu et al., 2023; Wang et al., 2024a; Ding et al., 2024; Ye & Mitchell, 2025). However, mention ambiguity varies substantially. As Figure 1 illustrates, some require many candidates to preserve recall due to ambiguous surface forms or limited context, while others are easier, with clearly dominant top candidates. Using the same candidate set size for all mentions is therefore computationally inefficient, as the reranker performs substantial computation over candidates that are unlikely to be correct, particularly for trivial mentions where the correct entity is already highly ranked by the candidate generator. As a result, a large portion of the reranking effort is wasted on low-value candidates that do not contribute to improving prediction accuracy.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

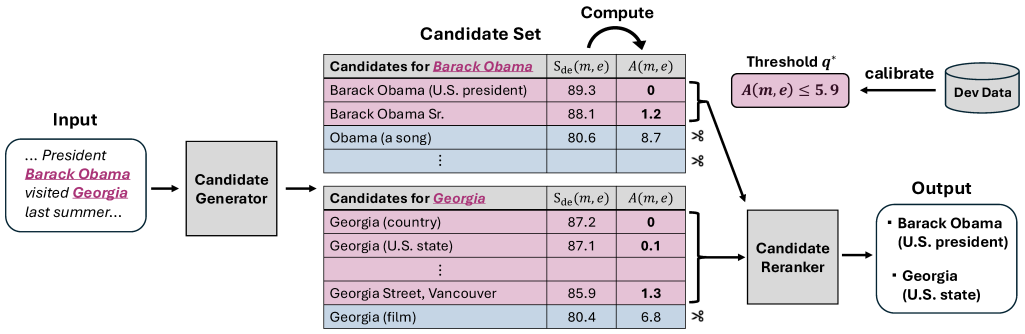

Figure 2: Overview of our adaptive candidate pruning method. For each mention m and candidate entity e, the candidate generator assigns a similarity score $s_{\text{de}}(m, e)$, and the nonconformity score $A(m, e)$ is computed as the difference from the score of the top candidate. The threshold q^*, selected on the development set, retains candidates satisfying $A(m, e) \leq q^*$ for reranking. The input text and scores are for illustrative purposes only.

Reducing unnecessary reranking is important in practical ED and EL settings where latency and computational cost are key constraints. Conversational systems require real-time EL to ground dialogue in knowledge bases (Laskar et al., 2022; Hoveyda et al., 2024). In the biomedical domain, costly rerankers become a bottleneck when normalizing mentions to large ontologies across millions of documents (Sanz-Cruzado & Lever, 2025). Low latency is also critical in crisis response, where EL helps geolocate natural disasters from rapidly changing social media streams (Caillaut et al., 2024). These applications motivate ED methods that maintain reranking accuracy while reducing the number of candidates scored by the reranker.

To meet these demands, we propose a lightweight adaptive candidate pruning method that requires no additional model training, as shown in Figure 2. We define each candidate's nonconformity as the difference between its generator score and the top candidate's score, and retain candidates whose difference falls below a threshold, inspired by conformal prediction (Vovk et al., 2005; Angelopoulos & Bates, 2022). This yields smaller candidate sets for confident mentions and larger sets for ambiguous ones. We select the threshold that maximizes end-to-end ED accuracy on the development set and apply it at test time. Standard conformal prediction provides a formal coverage guarantee under exchangeability between calibration and test examples, an assumption that may not hold in ED, where development and test data often come from different domains. The absence of this guarantee, however, does not preclude effective candidate pruning. By adopting threshold selection based on the nonconformity score from conformal prediction, our adaptive candidate pruning method improves ED efficiency while maintaining accuracy. Because it uses scores already computed by the candidate generator and requires no additional training, the method adds negligible cost and can be integrated into existing systems with minimal overhead.

Experimental results on the heterogeneous ZELDA test datasets (Milich & Akbik, 2023) show that our method improves both ED efficiency and accuracy. Compared with reranking all 30 candidates, it reduces the average number of candidates passed to the cross-encoder from 30.0 to 1.79 (a 94.0% reduction) and achieves a 4.56× speedup, while improving macro-average accuracy by 0.8 points. It also outperforms pruning baselines in both efficiency and accuracy. Analysis of prediction changes explains this accuracy gain: the method removes incorrect candidates that the reranker would otherwise select as final predictions and consequently corrects more reranking errors than it introduces. Further analyses show that these gains are robust across a broad range of hyperparameter values and that our method places pruning boundaries close to the candidate depth required for each mention.

2 Background

2.1 Task Definition and ED pipeline

Entity disambiguation (ED) is the task of selecting the correct KB entry for a given mention whose span has already been identified in the input text. Formally, let $\boldsymbol{t} = (t_1, t_2, \ldots, t_n)$ denote an

input text represented as a sequence of tokens. We assume that a mention span m is given, where $m = t_i, \ldots, t_j$ is a contiguous subsequence of $\boldsymbol{t}$ satisfying $1 \leq i \leq j \leq n$. Given the mention m, its surrounding context $\boldsymbol{t}$, and a KB containing a set of entities $\mathcal{E}$, ED aims to predict an entity $e \in \mathcal{E}$ that corresponds to m in $\boldsymbol{t}$.

In principle, ED can be formulated as selecting the entity with the highest compatibility score from the entire KB. However, directly scoring every entity in $\mathcal{E}$ is often impractical, especially when the KB contains millions of entries. Therefore, many ED systems adopt a two-stage pipeline comprising a candidate generator and a candidate reranker (Sawada et al., 2026). Given a mention m and its context $\boldsymbol{t}$, the candidate generator first produces a candidate set $\mathcal{C}_K(m) \subset \mathcal{E}$ of size K ($K \ll |\mathcal{E}|$) that is likely to contain the correct entity. The candidate reranker then selects the most plausible entity from $\mathcal{C}_K(m)$. While various model architectures have been proposed for these two ED components (Nakatani et al., 2025; Févry et al., 2020; Barba et al., 2022; Wang et al., 2024a; Ding et al., 2024), pairing a dual-encoder candidate generator with a cross-encoder reranker is common practice (Wu et al., 2020; Botha et al., 2020; Liu et al., 2023; Gallego et al., 2024).

2.2 DUAL-ENCODER

The dual-encoder scores the relevance of a mention m in context $\boldsymbol{t}$ to a KB entity $e \in \mathcal{E}$ using the dot product of their embeddings (Wu et al., 2020):

$$s_{\mathrm{de}}(m, e) = \boldsymbol{h}_m^\top \boldsymbol{h}_e, \tag{1}$$

where $\boldsymbol{h}_m$ and $\boldsymbol{h}_e$ denote the vector representations of the mention and entity, respectively. These representations are obtained by independently encoding the mention-side input and the entity-side input:

$$\boldsymbol{h}_m = \mathrm{red}(E_m(\tau_m)), \quad \boldsymbol{h}_e = \mathrm{red}(E_e(\tau_e)), \tag{2}$$

where τ_m and τ_e are the input token sequences constructed from the mention context and the entity description, respectively. E_m and E_e are encoders for the mention and entity inputs, and $\mathrm{red}(\cdot)$ is a reduction function that maps the encoder output to a fixed-dimensional vector by extracting the hidden representation corresponding to the `[CLS]` token. The candidate generator then produces the candidate set $\mathcal{C}_K(m)$ based on dual-encoder scores:

$$\mathcal{C}_K(m) = \mathrm{TopK}_{e \in \mathcal{E}}\, s_{\mathrm{de}}(m, e), \tag{3}$$

where $\mathrm{TopK}_{e \in \mathcal{E}}\, s_{\mathrm{de}}(m, e)$ denotes the set of K entities in $\mathcal{E}$ with the largest values of $s_{\mathrm{de}}(m, e)$.

The dual-encoder is computationally efficient because entity representations can be precomputed and stored independently of input mentions. During inference, only $\boldsymbol{h}_m$ must be computed, and high-scoring entities can be selected efficiently by nearest-neighbor search over the stored entity vectors using libraries such as Faiss (Douze et al., 2026). However, this separate encoding prevents the dual-encoder from modeling detailed interactions between the mention and entity.

2.3 CROSS-ENCODER

Unlike the dual-encoder, which encodes mentions and entities independently, the cross-encoder jointly encodes each mention–entity pair (Wu et al., 2020). The candidate with the highest cross-encoder score is predicted as the ED output:

$$\hat{e} = \arg \max_{e \in \mathcal{C}_K(m)} s_{\mathrm{ce}}(m, e). \tag{4}$$

The score $s_{\mathrm{ce}}(m, e)$ is computed by applying a linear scoring layer to the joint representation $\boldsymbol{h}_{m,e}$ of the mention–entity pair:

$$s_{\mathrm{ce}}(m, e) = \boldsymbol{w}^\top \boldsymbol{h}_{m,e} + b, \tag{5}$$

$$\boldsymbol{h}_{m,e} = \mathrm{red}(E_{\mathrm{ce}}(\tau_{m,e})). \tag{6}$$

Here, $\tau_{m,e}$ denotes the input token sequence constructed from the mention context and the entity description, and E_{ce} is an encoder.

The cross-encoder is more expressive than the dual-encoder because it jointly encodes the mention context and entity description, capturing fine-grained interactions. However, this makes inference computationally expensive. Since $\boldsymbol{h}_{m,e}$ depends on each pair (m, e), entity representations cannot be precomputed. For a candidate set of size K, the cross-encoder must evaluate K mention–candidate pairs. Thus, the computational cost grows approximately linearly with K.

3 ADAPTIVE CANDIDATE PRUNING

In this study, we focus on an ED pipeline with a dual-encoder candidate generator and a cross-encoder reranker, as this configuration is widely adopted in ED systems. Because the computational cost of the cross-encoder grows approximately linearly with the number of candidates, passing a fixed-size candidate set without considering mention difficulty is inefficient. To address this, we propose a lightweight adaptive candidate pruning method that requires no additional model training, inspired by conformal prediction (Vovk et al., 2005; Angelopoulos & Bates, 2022).

3.1 CONFORMAL PREDICTION

Conformal prediction is a general framework for quantifying predictive uncertainty by constructing a set of plausible outputs rather than returning only a single prediction. It requires training data for fitting a predictive model and a separate held-out calibration set $\mathcal{D}_{\text{cal}} = \{(x_i, y_i)\}_{i=1}^{n_{\text{cal}}}$, where x_i is the input of the i-th example, y_i is the corresponding correct output, and n_{cal} is the number of calibration examples. For each calibration example, a nonconformity score $A(x_i, y_i)$ is calculated, where $A(x, y)$ measures how incompatible a possible output y is with an input x. A larger score indicates greater incompatibility. Given a user-specified error rate $\alpha \in (0, 1)$, the threshold q_α is set to the $\lceil (n_{\text{cal}} + 1)(1 - \alpha) \rceil$-th smallest nonconformity score among the calibration examples.

For a new input x, conformal prediction constructs the prediction set $\Gamma_\alpha(x)$ by including all outputs whose nonconformity scores do not exceed the calibrated threshold q_α:

$$\Gamma_\alpha(x) = \{y \in \mathcal{Y} \mid A(x, y) \leq q_\alpha\}, \tag{7}$$

where $\mathcal{Y}$ denotes the space of possible outputs. If the calibration and test examples are exchangeable, conformal prediction guarantees that the correct output is included in the prediction set with probability at least $1 - \alpha$:

$$\Pr\left(Y_{\text{test}} \in \Gamma_\alpha(X_{\text{test}})\right) \geq 1 - \alpha, \tag{8}$$

where X_{test} and Y_{test} denote a random test input and its corresponding correct output, respectively.

3.2 ADAPTATION TO ENTITY DISAMBIGUATION

The coverage guarantee in Equation 8 relies on the calibration and test examples being exchangeable. Although this assumption is reasonable in many standard machine learning settings, it is often difficult to satisfy in ED because systems are frequently evaluated out of domain. For example, ZELDA (Milich & Akbik, 2023), a commonly used ED benchmark, uses training data derived from Wikipedia but evaluates models on text from diverse domains, including web pages, news articles, and social media. In such settings, obtaining a held-out calibration set that follows the same distribution as the test data is generally impractical.

Nevertheless, the central idea of retaining a set of plausible outputs is well suited to adaptive candidate pruning: the candidate set should be small for confident mentions while remaining large enough for ambiguous ones. Therefore, we adapt the prediction set construction of conformal prediction to candidate pruning in ED, although the coverage guarantee in Equation 8 may not hold. Prior work has used in-distribution calibration data to select conformal thresholds and applied them to distribution-shifted test data without assuming that the standard coverage guarantee holds (Kasa et al., 2025). Conformal prediction has also been used to guide practical downstream decisions under distribution shift (Pearson et al., 2026). Following this perspective, we adopt the threshold selection framework based on the nonconformity score from conformal prediction as an empirically useful mechanism for adaptive candidate pruning, instead of a means of obtaining the theoretical coverage guarantee. Rather than interpreting α as a user-specified error rate and estimating a threshold from a separate, exchangeable calibration set, we treat α as a hyperparameter and tune it on the development split used for hyperparameter selection in model training.

3.3 THRESHOLD SELECTION AND PRUNING PROCEDURE

Figure 2 provides the overview of the proposed pruning procedure. For each candidate set $\mathcal{C}_K(m)$, let $e_{(r)}$ denote its r-th highest-scoring entity according to the candidate generator, such that

$s_{\mathrm{de}}(m, e_{(1)}) \geq \cdots \geq s_{\mathrm{de}}(m, e_{(K)})$. We define the nonconformity score of an entity $e \in \mathcal{C}_K(m)$ as its score difference from the top-ranked candidate:

$$A(m, e) = s_{\mathrm{de}}(m, e_{(1)}) - s_{\mathrm{de}}(m, e). \tag{9}$$

This definition assigns a nonconformity score of zero to the top-ranked candidate and larger values to candidates with substantially lower dual-encoder scores. The candidate generator is trained to assign high scores to gold entities, but it may also assign high scores to incorrect entities that are difficult to distinguish using its representations. Prior ED and EL work treats such high-scoring incorrect entities as hard negatives and uses them to improve retrieval models (Zhang & Stratos, 2021; Monath et al., 2023). These observations motivate treating candidates whose scores are close to the top score as candidates requiring closer examination by the candidate reranker.

Given a threshold q, we retain candidates whose nonconformity scores do not exceed it:

$$\widetilde{\mathcal{C}}(m; q) = \{e \in \mathcal{C}_K(m) \mid A(m, e) \leq q\}. \tag{10}$$

The pruning rule retains the top-ranked elements of $\mathcal{C}_K(m)$ up to an adaptive cutoff, and $\widetilde{\mathcal{C}}(m; q)$ is always nonempty because it includes $e_{(1)}$. Its size varies across mentions: a large score drop near the top of the ranking yields a small candidate set, whereas a relatively flat score distribution retains more candidates.

To select the pruning threshold, we consider candidate values of α from a predefined search space $\mathcal{A}$. For each development example, we compute the nonconformity score of its gold entity. If the gold entity is not in the candidate set $\mathcal{C}_K(m)$, we assign it a nonconformity score of $+\infty$. For each $\alpha \in \mathcal{A}$, we derive a corresponding threshold q_α as the $\lceil (n_{\mathrm{dev}} + 1)(1 - \alpha) \rceil$-th smallest value of these scores. We then apply each threshold to every development example, rerank the retained candidates with the cross-encoder, and measure the end-to-end ED accuracy. When only one candidate is retained, it is returned as the final prediction directly without a cross-encoder forward pass. The threshold that achieves the highest development accuracy is selected:

$$\begin{aligned} q^* &= q_{\alpha^*}, \\ \alpha^* &= \arg\max_{\alpha \in \mathcal{A}} \mathrm{Acc}_{\mathrm{dev}}(q_\alpha). \end{aligned} \tag{11}$$

When thresholds derived from multiple α values achieve the same accuracy, we select the one among them that retains the smallest average number of candidates passed to the cross-encoder.

At test time, the selected threshold q^* is fixed and applied to the candidate set for each mention. The cross-encoder then predicts

$$\widehat{e} = \arg\max_{e \in \widetilde{\mathcal{C}}(m; q^*)} s_{\mathrm{ce}}(m, e). \tag{12}$$

As in the threshold selection procedure, when only one candidate is retained, it is returned without cross-encoder inference. The pruning procedure itself uses only the dual-encoder scores already computed during candidate generation and thus introduces negligible additional inference cost.

4 Experimental Setup

Dataset We use ZELDA (Milich & Akbik, 2023), a widely used ED benchmark, for model training and evaluation. ZELDA provides a training split comprising 95K Wikipedia paragraphs with 2.6 million mention spans, a KB containing 821K entities, and nine evaluation datasets spanning domains such as Wikipedia, news articles, and tweets. We divide the original training split into training and development sets in a ratio of 9:1.

Training, Threshold Selection, and Inference We construct an ED pipeline using a dual-encoder as the candidate generator and a cross-encoder as the candidate reranker, both initialized with BERT[1] (Devlin et al., 2019). Using the training and development sets, we first train the dual-encoder and use it to generate a candidate set for each mention; we then train the cross-encoder on these generated candidate sets. To select the candidate pruning threshold, we perform end-to-end inference on the development set with the trained models and evaluate thresholds derived from $\alpha \in \mathcal{A}$. The

[1] https://huggingface.co/google-bert/bert-base-uncased

Table 1: Dual-encoder and cross-encoder pipeline performance on ZELDA. Acc. is macro-average test accuracy. Inference time covers the complete test set. Avg. Cands is the micro-average number of candidates passed to the cross-encoder. Down arrows indicate that lower is better. For Acc. and each test set, bold and underlined values indicate the best and second-best results, respectively, excluding Oracle. For inference time and Avg. Cands, bold and underlined values indicate the best and second-best results, respectively, excluding Oracle and No-Reranking.

	Acc.	Inference Time (s) ↓	Avg. Cands ↓	AIDA-B	TWEEKI	REDDIT-POSTS	REDDIT-COMM.	WNED-CWEB	WNED-WIKI	SLINKS-TAIL	SLINKS-SHADOW	SLINKS-TOP
Oracle	87.2	$437_{\pm 3.6}$	2.84	89.7	83.7	95.6	92.5	76.1	93.3	99.6	75.8	78.2
No-Reranking	81.0	$271_{\pm 0.6}$	0.00	83.5	77.9	92.3	87.1	70.5	90.2	99.0	61.9	66.2
No-Pruning	82.7	$1732_{\pm 16.3}$	30.0	83.2	79.1	<u>93.0</u>	89.8	<u>71.9</u>	90.8	<u>99.3</u>	**69.7**	67.8
Tuned-Fixed-k	82.8	$747_{\pm 12.5}$	10.0	83.8	79.2	<u>93.0</u>	90.0	71.7	<u>90.9</u>	<u>99.3</u>	<u>69.0</u>	<u>68.3</u>
Adaptive-k	<u>82.9</u>	<u>508</u>$_{\pm 1.7}$	<u>4.38</u>	<u>84.3</u>	<u>80.0</u>	**93.2**	<u>90.3</u>	71.7	90.8	**99.4**	68.4	67.6
Ours	**83.5**	**380**$_{\pm 3.3}$	**1.79**	**85.5**	**80.2**	**93.2**	**90.8**	**72.6**	**91.2**	**99.4**	68.1	**70.4**

search space $\mathcal{A}$ contains 0.001, 0.005, values from 0.01 to 0.10 in increments of 0.01, and 0.20. At test time, we apply the selected threshold q^* to each evaluation dataset and produce one entity prediction per mention. Following Sawada et al. (2026), we set the initial candidate set size to $K = 30$ during training, threshold selection, and inference. We report end-to-end accuracy on each ZELDA evaluation dataset and the macro-average across datasets, along with inference time and the average number of candidates passed to the cross-encoder. When pruning retains only one candidate, we return it directly without cross-encoder inference and therefore count zero candidates as having been passed to the cross-encoder for that mention.

Baselines and Oracle Reference We compare our method with four baselines. **No-Reranking** directly returns the top-ranked dual-encoder candidate $e_{(1)}$ without invoking the cross-encoder. **No-Pruning** passes all K candidates in $\mathcal{C}_K(m)$ to the cross-encoder. **Tuned-Fixed-k** selects a single value of $k \in \{1, 2, \ldots, K\}$ that maximizes end-to-end accuracy on the development set without using the nonconformity score and passes the top k candidates to the cross-encoder for every mention. **Adaptive-k** adapts the context selection method of Taguchi et al. (2025) to ED. It computes score drops between adjacent candidates, sets the cutoff at the largest drop, and retains all candidates up to the cutoff plus the next B candidates as a buffer. We tune B over $\{0, 1, 2, 3, 4, 5\}$ on the development set. For reference, we also report an **Oracle** that retains the shortest prefix of the ranked candidate list containing the gold entity, thereby providing an upper bound for mention-specific pruning. If the gold entity is absent from $\mathcal{C}_K(m)$, the Oracle retains the entire list.

Appendix B describes further details on the experimental setup.

5 Results and Discussion

Table 1 shows end-to-end accuracy, inference time, and average candidate set size used for the cross-encoder inference by our method and baselines. Development-set threshold selection yields $\alpha = 0.05$, $B = 3$, and $k = 10$ for the proposed method, Adaptive-k, and Tuned-Fixed-k, respectively.

Our method improves **both efficiency and effectiveness** of the ED pipeline. It achieves the highest macro-average accuracy among the non-oracle systems, reaching 83.5 while requiring only 1.79 candidates per mention on average for cross-encoder inference. Compared with No-Pruning, it reduces the average number of candidates passed to the cross-encoder from 30.0 to 1.79 (a 94.0% reduction) and the total inference time from 1,732 to 380 seconds (a 4.56× speedup). Despite this substantially smaller candidate set, accuracy improves by 0.8 points. Thus, reranking all the candidates from the candidate generator is not only inefficient but can also be detrimental: low-ranked candidates that are unlikely to be correct may receive spuriously high cross-encoder scores and distract the final prediction. The proposed method removes many such candidates while preserving alternatives when the dual-encoder scores indicate greater ambiguity.

Compared with Tuned-Fixed-k and Adaptive-k, our method also achieves higher macro-average accuracy while reranking substantially fewer candidates and requiring considerably less inference

time. This comparison demonstrates the benefit of combining mention-specific pruning with threshold selection using the development set. Unlike our method, Tuned-Fixed-k retains the same number of candidates for every mention and therefore cannot adapt the candidate set size to mention-specific score distributions. Although Adaptive-k varies the cutoff across mentions, it determines each boundary from the largest adjacent score drop, which may reflect a local fluctuation rather than the range of scores within which the gold entity is likely to exist. In contrast, our method measures every candidate relative to the top-ranked entity, derives candidate thresholds from the development-set distribution of gold-entity score differences, and selects the threshold that maximizes end-to-end development accuracy. Thus, it uses supervision across mentions to establish a globally meaningful notion of candidate plausibility while still producing a different candidate set size for each mention.

Additionally, our efficient pruning approach maintains or improves performance relative to the baselines across multiple evaluation domains, suggesting that a single domain does not drive the aggregate gain. This implies that the adaptive pruning threshold selected using the Wikipedia-based development set **transfers effectively to datasets from diverse domains**. In Appendix C, we show additional experimental results using a text-embedding model (Wang et al., 2024b) as a candidate generator and various candidate set sizes K, as well as a more challenging evaluation setup with the GERBIL (Röder et al., 2018) benchmark and the KILT (Petroni et al., 2021) KB, demonstrating that the gains of our approach persist across different model architectures, candidate set sizes, and benchmarks. We also report paired bootstrap confidence intervals for the performance differences between our method and each baseline, providing evidence for the reliability of the observed gains.

6 Analysis

We will focus on summarizing important aspects here and defer additional analyses to Appendix D.

6.1 α-Sweep Analysis

To assess sensitivity to α, we evaluate each value in the hyperparameter search space $\mathcal{A}$ in terms of macro-average accuracy and the micro-average number of candidates passed to the cross-encoder across the ZELDA test datasets. Figure 3 summarizes the results.

At $\alpha = 0.05$, our method achieves the highest test accuracy while reranking fewer than two candidates per mention on average, thereby improving both efficiency and accuracy over the reranking baselines, i.e., No-Pruning, Tuned-Fixed-k, and Adaptive-k. Notably, this is also the value selected on the development set. More broadly, the method remains effective across a reasonably broad range of α values. For $0.03 \leq \alpha \leq 0.07$, it substantially reduces the number of reranked candidates while achieving accuracy that is comparable to or higher than that of the reranking baselines. Furthermore, the selected value lies near the center of this high-performing range rather than at an isolated optimum, indicating that moderate deviations from the development-set choice do not eliminate the gains. Performance changes substantially only at the extremes of the search space. When $\alpha \leq 0.02$, the method retains all candidates and is therefore equivalent to No-Pruning. At $\alpha = 0.08$, pruning becomes overly aggressive, nearly eliminating cross-encoder reranking and reducing accuracy below the reranking baselines. When $\alpha \geq 0.09$, the method becomes equivalent to No-Reranking. Thus, the method provides **consistent efficiency and accuracy gains across the practically useful range** $0.03 \leq \alpha \leq 0.07$, with degradation occurring only under extreme settings.

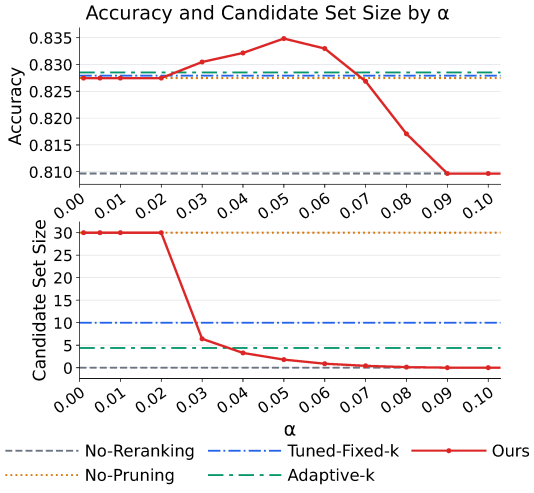

Figure 3: Macro-average accuracy and micro-average candidate set size across ZELDA test sets by α. The plot omits $\alpha = 0.2$ for clarity.

Table 2: Candidate set coverage and end-to-end prediction changes relative to No-Pruning. Cov. is candidate set coverage after candidate pruning (%). † indicates that a subset with infinite R/H (rescues but no harms) is excluded from the macro average.

	Macro		Micro			
	Cov.	R/H	Cov.	R/H	Rescue	Harm
Oracle	95.1	–	93.0	–	1149	0
No-Reranking	81.0	0.68	79.2	0.76	945	1250
No-Pruning	95.1	–	93.0	–	0	0
Tuned-Fixed-k	93.5	1.25	91.0	1.01	152	150
Adaptive-k	91.8	1.35†	89.3	1.09	286	262
Ours	89.1	1.93	86.7	1.73	536	309

Table 3: Absolute gap between the gold entity rank and pruned candidate set size. Lower values are better. Excl. omits mentions for which the gold entity is not among the 30 candidates from the candidate generator; Incl. assigns rank 31 to entities corresponding to those mentions.

	Macro		Micro	
	Excl.	Incl.	Excl.	Incl.
Oracle	0.00	0.05	0.00	0.07
No-Reranking	0.59	2.02	0.66	2.71
No-Pruning	28.4	27.1	28.3	26.4
Tuned-Fixed-k	8.68	9.29	8.68	9.54
Adaptive-k	3.31	4.41	3.33	4.91
Ours	0.95	1.81	0.95	2.26

6.2 Candidate Set Coverage and End-to-End Accuracy

We investigate how candidate pruning affects candidate set coverage and final ED predictions. Candidate set coverage measures how often the gold entity remains after pruning. Although it provides an upper bound on end-to-end accuracy, it does not capture whether pruning corrects or degrades the final prediction. Pruning can correct a prediction by removing an incorrect candidate that would otherwise outrank the gold entity, but it can also degrade one by removing the gold entity that the reranker would otherwise select. To analyze both of these aspects, we also compare each pruning method with No-Pruning using *rescue* and *harm* measures at the end-to-end prediction level.

Let $\mathcal{M}$ be the set of mentions in the test set, g_m the gold entity for mention m, and $\hat{e}_m^{\mathrm{NP}}$ and $\hat{e}_m^{\mathrm{P}}$ the predictions of No-Pruning and the evaluated pruning method, respectively. We define the rescue set $\mathcal{R}$ and harm set $\mathcal{H}$ as

$$\mathcal{R} = \{m \in \mathcal{M} \mid \hat{e}_m^{\mathrm{NP}} \neq g_m,\ \hat{e}_m^{\mathrm{P}} = g_m\}, \tag{13}$$

$$\mathcal{H} = \{m \in \mathcal{M} \mid \hat{e}_m^{\mathrm{NP}} = g_m,\ \hat{e}_m^{\mathrm{P}} \neq g_m\}. \tag{14}$$

The *rescue–harm ratio* is $\mathrm{R/H} = |\mathcal{R}|/|\mathcal{H}|$. A value above one indicates that pruning corrects more No-Pruning errors than the number of correct No-Pruning predictions it degrades. We report both micro and macro R/H. Micro R/H is the ratio of rescue and harm counts pooled across the test datasets, whereas macro R/H is the unweighted mean of the per-dataset ratios.

Table 2 reports these measures. Candidate set coverage decreases as pruning becomes more aggressive (see also "Avg. Cands" in Table 1), and the proposed method has lower coverage than the alternative pruning methods, i.e., Tuned-Fixed-k and Adaptive-k. Nevertheless, Table 1 shows that the proposed method achieves higher end-to-end accuracy. The rescue–harm analysis explains this apparent discrepancy. The proposed method achieves higher macro and micro R/H ratios than Tuned-Fixed-k and Adaptive-k. Tuned-Fixed-k and Adaptive-k correct approximately as many predictions as they degrade, whereas the **proposed method yields substantially more rescues than harms**. This favorable balance enables our method to improve end-to-end accuracy despite retaining the gold entity less frequently.

6.3 Gap between Gold Rank and Pruning Boundary

To assess how well each pruning strategy adapts its boundary to the candidate depth required for each mention, we measure the gap between the rank of the gold entity and the pruning boundary. Let r_m denote the rank of the gold entity in the initial candidate set $\mathcal{C}_K(m)$ and b_m the pruning boundary, i.e., the number of candidates retained for mention m. We compute the absolute rank gap $|r_m - b_m|$ for each test example. *Excl.* metric excludes mentions whose gold entity is absent from the initial candidate set. *Incl.* metric includes these mentions by assigning the gold entity the rank $K+1$. We report both the macro average across datasets and the micro average across all mentions.

Table 3 shows that our method places the pruning boundary substantially closer to the gold rank than No-Pruning, Tuned-Fixed-k, and Adaptive-k on both Incl. and Excl., and outperforms No-Reranking on Incl. Although No-Reranking has a slightly smaller Excl. gap, this likely occurs because gold entities often rank near the top and it retains only the top candidate. Table 2 shows that No-Reranking causes more harm than rescues, with macro and micro R/H ratios below one. Its small Excl. value therefore does not indicate a favorable rescue–harm balance. Overall, **our method places the pruning boundary close to the gold rank** while retaining enough candidates to achieve a favorable rescue–harm balance and higher end-to-end accuracy.

7 Related Work

7.1 Conformal Prediction and Distribution Shift

Conformal prediction has been applied to information retrieval and retrieval-augmented pipelines to construct calibrated candidate or prediction sets with statistical guarantees (Li et al., 2022; Wang & Huang, 2024; Li et al., 2024; Intrator et al., 2024). These guarantees rely on exchangeability between calibration and test examples, which may not hold in ED as stated in Section 3.2. Nevertheless, recent work has explored the practical utility of conformal methods under distribution shift, applying thresholds derived from a calibration set to shifted test distributions (Kasa et al., 2025) and using conformal prediction as a means of uncertainty estimation to guide downstream decision-making without relying on the formal coverage guarantee (Pearson et al., 2026). Our work follows this perspective and employs a threshold selection procedure inspired by conformal prediction for adaptive candidate pruning in ED, prioritizing practical utility over the formal coverage guarantee.

7.2 Efficient Candidate and Context Selection

Prior work has studied efficient candidate selection and adaptive reranking methods for information retrieval tasks. Song et al. (2024) prune candidates using extra self-attention layers added to the dual-encoder, while other works (Bahri et al., 2020; Wu et al., 2021; Gonzalez-Gamella et al., 2026) train separate truncation or reranking-decision models using Transformer (Vaswani et al., 2017), bidirectional LSTM (Graves & Schmidhuber, 2005), or XGBoost (Chen & Guestrin, 2016). Unlike our method, these approaches rely on additional learned components, requiring extra training and inference for candidate pruning. Our method instead operates directly on dual-encoder scores already available from retrieval, requiring no additional training and adding negligible pruning overhead.

Adaptive-k (Taguchi et al., 2025) offers a more directly comparable approach. Originally proposed for long-context question answering, Adaptive-k selects the number of passages based on the largest adjacent drop in retrieval scores and a fixed buffer. Among existing approaches, Adaptive-k is one of the few that, like our method, provide input-specific selection without additional training or substantial computational overhead. This close methodological similarity motivates its inclusion as our main adaptive pruning baseline. However, unlike our development-set-calibrated threshold, its selection boundary depends on a local score drop.

8 Conclusion

We presented a lightweight adaptive candidate pruning method for two-stage ED pipelines that does not require additional model training. The method measures each candidate's candidate generator score difference from the top-ranked candidate and applies a threshold selected on the development set, producing a mention-specific candidate set size without additional training or substantial computational overhead. Experiments on the heterogeneous ZELDA test datasets showed that our method outperformed the baselines in both efficiency and accuracy. The rescue–harm analysis further showed that pruning produced a net increase in correct predictions, helping explain its higher end-to-end accuracy. The method was also robust to hyperparameter choice and placed pruning boundaries close to the required candidate depth. Overall, adaptive pruning substantially reduces candidate reranker computation while improving prediction accuracy. Limitations and future extensions of this work are discussed in Appendix A.

AI USE STATEMENT

In this work, we used generative AI tools to design or provide feedback on research methodology or experiments, implement methods, assist with translation, and clean and reformat datasets. We did not use generative AI tools to help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in writing proofs, propose or refine hypotheses, or interpret results. Generating synthetic datasets and supporting qualitative or thematic data analysis were not applicable to this work. Additionally, we used generative AI tools to create or modify scientific figures or images, create or edit software code, create research artifacts and draft parts of this paper, summarize or analyze existing literature, brainstorm, search for information, edit the research paper to improve readability, and identify relevant literature. We have reviewed all AI-assisted work, independently verified its accuracy and appropriateness, and revised or corrected the outputs where necessary before incorporating them into the final manuscript and research artifacts. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This study uses the existing ZELDA benchmark and does not introduce new data collection, annotation, or experiments involving human participants. The benchmark contains text and entity information derived from existing sources. Therefore, its use may inherit biases present in the underlying corpora and knowledge base. In particular, entities that are less frequently represented, associated with underrepresented communities, or absent from the knowledge base may be linked less accurately. These concerns primarily arise from the underlying benchmark data rather than from the proposed candidate-pruning method itself, as this work focuses on developing an inference-time method without constructing or modifying the dataset. Nevertheless, our method may inherit and propagate these biases to downstream applications.

REPRODUCIBILITY STATEMENT

To support reproducibility, Section 3.3 specifies the concrete procedures of our candidate pruning method, including how candidate thresholds are constructed, selected using development-set accuracy, and applied at test time. Section 4 describes the datasets, training and development split, evaluation metrics, and baseline configurations, together with the hyperparameter search spaces used for threshold selection and baseline tuning. Appendix B details the training objectives, negative sampling procedures, checkpoint selection criteria, and computational resources. Training hyperparameters, including random seeds, learning rates, and batch sizes, are reported in Table 4. Appendix B also documents the inference-time measurement protocol, including the hardware, warm-up procedure, and repeated measurements. We are ready to publish all experimental code after acceptance to ensure reproducibility.

REFERENCES

Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification, 2022. URL https://arxiv.org/abs/2107.07511.

Tom Ayoola, Shubhi Tyagi, Joseph Fisher, Christos Christodoulopoulos, and Andrea Pierleoni. ReFinED: An efficient zero-shot-capable approach to end-to-end entity linking. In Anastassia Loukina, Rashmi Gangadharaiah, and Bonan Min (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pp. 209–220, Hybrid: Seattle, Washington + Online, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-industry.24. URL https://aclanthology.org/2022.naacl-industry.24/.

Dara Bahri, Yi Tay, Che Zheng, Donald Metzler, and Andrew Tomkins. Choppy: Cut transformer for ranked list truncation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pp. 1513–1516, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401188. URL https://doi.org/10.1145/3397271.3401188.

Edoardo Barba, Luigi Procopio, and Roberto Navigli. ExtEnD: Extractive entity disambiguation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2478–2488, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.177. URL https://aclanthology.org/2022.acl-long.177/.

Jan A. Botha, Zifei Shan, and Daniel Gillick. Entity Linking in 100 Languages. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7833–7845, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.630. URL https://aclanthology.org/2020.emnlp-main.630/.

Nicholas Botzer, Yifan Ding, and Tim Weninger. Reddit entity linking dataset. *Information Processing & Management*, 58(3):102479, 2021. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2020.102479. URL https://www.sciencedirect.com/science/article/pii/S0306457320309687.

Gaëtan Caillaut, Samuel Auclair, Cécile Gracianne, Nathalie Abadie, and Guillaume Touya. Entity linking for real-time geolocation of natural disasters from social network posts. *PLOS ONE*, 19(10):1–29, 10 2024. doi: 10.1371/journal.pone.0307254. URL https://doi.org/10.1371/journal.pone.0307254.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342322. doi: 10.1145/2939672.2939785. URL https://doi.org/10.1145/2939672.2939785.

Silviu Cucerzan. Large-scale named entity disambiguation based on Wikipedia data. In Jason Eisner (ed.), *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 708–716, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://aclanthology.org/D07-1074/.

Leon Derczynski, Diana Maynard, Giuseppe Rizzo, Marieke van Erp, Genevieve Gorrell, Raphaël Troncy, Johann Petrak, and Kalina Bontcheva. Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2):32–49, 2015. ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2014.10.006. URL https://www.sciencedirect.com/science/article/pii/S0306457314001034.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423/.

Yifan Ding, Qingkai Zeng, and Tim Weninger. ChatEL: Entity linking with chatbots. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 3086–3097, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.275/.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *IEEE Transactions on Big Data*, 12(2):346–361, 2026. doi: 10.1109/TBDATA.2025.3618474. URL https://ieeexplore.ieee.org/document/11202651.

Thibault Févry, Nicholas FitzGerald, Livio Baldini Soares, and Tom Kwiatkowski. Empirical evaluation of pretraining strategies for supervised entity linking. In *Automated Knowledge Base Construction*, 2020. doi: 10.24432/C59G6S. URL https://openreview.net/forum?id=iHXV8UGYyL.

Fernando Gallego, Guillermo López-García, Luis Gasco-Sánchez, Martin Krallinger, and Francisco J. Veredas. Clinlinker: Medical entity linking of clinical concept mentions in spanish. In Leonardo Franco, Clélia de Mulatier, Maciej Paszynski, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot (eds.), *Computational Science – ICCS 2024*, pp. 266–280, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-63775-9. URL https://link.springer.com/chapter/10.1007/978-3-031-63775-9_19.

Octavian-Eugen Ganea and Thomas Hofmann. Deep joint entity disambiguation with local neural attention. In Martha Palmer, Rebecca Hwa, and Sebastian Riedel (eds.), *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2619–2629, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1277. URL https://aclanthology.org/D17-1277/.

Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. Learning dense representations for entity retrieval. In Mohit Bansal and Aline Villavicencio (eds.), *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pp. 528–537, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/K19-1049. URL https://aclanthology.org/K19-1049/.

Carlos Gonzalez-Gamella, Daniel Sanchez-Santolaya, Francisco Moraleda-Moreno, and Jacobo Chaquet-Ulldemolins. When to rerank? adaptive reranking decisions for efficient information retrieval. In *Proceedings of the 28th International Conference on Enterprise Information Systems - Volume 1: ICEIS*, pp. 545–556. INSTICC, SciTePress, 2026. ISBN 978-989-758-834-1. doi: 10.5220/0014803000004018. URL https://www.scitepress.org/Link.aspx?doi=10.5220/0014803000004018.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5–6):602–610, 2005. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2005.06.042. URL https://www.sciencedirect.com/science/article/pii/S0893608005001206. IJCNN 2005.

Zhaochen Guo and Denilson Barbosa. Robust named entity disambiguation with random walks. *Semantic Web*, 9(4):459–479, 2018. doi: 10.3233/SW-170273. URL https://journals.sagepub.com/doi/abs/10.3233/SW-170273.

Bahareh Harandizadeh and Sameer Singh. Tweeki: Linking named entities on Twitter to a knowledge graph. In Wei Xu, Alan Ritter, Tim Baldwin, and Afshin Rahimi (eds.), *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pp. 222–231, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.wnut-1.29. URL https://aclanthology.org/2020.wnut-1.29/.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTav3: Improving deBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sE7-XhLxHA.

Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In Regina Barzilay and Mark Johnson (eds.), *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 782–792, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics. URL https://aclanthology.org/D11-1072/.

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

New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450311564. doi: 10.1145/2396761.2396832. URL https://doi.org/10.1145/2396761.2396832.

Mohanna Hoveyda, Arjen Vries, Faegheh Hasibi, and Maarten de Rijke. Real world conversational entity linking requires more than zero-shots. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 13938–13946, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.829. URL https://aclanthology.org/2024.findings-acl.829/.

Yotam Intrator, Regev Cohen, Ori Kelner, Roman Goldenberg, Ehud Rivlin, and Daniel Freedman. Streamlining conformal information retrieval via score refinement. In Michael Schlichtkrull, Yulong Chen, Chenxi Whitehouse, Zhenyun Deng, Mubashara Akhtar, Rami Aly, Zhijiang Guo, Christos Christodoulopoulos, Oana Cocarascu, Arpit Mittal, James Thorne, and Andreas Vlachos (eds.), *Proceedings of the Seventh Fact Extraction and VERification Workshop (FEVER)*, pp. 186–191, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.fever-1.22. URL https://aclanthology.org/2024.fever-1.22/.

Kevin Kasa, Zhiyu Zhang, Heng Yang, and Graham W. Taylor. Adapting prediction sets to distribution shifts without labels. In Silvia Chiappa and Sara Magliacane (eds.), *Proceedings of the Forty-first Conference on Uncertainty in Artificial Intelligence*, volume 286 of *Proceedings of Machine Learning Research*, pp. 1990–2010. PMLR, 21–25 Jul 2025. URL https://proceedings.mlr.press/v286/kasa25a.html.

Philipp Koehn. Statistical significance tests for machine translation evaluation. In Dekang Lin and Dekai Wu (eds.), *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pp. 388–395, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-3250/.

Nikolaos Kolitsas, Octavian-Eugen Ganea, and Thomas Hofmann. End-to-end neural entity linking. In Anna Korhonen and Ivan Titov (eds.), *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pp. 519–529, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/K18-1050. URL https://aclanthology.org/K18-1050/.

Md Tahmid Rahman Laskar, Cheng Chen, Aliaksandr Martsinovich, Jonathan Johnston, Xue-Yong Fu, Shashi Bhushan Tn, and Simon Corston-Oliver. BLINK with Elasticsearch for efficient entity linking in business conversations. In Anastassia Loukina, Rashmi Gangadharaiah, and Bonan Min (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pp. 344–352, Hybrid: Seattle, Washington + Online, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-industry.38. URL https://aclanthology.org/2022.naacl-industry.38/.

Minghan Li, Xinyu Zhang, Ji Xin, Hongyang Zhang, and Jimmy Lin. Certified error control of candidate set pruning for two-stage relevance ranking. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 333–345, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.23. URL https://aclanthology.org/2022.emnlp-main.23/.

Shuo Li, Sangdon Park, Insup Lee, and Osbert Bastani. TRAQ: Trustworthy retrieval augmented question answering via conformal prediction. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3799–3821, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.210. URL https://aclanthology.org/2024.naacl-long.210/.

Yi Liu, Yuan Tian, Jianxun Lian, Xinlong Wang, Yanan Cao, Fang Fang, Wen Zhang, Haizhen Huang, Weiwei Deng, and Qi Zhang. Towards better entity linking with multi-view enhanced distillation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of*

the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 9729–9743, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.542. URL https://aclanthology.org/2023.acl-long.542/.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. Zero-shot entity linking by reading entity descriptions. In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3449–3460, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1335. URL https://aclanthology.org/P19-1335/.

Marcel Milich and Alan Akbik. ZELDA: A comprehensive benchmark for supervised entity disambiguation. In Andreas Vlachos and Isabelle Augenstein (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 2061–2072, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.151. URL https://aclanthology.org/2023.eacl-main.151/.

Nicholas Monath, Manzil Zaheer, Kelsey Allen, and Andrew McCallum. Improving dual-encoder training through dynamic indexes for negative mining. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de Meent (eds.), *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pp. 9308–9330. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/monath23a.html.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20275–20321, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1025. URL https://aclanthology.org/2025.emnlp-main.1025/.

Hibiki Nakatani, Hiroki Teranishi, Shohei Higashiyama, Yuya Sawada, Hiroki Ouchi, and Taro Watanabe. A text embedding model with contrastive example mining for point-of-interest geocoding. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 7279–7291, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.486/.

Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, Aldo Gangemi, Darío Garigliotti, and Roberto Navigli. Open knowledge extraction challenge. In Fabien Gandon, Elena Cabrio, Milan Stankovic, and Antoine Zimmermann (eds.), *Semantic Web Evaluation Challenges*, pp. 3–15, Cham, 2015. Springer International Publishing. ISBN 978-3-319-25518-7. doi: 10.1007/978-3-319-25518-7_1. URL https://doi.org/10.1007/978-3-319-25518-7_1.

Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, Aldo Gangemi, Robert Meusel, and Heiko Paulheim. The second open knowledge extraction challenge. In Harald Sack, Stefan Dietze, Anna Tordai, and Christoph Lange (eds.), *Semantic Web Challenges*, pp. 3–16, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46565-4. doi: 10.1007/978-3-319-46565-4_1. URL https://doi.org/10.1007/978-3-319-46565-4_1.

Vasco Pearson, Jean V. Alves, Jacopo Bono, Mario A. T. Figueiredo, and Pedro Bizarro. Uncertainty-aware systems for human-AI collaboration. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=PiRYCyNBqQ.

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. KILT: a benchmark for knowledge intensive language

tasks. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2523–2544, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.200. URL https://aclanthology.org/2021.naacl-main.200/.

Vera Provatorova, Samarth Bhargav, Svitlana Vakulenko, and Evangelos Kanoulas. Robustness evaluation of entity disambiguation using prior probes: the case of entity overshadowing. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10501–10510, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.820. URL https://aclanthology.org/2021.emnlp-main.820/.

Michael Röder, Ricardo Usbeck, Sebastian Hellmann, Daniel Gerber, and Andreas Both. N³ - a collection of datasets for named entity recognition and disambiguation in the NLP interchange format. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (eds.), *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pp. 3529–3533, Reykjavik, Iceland, May 2014. European Language Resources Association (ELRA). URL https://aclanthology.org/L14-1662/.

Michael Röder, Ricardo Usbeck, and Axel-Cyrille Ngonga Ngomo. Gerbil – benchmarking named entity recognition and linking consistently. *Semantic Web*, 9(5):605–625, 2018. doi: 10.3233/SW-170286. URL https://journals.sagepub.com/doi/abs/10.3233/SW-170286.

Javier Sanz-Cruzado and Jake Lever. Accelerating cross-encoders in biomedical entity linking. In Dina Demner-Fushman, Sophia Ananiadou, Makoto Miwa, and Junichi Tsujii (eds.), *Proceedings of the 24th Workshop on Biomedical Language Processing*, pp. 136–147, Vienna, Austria, August 2025. Association for Computational Linguistics. ISBN 979-8-89176-275-6. doi: 10.18653/v1/2025.bionlp-1.13. URL https://aclanthology.org/2025.bionlp-1.13/.

Yuya Sawada, Tsuyoshi Fujita, Yusuke Sakai, and Taro Watanabe. entity-linkings: A unified library for entity linking. In Danilo Croce, Jochen Leidner, and Nafise Sadat Moosavi (eds.), *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 591–601, Rabat, Morocco, March 2026. Association for Computational Linguistics. ISBN 979-8-89176-382-1. doi: 10.18653/v1/2026.eacl-demo.42. URL https://aclanthology.org/2026.eacl-demo.42/.

Özge Sevgili, Artem Shelmanov, Mikhail Arkhipov, Alexander Panchenko, and Chris Biemann. Neural entity linking: A survey of models based on deep learning. *Semantic Web*, 13(3):527–570, 2022. doi: 10.3233/SW-222986. URL https://journals.sagepub.com/doi/abs/10.3233/SW-222986.

Hassan Shavarani and Anoop Sarkar. SpEL: Structured prediction for entity linking. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 11123–11137, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.686. URL https://aclanthology.org/2023.emnlp-main.686/.

Jonghyun Song, Cheyon Jin, Wenlong Zhao, Andrew McCallum, and Jay-Yoon Lee. Comparing neighbors together makes it easy: Jointly comparing multiple candidates for efficient and effective retrieval. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22255–22269, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1242. URL https://aclanthology.org/2024.emnlp-main.1242/.

Chihiro Taguchi, Seiji Maekawa, and Nikita Bhutani. Efficient context selection for long-context QA: No tuning, no iteration, just adaptive-k. In Christos Christodoulopoulos, Tanmoy

Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20105–20130, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.1017. URL https://aclanthology.org/2025.emnlp-main.1017/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer New York, NY, 2005. doi: 10.1007/b106715. URL https://doi.org/10.1007/b106715.

Junxiong Wang, Ali Mousavi, Omar Attia, Ronak Pradeep, Saloni Potdar, Alexander Rush, Umar Farooq Minhas, and Yunyao Li. Entity disambiguation via fusion entity decoding. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6524–6536, Mexico City, Mexico, June 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.363. URL https://aclanthology.org/2024.naacl-long.363/.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training, 2024b. URL https://arxiv.org/abs/2212.03533.

Sijia Wang and Lifu Huang. Debate as optimization: Adaptive conformal prediction and diverse retrieval for event extraction. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 16422–16435, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.958. URL https://aclanthology.org/2024.findings-emnlp.958/.

Chen Wu, Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, and Xueqi Cheng. Learning to truncate ranked lists for information retrieval. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4453–4461, May 2021. ISSN 2159-5399. doi: 10.1609/aaai.v35i5.16572. URL http://dx.doi.org/10.1609/aaai.v35i5.16572.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. Scalable zero-shot entity linking with dense entity retrieval. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6397–6407, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.519. URL https://aclanthology.org/2020.emnlp-main.519/.

Zhenran Xu, Yulin Chen, and Baotian Hu. Improving biomedical entity linking with cross-entity interaction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13869–13877, 2023. ISSN 2159-5399. doi: 10.1609/aaai.v37i11.26624. URL http://dx.doi.org/10.1609/aaai.v37i11.26624.

Christophe Ye and Cassie S. Mitchell. LLM as entity disambiguator for biomedical entity-linking. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 301–312, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-252-7. doi: 10.18653/v1/2025.acl-short.25. URL https://aclanthology.org/2025.acl-short.25/.

Wenzheng Zhang and Karl Stratos. Understanding hard negatives in noise contrastive estimation. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1090–1101, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.86. URL https://aclanthology.org/2021.naacl-main.86/.

Wenzheng Zhang, Wenyue Hua, and Karl Stratos. EntQA: Entity linking as question answering. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=US2rTP5nm_.

A Limitations and Future Work

Design of Nonconformity Score Our method defines nonconformity for each candidate as its candidate generator score gap from the top-ranked entity, using this relative separation to determine the pruning boundary. However, the score does not reflect the absolute score level or how likely the top-ranked entity is to be correct. Consequently, candidate sets with identical gaps are treated similarly even when their overall scores differ, and a weak or incorrect top-ranked entity may serve as an unreliable reference. Nonetheless, as shown in Section 5 and Appendix D.2, our method achieves improvements in both efficiency and accuracy compared with the baselines and the variants that use straightforward normalization of score gaps or the candidate score distribution in the computation of nonconformity scores. While these results demonstrate the effectiveness of the proposed method, further improvements by considering the reliability of the top-ranked entity or the overall distribution of candidate scores remain important directions for future work.

Threshold Selection Strategy Our method derives pruning thresholds from empirical quantiles of the gold-entity nonconformity scores on the development set and selects α according to end-to-end development accuracy. This procedure is inspired by conformal prediction, but it represents only one possible strategy for selecting a pruning threshold when formal coverage is not required. Other development-set threshold selection strategies could include directly tuning the score-gap threshold; selecting a threshold subject to a target candidate budget, latency, or candidate-coverage constraint; or optimizing a combined accuracy–efficiency objective. Nevertheless, Section 6.1 shows the robustness of our method to hyperparameter changes, suggesting that the small changes in the threshold values that may be introduced by alternative threshold selection strategies are unlikely to considerably influence the overall efficiency gains. Thus, this work focuses on demonstrating the practical utility of our adaptive candidate pruning using a single threshold selection strategy, and leaves a systematic comparison of alternative selection strategies for future work.

Extension to End-to-End Models This study focuses on the widely adopted two-stage ED pipeline, which comprises a candidate generator and a candidate reranker, and introduces a lightweight method for adaptive candidate pruning. On the other hand, some studies explore end-to-end EL architectures that jointly identify mention spans and disambiguate them (Ayoola et al., 2022; Shavarani & Sarkar, 2023). Although our experiments focus on two-stage pipelines, our method could also prune unlikely entities in end-to-end models that explicitly use external candidate lists, such as lists constructed from entity priors estimated from Wikipedia hyperlinks (Ganea & Hofmann, 2017). For end-to-end models without an explicit candidate list, extending our approach would require identifying an internal stage at which unlikely entity hypotheses could be discarded before computationally expensive processing. We leave the adaptation and evaluation of our approach for end-to-end EL models to future work.

B Detailed Experimental Setup

B.1 Dataset

ZELDA standardizes nine evaluation datasets drawn from multiple text domains. The nine datasets are AIDA-B (Hoffart et al., 2011), TWEEKI (Harandizadeh & Singh, 2020),

Table 4: Hyperparameters for candidate generator and reranker. – indicates not applicable.

Parameters	Generator Stage 1	Generator Stage 2	Reranker
Seed	42	42	42
Training epochs	2	2	1
Hard negatives	0	10	–
Training batch size	128	32	8
Candidates	–	–	30
Max token length (context)	128	128	128
Max token length (candidate)	50	50	50
Learning rate	1e-5	1e-5	2e-5
Gradient accumulation steps	1	4	4
Scheduler	linear	linear	linear
Optimizer	AdamW	AdamW	AdamW
Warmup ratio	0.10	0.10	0.10
Weight decay	0	0	0.01
Max grad norm	0	0	0
Adam beta	$[0.9, 0.98]$	$[0.9, 0.98]$	$[0.9, 0.98]$
Adam epsilon	1e-6	1e-6	1e-6

REDDIT-POSTS, REDDIT-COMMENTS (REDDIT-COMM) (Botzer et al., 2021), WNED-WIKI, WNED-CWEB (Guo & Barbosa, 2018), SHADOWLINK-TOP (SLINKS-TOP), SHADOWLINK-SHADOW (SLINKS-SHADOW), and SHADOWLINK-TAIL (SLINKS-TAIL) (Provatorova et al., 2021).

AIDA-B is the test split of AIDA and consists of manually annotated Reuters news articles. TWEEKI contains manually annotated tweets and evaluates ED on short, noisy text from social media. REDDIT-POSTS and REDDIT-COMMENTS contain highly rated posts and comments from Reddit, respectively, and ZELDA uses their gold annotations, on which all annotators agreed. WNED-WIKI and WNED-CWEB comprise Wikipedia articles and general web pages, respectively, and both also provide document-level difficulty annotations. The remaining three datasets, SHADOWLINK-TOP, SHADOWLINK-SHADOW, and SHADOWLINK-TAIL, are designed to evaluate different forms of mention ambiguity. In SHADOWLINK-TOP, the gold entity is the most frequent sense of the mention; in SHADOWLINK-SHADOW, the gold entity is a less prominent sense overshadowed by a more popular entity; and in SHADOWLINK-TAIL, the gold entities are rare in Wikipedia.

B.2 Model Training

We initialize the mention and entity encoders of the dual-encoder separately from BERT–base (110M) (Devlin et al., 2019). During training, we compute the probability that an entity e is the gold entity for a mention m based on their dual-encoder similarity score $s_{\mathrm{de}}(m, e)$. To reduce computational cost, we approximate this probability as:

$$p(e \mid m) \simeq \frac{\exp(s_{\mathrm{de}}(m, e))}{\sum_{e' \in \mathcal{E}_{\mathrm{norm}}} \exp(s_{\mathrm{de}}(m, e'))}, \tag{15}$$

where e' ranges over a normalization set $\mathcal{E}_{\mathrm{norm}} \subset \mathcal{E}$ that contains e and replaces the full KB entity set $\mathcal{E}$ in the denominator. For a mini-batch of mentions $\mathcal{B}$, let e_m^+ denote the gold entity for m. The normalization set is $\mathcal{E}_{\mathrm{norm}} = \mathcal{E}_{\mathcal{B}} \cup \mathcal{E}_{\mathrm{hard}}$, where $\mathcal{E}_{\mathcal{B}}$ contains the distinct gold entities of the mentions in the current mini-batch and $\mathcal{E}_{\mathrm{hard}}$ contains non-gold entities selected through hard-negative mining. We minimize the following loss:

$$\mathcal{L}_{\mathrm{de}} = -\frac{1}{|\mathcal{B}|} \sum_{m \in \mathcal{B}} \log p(e_m^+ \mid m). \tag{16}$$

Training proceeds in two stages. In the first stage, we set $\mathcal{E}_{\mathrm{hard}} = \emptyset$, so the gold entities of the other examples in the same mini-batch serve as negatives. We then use the resulting model to select the ten

Table 5: Dual-encoder pipeline performance on the ZELDA test datasets for different initial candidate set sizes. Acc. is macro-average test accuracy. Inference time covers the complete ZELDA test set. Avg. Cands is the micro-average number of candidates passed to the cross-encoder. Down arrows indicate lower is better. For Acc. and each test set, bold and underlined values indicate the best and second-best results, respectively, excluding Oracle. For inference time and Avg. Cands, bold and underlined values indicate the best and second-best results, respectively, excluding Oracle and No-Reranking.

	Acc.	Inference Time (s) ↓	Avg. Cands ↓	AIDA-B	TWEEKI	REDDIT-POSTS	REDDIT-COMM.	WNED-CWEB	WNED-WIKI	SLINKS-TAIL	SLINKS-SHADOW	SLINKS-TOP
Candidate set size: $K = 10$												
Oracle	86.6	$325_{\pm 3.7}$	1.29	89.1	83.4	95.5	92.2	75.3	93.1	99.6	74.7	77.0
No-Reranking	81.0	$271_{\pm 0.6}$	0.00	83.5	77.9	92.3	87.1	70.5	90.2	99.0	61.9	66.2
No-Pruning	82.8	$747_{\pm 12.5}$	10.0	83.8	79.2	93.0	90.0	71.7	90.9	99.3	**69.0**	68.3
Tuned-Fixed-k	82.8	$747_{\pm 12.5}$	10.0	83.8	79.2	93.0	90.0	71.7	90.9	99.3	**69.0**	68.3
Adaptive-k	82.8	$480_{\pm 2.0}$	4.35	84.3	80.0	93.2	90.3	71.7	90.8	**99.4**	68.3	67.7
Ours	**83.3**	**329**$_{\pm 1.9}$	**1.30**	**85.4**	**80.3**	**93.3**	**90.6**	**72.3**	**91.1**	**99.4**	67.8	**69.6**
Candidate set size: $K = 30$												
Oracle	87.2	$437_{\pm 3.6}$	2.84	89.7	83.7	95.6	92.5	76.1	93.3	99.6	75.8	78.2
No-Reranking	81.0	$271_{\pm 0.6}$	0.00	83.5	77.9	92.3	87.1	70.5	90.2	99.0	61.9	66.2
No-Pruning	82.7	$1732_{\pm 16.3}$	30.0	83.2	79.1	93.0	89.8	71.9	90.8	99.3	**69.7**	67.8
Tuned-Fixed-k	82.8	$747_{\pm 12.5}$	10.0	83.8	79.2	93.0	90.0	71.7	90.9	99.3	69.0	68.3
Adaptive-k	82.9	$508_{\pm 1.7}$	4.38	84.3	80.0	**93.2**	90.3	71.7	90.8	**99.4**	68.4	67.6
Ours	**83.5**	**380**$_{\pm 3.3}$	**1.79**	**85.5**	**80.2**	**93.2**	**90.8**	**72.6**	**91.2**	**99.4**	68.1	**70.4**
Candidate set size: $K = 50$												
Oracle	87.2	$526_{\pm 2.6}$	4.15	89.8	83.8	95.6	92.5	76.3	93.4	99.6	75.9	78.3
No-Reranking	81.0	$271_{\pm 0.6}$	0.00	83.5	77.9	92.3	87.1	70.5	90.2	99.0	61.9	66.2
No-Pruning	82.6	$2744_{\pm 16.1}$	50.0	82.9	78.6	93.0	89.7	71.8	90.8	99.3	**69.4**	67.8
Tuned-Fixed-k	82.8	$747_{\pm 12.5}$	10.0	83.8	79.2	93.0	90.0	71.7	90.9	99.3	69.0	68.3
Adaptive-k	82.9	$531_{\pm 0.3}$	4.38	84.3	80.0	**93.2**	90.3	71.7	90.8	**99.4**	68.4	67.6
Ours	**83.5**	**419**$_{\pm 2.3}$	**2.01**	**85.5**	**80.2**	**93.2**	**90.8**	**72.6**	**91.2**	**99.4**	68.1	**70.4**
Candidate set size: $K = 100$												
Oracle	87.4	$750_{\pm 1.5}$	6.96	89.9	84.1	95.7	92.5	76.5	93.4	99.6	76.0	78.9
No-Reranking	81.0	$271_{\pm 0.6}$	0.00	83.5	77.9	92.3	87.1	70.5	90.2	99.0	61.9	66.2
No-Pruning	82.5	$5307_{\pm 22.9}$	100	82.7	78.8	93.0	89.3	71.6	90.7	99.3	**69.4**	67.9
Tuned-Fixed-k	82.8	$747_{\pm 12.5}$	10.0	83.8	79.2	93.0	90.0	71.7	90.9	99.3	69.0	68.3
Adaptive-k	82.9	$602_{\pm 0.9}$	4.39	84.3	80.0	93.2	90.3	71.7	90.8	**99.4**	68.4	67.6
Ours	**83.5**	**497**$_{\pm 1.3}$	**2.16**	**85.6**	**80.2**	**93.3**	**90.8**	**72.6**	**91.2**	**99.4**	68.1	**70.5**

highest-scoring non-gold entities for every mention in the training set as hard negatives (Gillick et al., 2019). In the second stage, we initialize the model using the parameters of the model constructed in the first stage and add the selected hard negative entities to $\mathcal{E}_{\text{hard}}$. In both stages, we retain the checkpoint with the lowest loss on the development set.

After the second stage, we use the resulting dual-encoder to create the cross-encoder training sets. For each mention, we retain the gold entity and the 29 highest-scoring non-gold entities, yielding 30 candidates in total. Given the resulting candidate set $\mathcal{C}^{\text{trn}}(m)$, we optimize the cross-encoder using

$$\mathcal{L}_{\text{ce}} = -\frac{1}{|\mathcal{B}|} \sum_{m \in \mathcal{B}} \log \frac{\exp(s_{\text{ce}}(m, e_m^+))}{\sum_{e \in \mathcal{C}^{\text{trn}}(m)} \exp(s_{\text{ce}}(m, e))}. \tag{17}$$

We retain the cross-encoder checkpoint with the highest accuracy on the development set. Table 4 aggregates the hyperparameters for model training. Training took approximately 38 hours and 57 hours for the dual-encoder and cross-encoder, respectively, on a single NVIDIA RTX A6000 GPU with 48 GB of memory.

B.3 Inference Time Measurement

We measure inference time on a single NVIDIA RTX 3090 GPU with 24 GB of memory. Dual-encoder and cross-encoder batch sizes are set to 256 and 8, respectively. Before measurement, we perform one warm-up run on the first 32 test examples using candidate set size $K = 30$. We then run each method three times over the complete ZELDA test set and report the mean end-to-end inference

Table 6: E5 pipeline performance on the ZELDA test datasets for different initial candidate set sizes. Metric definitions and formatting conventions are the same as in Table 5.

	Acc.	Inference Time (s) ↓	Avg. Cands ↓	AIDA-B	TWEEKI	REDDIT-POSTS	REDDIT-COMM.	WNED-CWEB	WNED-WIKI	SLINKS-TAIL	SLINKS-SHADOW	SLINKS-TOP
Candidate set size: $K = 10$												
Oracle	87.5	$321_{\pm 1.1}$	1.12	89.7	85.5	94.9	93.7	76.6	93.3	99.6	76.8	77.4
No-Reranking	82.9	$273_{\pm 2.4}$	0.00	84.2	82.6	91.9	91.1	72.3	90.6	99.1	66.9	67.6
No-Pruning	82.8	$740_{\pm 1.3}$	10.0	84.4	79.9	91.8	90.3	71.8	**91.0**	**99.2**	68.5	68.3
Tuned-Fixed-k	82.8	$740_{\pm 1.3}$	10.0	84.4	79.9	91.8	90.3	71.8	**91.0**	**99.2**	68.5	68.3
Adaptive-k	83.6	**298**$_{\pm 3.3}$	**0.59**	85.8	81.9	93.0	91.7	73.0	90.9	**99.2**	68.5	68.8
Ours	**84.0**	$309_{\pm 2.0}$	0.85	**86.4**	**83.0**	**93.2**	**92.0**	**73.4**	**91.0**	99.1	**68.8**	**69.5**
Candidate set size: $K = 30$												
Oracle	87.7	$412_{\pm 0.8}$	2.36	90.1	85.5	95.2	94.0	77.0	93.4	99.6	77.1	77.8
No-Reranking	82.9	$273_{\pm 2.4}$	0.00	84.2	82.6	91.9	91.1	72.3	90.6	99.1	66.9	67.6
No-Pruning	82.4	$1726_{\pm 6.8}$	30.0	83.5	79.3	91.6	90.1	71.5	90.9	**99.2**	68.0	67.4
Tuned-Fixed-k	82.8	$740_{\pm 1.3}$	10.0	84.4	79.9	91.8	90.3	71.8	**91.0**	**99.2**	68.5	68.3
Adaptive-k	83.6	**323**$_{\pm 2.5}$	**0.63**	85.8	81.9	**93.0**	91.7	73.0	90.9	**99.2**	68.3	68.9
Ours	**84.1**	$342_{\pm 0.4}$	0.99	**86.3**	**83.1**	92.9	**92.5**	**73.5**	**91.0**	99.1	**68.9**	**69.7**
Candidate set size: $K = 50$												
Oracle	87.8	$493_{\pm 1.3}$	3.40	90.3	85.5	95.2	94.0	77.0	93.4	99.6	77.4	77.9
No-Reranking	82.9	$273_{\pm 2.4}$	0.00	84.2	82.6	91.9	91.1	72.3	90.6	99.1	66.9	67.6
No-Pruning	82.3	$2738_{\pm 13.3}$	50.0	83.3	79.3	91.5	90.1	71.2	**91.0**	**99.2**	68.3	66.9
Tuned-Fixed-k	82.8	$740_{\pm 1.3}$	10.0	84.4	79.9	91.8	90.3	71.8	**91.0**	**99.2**	68.5	68.3
Adaptive-k	83.6	**348**$_{\pm 2.5}$	**0.64**	85.8	81.9	**93.0**	91.7	73.0	90.9	**99.2**	68.3	68.9
Ours	**84.1**	$368_{\pm 0.0}$	1.01	**86.3**	**83.1**	92.9	**92.5**	**73.5**	**91.0**	99.1	**68.9**	**69.7**
Candidate set size: $K = 100$												
Oracle	87.9	$681_{\pm 0.3}$	5.61	90.3	85.5	95.2	94.0	77.2	93.5	99.6	77.5	78.1
No-Reranking	82.9	$273_{\pm 2.4}$	0.00	84.2	82.6	91.9	91.1	72.3	90.6	99.1	66.9	67.6
No-Pruning	82.2	$5321_{\pm 32.2}$	100	83.0	79.2	91.5	89.8	71.2	90.9	**99.2**	68.1	66.7
Tuned-Fixed-k	82.8	$740_{\pm 1.3}$	10.0	84.4	79.9	91.8	90.3	71.8	**91.0**	**99.2**	68.5	68.3
Adaptive-k	83.6	**416**$_{\pm 3.4}$	**0.65**	85.8	81.9	**93.0**	91.7	73.0	90.9	**99.2**	68.3	68.9
Ours	**84.1**	$435_{\pm 2.2}$	1.03	**86.3**	**83.1**	92.9	**92.5**	**73.5**	**91.0**	99.1	**68.9**	**69.7**

time. The measured interval includes candidate generation by the dual-encoder, candidate pruning, and cross-encoder reranking, but excludes model and data loading, pruning threshold selection, and metric computation. As described in Section 4, mentions for which only one candidate is retained after pruning bypass the cross-encoder. We use the entity-linkings library (Sawada et al., 2026) released under the MIT License for the data download, model training, and evaluation.

C Supplemental Experiments

C.1 Text-Embedding Model Pipeline and Various Candidate Set Sizes

We conduct two supplemental experiments to examine whether the proposed pruning method generalizes across independently trained ED pipelines and initial candidate set sizes. We refer to the system used in the main experiments described in Section 4 as the *dual-encoder pipeline*. Its candidate generator uses separately parameterized mention and entity encoders, and its cross-encoder is trained on candidate sets produced by this generator. We additionally construct an *E5 pipeline*. Its candidate generator uses a single text-embedding model, initialized with E5-base (110M)[2] (Wang et al., 2024b), whose parameters are shared by the mention encoder E_m and entity encoder E_e. We then train a separate cross-encoder on candidate sets produced by the E5 model. Accordingly, the E5 pipeline is a distinct, separately trained two-stage ED system, rather than a version of the original pipeline in which only the candidate generator is replaced at inference time. The two pipelines use the same cross-encoder architecture and otherwise follow the same training, threshold-selection, inference, and evaluation settings. Training of the candidate generator and candidate reranker in the E5 pipeline took approximately 40 hours and 54 hours, respectively, on a single NVIDIA RTX A6000 GPU with 48 GB of memory.

[2] https://huggingface.co/intfloat/e5-base

We evaluate both pipelines with initial candidate set sizes $K \in \{10, 30, 50, 100\}$. This range includes the main setting of $K = 30$ as well as smaller and larger candidate pools, allowing us to test whether the method remains effective under different configurations of candidate generation.

Tables 5 and 6 report the results for the dual-encoder and E5 pipelines, respectively. Across all values of K and both independently trained pipelines, our method achieves the highest macro-average accuracy among the non-oracle methods while passing only a small number of candidates to the cross-encoder.

For the dual-encoder pipeline, our method consistently achieves higher macro-average accuracy than No-Pruning, Tuned-Fixed-k, and Adaptive-k while passing only a small candidate set to the cross-encoder. As K increases, the average number of reranked candidates grows only modestly, whereas the cost of No-Pruning grows directly with the initial candidate set size. The accuracy of our method also remains stable across the evaluated settings. Thus, both its efficiency and effectiveness are robust to substantial changes in the size of the initial candidate set.

The E5 pipeline exhibits a similar pattern. Our method drastically reduces the candidate set relative to No-Pruning and Tuned-Fixed-k while improving the macro-average accuracy over both baselines. It also achieves higher accuracy than Adaptive-k, although Adaptive-k retains slightly fewer candidates in this pipeline. However, the additional pruning by Adaptive-k provides only a modest reduction in inference time, whereas our method consistently achieves higher accuracy. This suggests that retaining a small number of additional candidates preserves useful alternatives, resulting in a more favorable trade-off between efficiency and accuracy.

Overall, the efficiency and accuracy gains are consistent across both the dual-encoder and E5 pipelines and all evaluated values of K. These results show that the gains persist even when the pipeline architecture and the initial candidate set size vary.

C.2 Evaluation on the GERBIL Benchmark

To assess our method with a larger KB and additional evaluation domains, we conduct experiments on the GERBIL benchmark (Röder et al., 2018) using the KILT knowledge base (KB) (Petroni et al., 2021). We train separate dual-encoder and E5 pipelines on approximately 8.6 million examples derived from Wikipedia hyperlinks, with a KB of approximately 5.9 million Wikipedia entities, which is more than seven times larger than the KB of ZELDA; KILT provides both resources. Of the $\sim$8.6 million training examples, we use 10,000 examples as the development set. Training the dual-encoder pipeline took approximately 62 hours for the candidate generator and 78 hours for the reranker, while training the E5 pipeline took approximately 77 and 78 hours, respectively. Each pipeline was trained on two NVIDIA RTX A6000 GPUs with 48 GB of memory per GPU.

Following prior ED evaluations (Kolitsas et al., 2018; Wang et al., 2024a; Zhang et al., 2022), we use eight GERBIL test datasets: AIDA-B (Hoffart et al., 2011), MSNBC (Cucerzan, 2007), DERCZYNSKI (DER) (Derczynski et al., 2015), KORE50 (Hoffart et al., 2012), N3-REUTERS-128 (N3-128) and N3-RSS-500 (N3-500) (Röder et al., 2014), and OKE-2015 and OKE-2016 (Nuzzolese et al., 2015; 2016). AIDA-B, MSNBC, and N3-128 contain news text; DER contains tweets; N3-500 contains RSS feed text. KORE50 comprises short sentences with ambiguous entity mentions, while the OKE datasets contain short, annotated sentences from the Open Knowledge Extraction challenges. For both the dual-encoder and E5 pipelines, we search over $\alpha \in \{0.001, 0.005, 0.01, 0.02, \ldots, 0.25\}$ when selecting the pruning threshold. We otherwise follow the training, threshold-selection, baseline, and evaluation procedures of the main experiment, and evaluate the same candidate set sizes $K \in \{10, 30, 50, 100\}$ as in the supplemental experiment in Appendix C.1.

Tables 7 and 8 show the results for the dual-encoder and E5 pipelines, respectively. For the dual-encoder pipeline, the proposed method sends fewer candidates to the cross-encoder while improving or maintaining end-to-end accuracy relative to the baselines in most configurations. It also generally requires less inference time than the other reranking methods. These results show that its efficiency and accuracy gains largely persist with the larger KILT KB and across different initial candidate set sizes.

For the E5 pipeline, the proposed method improves the end-to-end accuracy over all the baselines for every value of K, although it passes more candidates to the cross-encoder than Adaptive-k. Importantly, as the initial candidate set grows, the average candidate count after Oracle pruning

Table 7: Dual-encoder pipeline performance on GERBIL with the KILT KB for different initial candidate set sizes. Metric definitions and formatting conventions are the same as in Table 5.

	Acc.	Inference Time (s) ↓	Avg. Cands ↓	AIDA-B	MSNBC	DER	KORE50	N3-R128	N3-R500	OKE-2015	OKE-2016
Candidate set size: $K = 10$											
Oracle	49.9	$345_{\pm 0.4}$	3.52	78.9	53.4	41.1	38.5	45.9	29.7	51.9	59.7
No-Reranking	47.2	$291_{\pm 0.8}$	0.00	74.1	**51.5**	39.4	34.5	43.0	28.5	49.6	56.8
No-Pruning	47.8	$451_{\pm 0.8}$	10.0	74.6	50.3	**39.7**	36.5	44.2	**28.7**	50.1	58.5
Tuned-Fixed-k	47.9	$436_{\pm 0.2}$	9.00	74.6	50.3	**39.7**	37.2	44.3	**28.7**	50.1	58.5
Adaptive-k	48.0	$353_{\pm 0.5}$	3.51	74.7	50.9	**39.7**	37.2	44.8	28.5	49.8	58.5
Ours	**48.3**	$\mathbf{345}_{\pm 1.2}$	**3.24**	**75.1**	51.0	**39.7**	**37.8**	**44.9**	28.6	**50.2**	**58.8**
Candidate set size: $K = 30$											
Oracle	50.0	$439_{\pm 1.2}$	9.98	79.0	53.5	41.1	38.5	46.4	29.8	52.3	59.7
No-Reranking	47.2	$291_{\pm 0.8}$	0.00	74.1	**51.5**	39.4	34.5	43.0	28.5	49.6	56.8
No-Pruning	47.7	$759_{\pm 1.0}$	30.0	74.3	50.1	39.7	35.8	44.1	**28.8**	50.1	58.5
Tuned-Fixed-k	47.7	$744_{\pm 0.5}$	29.0	74.4	50.1	39.7	35.8	44.1	**28.8**	50.2	58.5
Adaptive-k	47.8	$402_{\pm 1.8}$	6.59	74.8	50.3	39.7	36.5	44.2	28.7	49.9	58.5
Ours	**48.4**	$\mathbf{374}_{\pm 1.2}$	**5.11**	**75.2**	**51.5**	**40.1**	**37.8**	**44.5**	28.7	**50.7**	**58.8**
Candidate set size: $K = 50$											
Oracle	50.1	$531_{\pm 0.5}$	16.3	79.0	53.5	41.1	38.5	46.5	29.8	52.3	59.7
No-Reranking	47.2	$291_{\pm 0.8}$	0.00	74.1	**51.5**	39.4	34.5	43.0	28.5	49.6	56.8
No-Pruning	47.6	$1067_{\pm 1.9}$	50.0	74.3	50.1	39.7	35.1	44.0	**28.8**	49.9	58.5
Tuned-Fixed-k	47.7	$744_{\pm 0.5}$	29.0	74.4	50.1	39.7	35.8	44.1	**28.8**	50.2	58.5
Adaptive-k	47.8	$402_{\pm 1.9}$	6.59	74.8	50.3	39.7	36.5	44.2	28.7	49.9	58.5
Ours	**48.3**	$\mathbf{392}_{\pm 0.9}$	**6.13**	**75.2**	**51.5**	**40.1**	**37.2**	**44.5**	28.7	**50.7**	**58.8**
Candidate set size: $K = 100$											
Oracle	50.1	$758_{\pm 1.0}$	32.2	79.0	53.5	41.1	38.5	46.5	29.8	52.5	59.7
No-Reranking	47.2	$291_{\pm 0.8}$	0.00	74.1	**51.5**	39.4	34.5	43.0	28.5	49.6	56.8
No-Pruning	47.4	$1829_{\pm 3.1}$	100	74.2	50.1	39.4	35.1	43.4	28.5	50.1	**58.5**
Tuned-Fixed-k	47.7	$744_{\pm 0.5}$	29.0	74.4	50.1	**39.7**	35.8	44.1	**28.8**	**50.2**	**58.5**
Adaptive-k	**47.8**	$\mathbf{401}_{\pm 1.3}$	**6.59**	**74.8**	50.3	**39.7**	**36.5**	**44.2**	28.7	49.9	**58.5**
Ours	47.7	$597_{\pm 1.4}$	19.2	74.6	50.5	39.4	35.8	**44.2**	28.7	**50.2**	**58.5**

increases much more rapidly on GERBIL than on ZELDA. This suggests that gold entities appear at deeper positions in the GERBIL candidate rankings relatively frequently. Correspondingly, the average candidate count of the proposed method increases with K, while those of Tuned-Fixed-k and Adaptive-k remain nearly constant. Thus, the proposed method adjusts its pruning boundary as the initial candidate set exposes plausible entities at deeper ranks, whereas the baselines do not adequately respond to this change in the required candidate depth. This behavior is consistent with the analysis in Section 6.3, which shows that our method places its pruning boundary closer to the gold entity rank than the baselines. The larger candidate sets produced by our method on GERBIL therefore reflect adaptive allocation of cross-encoder computation to preserve useful candidates. At the same time, the method still prunes most of the initial candidate set and substantially reduces computation relative to No-Pruning, yielding a favorable balance between efficiency and accuracy in this more challenging setting.

C.3 Paired Bootstrap Confidence Intervals

For each pipeline, benchmark, candidate set size, and baseline discussed in Section 5 and Appendices C.1 and C.2, we estimate uncertainty in the accuracy difference using a paired bootstrap over test mentions (Koehn, 2004; Muennighoff et al., 2025). Let $Y^{\mathrm{P}}_{d,i}, Y^{\mathrm{B}}_{d,i} \in \{0, 1\}$ be the correctness indicators for mention i in dataset d. Each indicator equals 1 if the entity predicted by the corresponding method matches the gold entity and 0 otherwise. We define the paired difference:

$$D_{d,i} = Y^{\mathrm{P}}_{d,i} - Y^{\mathrm{B}}_{d,i} \in \{-1, 0, 1\}, \tag{18}$$

Table 8: E5 pipeline performance on GERBIL with the KILT KB for different initial candidate set sizes. Metric definitions and formatting conventions are the same as in Table 5.

	Acc.	Inference Time (s) ↓	Avg. Cands ↓	AIDA-B	MSNBC	DER	KORE50	N3-R128	N3-R500	OKE-2015	OKE-2016
Candidate set size: $K = 10$											
Oracle	49.3	$353_{\pm 0.7}$	3.85	75.1	50.9	43.2	42.6	40.7	27.5	52.3	62.4
No-Reranking	46.6	$293_{\pm 3.6}$	0.00	68.9	**49.5**	**42.5**	37.8	38.1	26.5	50.1	59.1
No-Pruning	47.0	$459_{\pm 3.6}$	10.0	70.8	46.1	41.4	**40.5**	39.2	26.5	**51.4**	**60.0**
Tuned-Fixed-k	47.0	$444_{\pm 1.5}$	9.00	70.8	46.2	41.4	**40.5**	39.2	26.4	**51.4**	**60.0**
Adaptive-k	47.0	$\mathbf{373}_{\pm 1.2}$	**4.52**	**70.9**	46.6	41.8	39.9	39.2	**26.7**	51.0	**60.0**
Ours	**47.2**	$379_{\pm 2.5}$	4.91	**70.9**	47.2	41.4	**40.5**	**39.4**	26.5	**51.4**	**60.0**
Candidate set size: $K = 30$											
Oracle	49.5	$463_{\pm 2.5}$	11.0	75.4	51.0	43.2	42.6	40.7	27.6	53.1	62.4
No-Reranking	46.6	$293_{\pm 3.6}$	0.00	68.9	**49.5**	**42.5**	37.8	38.1	26.5	50.1	59.1
No-Pruning	46.9	$778_{\pm 1.1}$	30.0	70.7	46.0	41.4	39.9	39.0	26.6	**51.9**	59.7
Tuned-Fixed-k	47.0	$444_{\pm 1.5}$	9.00	70.8	46.2	41.4	40.5	39.2	26.4	51.4	**60.0**
Adaptive-k	47.0	$\mathbf{373}_{\pm 3.4}$	**4.61**	71.0	46.6	41.8	39.9	39.2	**26.7**	51.0	**60.0**
Ours	**47.4**	$426_{\pm 0.8}$	8.18	**71.1**	47.7	42.1	**41.2**	**39.3**	26.4	51.6	59.7
Candidate set size: $K = 50$											
Oracle	49.5	$571_{\pm 1.5}$	18.1	75.5	51.0	43.2	42.6	40.7	27.6	53.2	62.4
No-Reranking	46.6	$293_{\pm 3.6}$	0.00	68.9	**49.5**	**42.5**	37.8	38.1	26.5	50.1	59.1
No-Pruning	46.9	$1096_{\pm 0.3}$	50.0	70.7	45.8	41.4	39.9	39.0	26.5	**52.0**	59.7
Tuned-Fixed-k	47.0	$444_{\pm 1.5}$	9.00	70.8	46.2	41.4	40.5	39.2	26.4	51.4	**60.0**
Adaptive-k	47.0	$\mathbf{373}_{\pm 1.1}$	**4.62**	71.0	46.8	41.8	39.9	39.2	**26.7**	51.0	**60.0**
Ours	**47.4**	$471_{\pm 3.3}$	10.7	**71.2**	47.7	42.1	**41.2**	**39.4**	26.4	51.7	59.7
Candidate set size: $K = 100$											
Oracle	49.6	$841_{\pm 4.0}$	35.6	75.6	51.0	43.2	42.6	40.9	27.7	53.4	62.6
No-Reranking	46.6	$293_{\pm 3.6}$	0.00	68.9	**49.5**	**42.5**	37.8	38.1	26.5	50.1	59.1
No-Pruning	46.8	$1890_{\pm 4.7}$	100	70.5	45.6	41.4	39.2	39.2	26.6	**52.2**	**60.0**
Tuned-Fixed-k	47.0	$444_{\pm 1.5}$	9.00	70.8	46.2	41.4	40.5	39.2	26.4	51.4	**60.0**
Adaptive-k	47.0	$\mathbf{373}_{\pm 0.7}$	**4.63**	71.0	46.8	41.8	39.9	39.2	**26.7**	51.0	**60.0**
Ours	**47.5**	$550_{\pm 4.5}$	15.1	**71.2**	47.7	42.1	**41.2**	**39.4**	26.5	51.9	**60.0**

where $D_{d,i} = 1$ means that the proposed method corrects a baseline error and $D_{d,i} = -1$ means that it introduces an error on a mention answered correctly by the baseline. The observed dataset-level accuracy difference is

$$\widehat{\Delta}_d = \frac{1}{N_d} \sum_{i=1}^{N_d} D_{d,i}, \tag{19}$$

where N_d is the number of test mentions in dataset d.

After the models are trained, we keep their predictions fixed and generate 10,000 bootstrap samples for every test dataset. We index these samples by $b \in \{1, \ldots, 10{,}000\}$. The b-th bootstrap sample for dataset d contains N_d mentions drawn at random with replacement from that dataset. Its size is therefore equal to the size of the original test set, although some mentions may appear more than once and others may not appear. We use the same sampled mentions for both methods, calculate their accuracies, and subtract the baseline accuracy from the accuracy of the proposed method. We denote this accuracy difference on the b-th bootstrap sample of dataset d by $\widehat{\Delta}_d^{(b)}$. This shared sample makes the comparison paired.

For each value of b, we average the differences from the nine ZELDA test datasets to obtain one bootstrap estimate of the macro-average difference:

$$\widehat{\Delta}_{\text{macro}}^{(b)} = \frac{1}{9} \sum_{d=1}^{9} \widehat{\Delta}_d^{(b)}. \tag{20}$$

Table 9: Paired bootstrap accuracy differences between the proposed method and each baseline with the dual-encoder pipeline evaluated on ZELDA for different initial candidate set sizes, in percentage points. For each baseline, the first row reports the observed accuracy difference between the proposed method and the baseline, and the second row reports its 95% confidence interval. Acc. is the macro-average accuracy. Pink shading denotes a positive accuracy difference, while **red font** denotes a confidence interval that excludes zero.

Baseline	Acc.	AIDA-B	TWEEKI	REDDIT-POSTS	REDDIT-COMM.	WNED-CWEB	WNED-WIKI	SLINKS-TAIL	SLINKS-SHADOW	SLINKS-TOP
Candidate set size: $K = 10$										
No-Reranking	**+2.4**	**+1.9**	**+2.4**	+1.0	**+3.4**	**+1.8**	**+0.9**	**+0.4**	**+5.9**	**+3.4**
	[1.9, 2.8]	**[1.1, 2.6]**	**[0.9, 4.1]**	[−0.4, 2.4]	**[1.6, 5.3]**	**[1.4, 2.3]**	**[0.4, 1.3]**	**[0.1, 0.9]**	**[3.7, 8.2]**	**[1.1, 5.8]**
No-Pruning	**+0.5**	**+1.6**	+1.2	+0.3	+0.6	**+0.6**	+0.2	+0.1	−1.2	+1.3
	[0.2, 0.8]	**[1.0, 2.2]**	[0.0, 2.3]	[−0.3, 1.0]	[0.0, 1.4]	**[0.4, 0.9]**	[0.0, 0.5]	[−0.2, 0.4]	[−2.5, 0.1]	[0.0, 2.8]
Tuned-Fixed-k	**+0.5**	**+1.6**	+1.2	+0.3	+0.6	**+0.6**	+0.2	+0.1	−1.2	+1.3
	[0.2, 0.8]	**[1.0, 2.2]**	[0.0, 2.3]	[−0.4, 1.0]	[0.0, 1.4]	**[0.4, 0.9]**	[0.0, 0.5]	[−0.2, 0.6]	[−2.5, 0.1]	[0.0, 2.8]
Adaptive-k	**+0.5**	**+1.1**	+0.3	+0.1	+0.3	**+0.6**	**+0.3**	+0.0	−0.4	**+1.9**
	[0.2, 0.7]	**[0.6, 1.6]**	[−0.7, 1.4]	[−0.6, 0.9]	[−0.3, 0.9]	**[0.4, 0.9]**	**[0.1, 0.6]**	[−0.3, 0.3]	[−1.8, 0.9]	**[0.6, 3.2]**
Candidate set size: $K = 30$										
No-Reranking	**+2.5**	**+2.0**	**+2.3**	+0.9	**+3.6**	**+2.1**	**+0.9**	**+0.4**	**+6.2**	**+4.2**
	[2.0, 3.0]	**[1.2, 2.7]**	**[0.8, 4.0]**	[−0.6, 2.3]	**[1.7, 5.5]**	**[1.6, 2.5]**	**[0.5, 1.4]**	**[0.1, 0.9]**	**[4.0, 8.5]**	**[1.9, 6.5]**
No-Pruning	**+0.7**	**+2.3**	+1.2	+0.1	+0.9	**+0.7**	**+0.3**	+0.1	−1.5	**+2.5**
	[0.4, 1.1]	**[1.6, 3.0]**	[−0.1, 2.4]	[−0.7, 1.0]	[0.0, 1.9]	**[0.4, 1.0]**	**[0.0, 0.6]**	[−0.2, 0.4]	[−3.1, 0.0]	**[0.9, 4.3]**
Tuned-Fixed-k	**+0.7**	**+1.7**	+1.0	+0.1	+0.8	**+0.9**	+0.3	+0.1	−0.9	**+2.1**
	[0.4, 1.0]	**[1.1, 2.3]**	[−0.1, 2.2]	[−0.7, 1.0]	[0.0, 1.6]	**[0.6, 1.2]**	[0.0, 0.6]	[−0.2, 0.6]	[−2.3, 0.6]	**[0.6, 3.7]**
Adaptive-k	**+0.6**	**+1.2**	+0.2	+0.0	+0.5	**+0.9**	**+0.4**	+0.0	−0.2	**+2.8**
	[0.3, 0.9]	**[0.7, 1.8]**	[−0.8, 1.3]	[−0.7, 0.7]	[−0.2, 1.3]	**[0.6, 1.1]**	**[0.1, 0.6]**	[−0.3, 0.3]	[−1.5, 1.1]	**[1.3, 4.2]**
Candidate set size: $K = 50$										
No-Reranking	**+2.5**	**+2.0**	**+2.3**	+0.9	**+3.6**	**+2.1**	**+0.9**	**+0.4**	**+6.2**	**+4.2**
	[2.0, 3.0]	**[1.3, 2.7]**	**[0.8, 4.0]**	[−0.6, 2.3]	**[1.7, 5.5]**	**[1.7, 2.6]**	**[0.5, 1.3]**	**[0.1, 0.9]**	**[3.9, 8.5]**	**[1.9, 6.5]**
No-Pruning	**+0.9**	**+2.6**	**+1.6**	+0.1	**+1.1**	**+0.8**	**+0.3**	+0.1	−1.2	**+2.5**
	[0.5, 1.2]	**[1.9, 3.3]**	**[0.2, 3.0]**	[−0.7, 1.0]	**[0.2, 2.2]**	**[0.5, 1.1]**	**[0.0, 0.7]**	[−0.2, 0.6]	[−2.8, 0.3]	**[0.8, 4.3]**
Tuned-Fixed-k	**+0.7**	**+1.7**	+1.0	+0.1	+0.8	**+0.9**	+0.3	+0.1	−0.9	**+2.1**
	[0.4, 1.0]	**[1.1, 2.4]**	[−0.1, 2.2]	[−0.7, 1.0]	[0.0, 1.6]	**[0.6, 1.2]**	[0.0, 0.6]	[−0.2, 0.6]	[−2.3, 0.6]	**[0.6, 3.7]**
Adaptive-k	**+0.6**	**+1.2**	+0.2	+0.0	+0.5	**+0.9**	**+0.3**	+0.0	−0.2	**+2.8**
	[0.3, 0.9]	**[0.7, 1.8]**	[−0.8, 1.3]	[−0.7, 0.7]	[−0.2, 1.3]	**[0.6, 1.2]**	**[0.1, 0.6]**	[−0.3, 0.3]	[−1.5, 1.1]	**[1.3, 4.2]**
Candidate set size: $K = 100$										
No-Reranking	**+2.6**	**+2.1**	**+2.3**	+1.0	**+3.6**	**+2.1**	**+0.9**	**+0.4**	**+6.2**	**+4.3**
	[2.1, 3.1]	**[1.3, 2.8]**	**[0.8, 4.0]**	[−0.4, 2.4]	**[1.7, 5.5]**	**[1.7, 2.6]**	**[0.5, 1.3]**	**[0.1, 0.9]**	**[3.9, 8.5]**	**[2.0, 6.6]**
No-Pruning	**+1.0**	**+2.9**	+1.4	+0.3	**+1.4**	**+1.1**	**+0.5**	+0.1	−1.2	**+2.5**
	[0.6, 1.4]	**[2.1, 3.6]**	[0.0, 2.8]	[−0.9, 1.3]	**[0.3, 2.5]**	**[0.7, 1.4]**	**[0.1, 0.8]**	[−0.2, 0.4]	[−2.9, 0.3]	**[0.8, 4.4]**
Tuned-Fixed-k	**+0.7**	**+1.8**	+1.0	+0.3	+0.8	**+1.0**	+0.3	+0.1	−0.9	**+2.2**
	[0.4, 1.1]	**[1.2, 2.4]**	[−0.1, 2.2]	[−0.6, 1.1]	[0.0, 1.6]	**[0.7, 1.3]**	[0.0, 0.6]	[−0.2, 0.6]	[−2.3, 0.6]	**[0.7, 3.8]**
Adaptive-k	**+0.7**	**+1.3**	+0.2	+0.1	+0.5	**+0.9**	**+0.3**	+0.0	−0.2	**+2.9**
	[0.4, 1.0]	**[0.7, 1.8]**	[−0.8, 1.3]	[−0.6, 0.9]	[−0.2, 1.3]	**[0.6, 1.2]**	**[0.1, 0.6]**	[−0.3, 0.3]	[−1.5, 1.1]	**[1.4, 4.3]**

This procedure gives every dataset equal weight irrespective of its number of mentions, matching the main evaluation measure. Repeating the procedure produces a distribution of 10,000 accuracy differences. We take its 2.5th and 97.5th percentiles as the 95% confidence interval. An interval entirely above zero supports higher accuracy for the proposed method.

Tables 9 and 10 report the paired differences for the dual-encoder and E5 pipelines on ZELDA, respectively, and Tables 11 and 12 report those for the dual-encoder and E5 pipelines on GERBIL. For every candidate set size $K \in \{10, 30, 50, 100\}$ in both pipelines on ZELDA, our method has a positive macro-average accuracy difference relative to No-Reranking, No-Pruning, Tuned-Fixed-k, and Adaptive-k, and every corresponding 95% confidence interval lies strictly above zero. On GERBIL, the same patterns are observed for most combinations of K and baseline. Taken together, these results provide evidence that the macro-average accuracy advantage of the proposed method is robust to resampling the test mentions and remains consistent across both evaluated pipeline architectures, both benchmarks, and all evaluated candidate set sizes.

C.4 Multiple Random Seeds for Model Training

To assess the sensitivity of our results to randomness in model training, we train the dual-encoder pipeline on ZELDA with four additional random seeds: 40, 41, 43, and 44. Together with the

Table 10: Paired bootstrap accuracy differences between the proposed method and each baseline with the E5 pipeline evaluated on ZELDA for different initial candidate set sizes. Metric definitions and formatting conventions are the same as in Table 9.

Baseline	Acc.	AIDA-B	TWEEKI	REDDIT-POSTS	REDDIT-COMM.	WNED-CWEB	WNED-WIKI	SLINKS-TAIL	SLINKS-SHADOW	SLINKS-TOP
Candidate set size: $K = 10$										
No-Reranking	**+1.1** **[0.7, 1.6]**	**+2.2** **[1.5, 2.9]**	+0.5 [−0.8, 1.9]	**+1.3** **[0.1, 2.6]**	+0.9 [−0.5, 2.4]	**+1.1** **[0.6, 1.5]**	+0.4 [−0.1, 0.8]	+0.0 [−0.4, 0.4]	+1.9 [−0.3, 4.0]	+1.9 [−0.2, 4.0]
No-Pruning	**+1.3** **[0.9, 1.6]**	**+2.0** **[1.4, 2.6]**	**+3.1** **[1.9, 4.5]**	**+1.4** **[0.1, 2.7]**	**+1.7** **[0.6, 3.0]**	**+1.6** **[1.3, 1.9]**	+0.0 [−0.3, 0.3]	−0.1 [−0.3, 0.0]	+0.3 [−1.2, 1.9]	+1.2 [−0.4, 2.9]
Tuned-Fixed-k	**+1.3** **[0.9, 1.6]**	**+2.0** **[1.4, 2.6]**	**+3.1** **[1.9, 4.4]**	**+1.4** **[0.1, 2.7]**	**+1.7** **[0.6, 3.0]**	**+1.6** **[1.3, 1.9]**	+0.0 [−0.3, 0.3]	−0.1 [−0.3, 0.0]	+0.3 [−1.2, 1.9]	+1.2 [−0.4, 2.9]
Adaptive-k	**+0.4** **[0.1, 0.7]**	**+0.6** **[0.2, 1.0]**	**+1.2** **[0.2, 2.1]**	+0.1 [−0.4, 0.7]	+0.3 [−0.5, 1.3]	**+0.4** **[0.2, 0.7]**	+0.1 [−0.1, 0.3]	−0.1 [−0.6, 0.2]	+0.3 [−0.9, 1.5]	+0.7 [−0.4, 1.9]
Candidate set size: $K = 30$										
No-Reranking	**+1.2** **[0.7, 1.6]**	**+2.1** **[1.4, 2.8]**	+0.6 [−0.7, 1.9]	+1.0 [−0.1, 2.1]	+1.4 [0.0, 2.8]	**+1.1** **[0.7, 1.5]**	+0.3 [−0.1, 0.8]	+0.0 [−0.4, 0.4]	+2.0 [−0.2, 4.1]	+2.1 [0.0, 4.2]
No-Pruning	**+1.7** **[1.3, 2.1]**	**+2.8** **[2.2, 3.5]**	**+3.8** **[2.4, 5.3]**	+1.3 [−0.1, 2.8]	**+2.4** **[0.9, 3.9]**	**+2.0** **[1.6, 2.4]**	+0.0 [−0.3, 0.3]	−0.1 [−0.3, 0.0]	+0.9 [−0.9, 2.5]	**+2.3** **[0.6, 4.2]**
Tuned-Fixed-k	**+1.3** **[0.9, 1.7]**	**+1.9** **[1.3, 2.5]**	**+3.3** **[1.9, 4.7]**	+1.1 [−0.1, 2.6]	**+2.2** **[0.9, 3.6]**	**+1.7** **[1.3, 2.0]**	+0.0 [−0.3, 0.2]	−0.1 [−0.3, 0.0]	+0.4 [−1.1, 2.1]	+1.4 [−0.3, 3.1]
Adaptive-k	**+0.5** **[0.2, 0.7]**	**+0.5** **[0.1, 0.9]**	**+1.3** **[0.5, 2.2]**	−0.1 [−0.9, 0.6]	+0.8 [−0.2, 1.7]	**+0.5** **[0.2, 0.7]**	+0.0 [−0.2, 0.3]	−0.1 [−0.4, 0.2]	+0.7 [−0.6, 1.9]	+0.8 [−0.4, 2.0]
Candidate set size: $K = 50$										
No-Reranking	**+1.2** **[0.7, 1.6]**	**+2.1** **[1.4, 2.8]**	+0.6 [−0.7, 1.9]	+1.0 [−0.1, 2.1]	+1.4 [0.0, 2.8]	**+1.1** **[0.7, 1.6]**	+0.3 [−0.1, 0.8]	+0.0 [−0.4, 0.4]	+2.0 [−0.2, 4.1]	+2.1 [0.0, 4.2]
No-Pruning	**+1.8** **[1.4, 2.2]**	**+3.0** **[2.3, 3.7]**	**+3.8** **[2.4, 5.3]**	+1.4 [−0.1, 3.0]	**+2.4** **[0.9, 3.8]**	**+2.3** **[1.9, 2.6]**	+0.0 [−0.3, 0.3]	−0.1 [−0.3, 0.0]	+0.7 [−1.1, 2.4]	**+2.8** **[1.0, 4.6]**
Tuned-Fixed-k	**+1.3** **[0.9, 1.7]**	**+1.9** **[1.3, 2.5]**	**+3.3** **[1.9, 4.7]**	+1.1 [−0.1, 2.6]	**+2.2** **[0.9, 3.6]**	**+1.7** **[1.3, 2.0]**	+0.0 [−0.3, 0.2]	−0.1 [−0.3, 0.0]	+0.4 [−1.1, 2.1]	+1.4 [−0.3, 3.1]
Adaptive-k	**+0.5** **[0.2, 0.7]**	**+0.5** **[0.1, 0.9]**	**+1.3** **[0.5, 2.2]**	−0.1 [−0.9, 0.6]	+0.8 [−0.2, 1.7]	**+0.5** **[0.2, 0.7]**	+0.0 [−0.2, 0.2]	−0.1 [−0.6, 0.2]	+0.7 [−0.6, 1.9]	+0.8 [−0.4, 2.0]
Candidate set size: $K = 100$										
No-Reranking	**+1.2** **[0.7, 1.6]**	**+2.1** **[1.4, 2.8]**	+0.6 [−0.7, 2.0]	+1.0 [−0.1, 2.1]	+1.4 [0.0, 2.8]	**+1.2** **[0.7, 1.6]**	+0.3 [−0.1, 0.8]	+0.0 [−0.4, 0.4]	+2.0 [−0.2, 4.1]	+2.1 [0.0, 4.2]
No-Pruning	**+1.9** **[1.5, 2.3]**	**+3.3** **[2.6, 4.1]**	**+4.0** **[2.6, 5.5]**	+1.4 [−0.1, 3.0]	**+2.7** **[1.3, 4.2]**	**+2.3** **[1.9, 2.7]**	+0.0 [−0.3, 0.3]	−0.1 [−0.3, 0.0]	+0.8 [−1.1, 2.7]	**+3.0** **[1.1, 4.9]**
Tuned-Fixed-k	**+1.3** **[1.0, 1.7]**	**+1.9** **[1.3, 2.5]**	**+3.3** **[1.9, 4.7]**	+1.1 [−0.3, 2.6]	**+2.2** **[0.9, 3.4]**	**+1.7** **[1.3, 2.0]**	+0.0 [−0.3, 0.2]	−0.1 [−0.3, 0.0]	+0.4 [−1.1, 2.1]	+1.4 [−0.3, 3.1]
Adaptive-k	**+0.5** **[0.2, 0.7]**	**+0.5** **[0.1, 0.9]**	**+1.3** **[0.5, 2.2]**	−0.1 [−0.9, 0.6]	+0.8 [−0.2, 1.7]	**+0.5** **[0.3, 0.7]**	+0.0 [−0.2, 0.3]	−0.1 [−0.6, 0.2]	+0.7 [−0.6, 1.9]	+0.8 [−0.4, 2.0]

seed-42 run used in the main experiment, these yield five independently trained pipelines. We evaluate the proposed method and the baselines with each pipeline for initial candidate set sizes $K \in \{10, 30, 50, 100\}$. All other experimental settings follow the main experiment, with pruning thresholds and baseline hyperparameters selected independently for each seed using the development set.

Figure 4 visualizes the accuracy–efficiency trade-offs for the individual training seeds. The proposed method occupies the favorable upper-left region relative to the reranking baselines in both the inference-time and candidate-count plots, combining high accuracy with low computational cost. Despite variation across training seeds, its results remain concentrated in this favorable region across the evaluated candidate set sizes. No-Reranking has a lower computational cost but also substantially lower accuracy. These results indicate that the favorable accuracy–efficiency trade-off of the proposed method persists across the evaluated training seeds.

D Additional Analysis

D.1 Distribution of Gold Entity Rank and Pruned Candidate Set Size

Section 6.3 showed that the proposed method places its pruning boundary closer to the gold entity rank than Adaptive-k. To better understand this result, we examine how the two methods distribute their candidate budgets across mentions of different difficulty. Using the dual-encoder pipeline with initial candidate set size $K = 30$, we compare the rank distribution of the gold entity in the original

Table 11: Paired bootstrap accuracy differences between the proposed method and each baseline with the dual-encoder pipeline evaluated on GERBIL with the KILT KB for different initial candidate set sizes. Metric definitions and formatting conventions are the same as in Table 9.

Baseline	Acc.	AIDA-B	MSNBC	DER	KORE50	N3-R128	N3-R500	OKE-2015	OKE-2016
Candidate set size: $K = 10$									
No-Reranking	**+1.1** **[0.5, 1.7]**	**+0.9** **[0.2, 1.8]**	−0.5 [−2.0, 0.9]	+0.3 [−1.7, 2.4]	**+3.4** **[0.7, 6.8]**	**+1.9** **[0.8, 3.1]**	+0.1 [−0.7, 0.9]	+0.6 [−0.8, 2.0]	+2.1 [0.0, 4.1]
No-Pruning	**+0.4** **[0.2, 0.7]**	**+0.4** **[0.1, 0.7]**	**+0.7** **[0.1, 1.3]**	+0.0 [0.0, 0.0]	+1.4 [0.0, 3.4]	+0.7 [0.0, 1.4]	−0.1 [−0.3, 0.0]	+0.2 [−0.3, 0.8]	+0.3 [0.0, 0.9]
Tuned-Fixed-k	**+0.3** **[0.1, 0.6]**	**+0.5** **[0.2, 0.8]**	**+0.7** **[0.1, 1.3]**	+0.0 [0.0, 0.0]	+0.7 [0.0, 2.0]	+0.6 [−0.1, 1.3]	−0.1 [−0.3, 0.0]	+0.2 [−0.3, 0.8]	+0.3 [0.0, 0.9]
Adaptive-k	+0.3 [−0.1, 0.6]	+0.3 [0.0, 0.7]	+0.1 [−0.7, 0.9]	+0.0 [0.0, 0.0]	+0.7 [−1.4, 2.7]	+0.1 [−0.6, 0.8]	+0.1 [−0.2, 0.5]	+0.5 [0.0, 1.1]	+0.3 [0.0, 0.9]
Candidate set size: $K = 30$									
No-Reranking	**+1.2** **[0.7, 1.9]**	**+1.0** **[0.2, 1.8]**	+0.0 [−1.3, 1.3]	+0.7 [−1.0, 2.7]	**+3.4** **[0.7, 6.8]**	**+1.6** **[0.5, 2.7]**	+0.2 [−0.6, 1.0]	+1.1 [−0.3, 2.6]	+2.1 [0.0, 4.1]
No-Pruning	**+0.7** **[0.4, 1.1]**	**+0.8** **[0.4, 1.2]**	**+1.5** **[0.7, 2.4]**	+0.3 [0.0, 1.0]	+2.0 [0.0, 4.7]	+0.5 [−0.3, 1.4]	−0.1 [−0.3, 0.0]	+0.6 [−0.2, 1.5]	+0.3 [0.0, 0.9]
Tuned-Fixed-k	**+0.7** **[0.4, 1.1]**	**+0.8** **[0.4, 1.2]**	**+1.5** **[0.7, 2.4]**	+0.3 [0.0, 1.0]	+2.0 [0.0, 4.7]	+0.5 [−0.3, 1.4]	−0.1 [−0.3, 0.0]	+0.5 [−0.3, 1.2]	+0.3 [0.0, 0.9]
Adaptive-k	**+0.6** **[0.2, 1.0]**	**+0.4** **[0.0, 0.7]**	**+1.2** **[0.5, 2.0]**	+0.3 [0.0, 1.0]	+1.4 [−1.4, 4.1]	+0.3 [−0.5, 1.1]	+0.0 [−0.3, 0.3]	+0.8 [0.0, 1.5]	+0.3 [0.0, 0.9]
Candidate set size: $K = 50$									
No-Reranking	**+1.2** **[0.6, 1.8]**	**+1.0** **[0.3, 1.8]**	+0.0 [−1.3, 1.3]	+0.7 [−1.0, 2.7]	+2.7 [0.0, 6.1]	**+1.6** **[0.6, 2.7]**	+0.2 [−0.6, 1.0]	+1.1 [−0.3, 2.6]	+2.1 [0.0, 4.1]
No-Pruning	**+0.8** **[0.4, 1.2]**	**+0.9** **[0.5, 1.3]**	**+1.5** **[0.7, 2.4]**	+0.3 [0.0, 1.0]	+2.0 [0.0, 4.7]	+0.6 [−0.2, 1.5]	−0.1 [−0.3, 0.0]	+0.8 [−0.2, 1.7]	+0.3 [0.0, 0.9]
Tuned-Fixed-k	**+0.6** **[0.3, 1.0]**	**+0.8** **[0.4, 1.2]**	**+1.5** **[0.7, 2.4]**	+0.3 [0.0, 1.0]	+1.4 [−1.4, 4.1]	+0.5 [−0.3, 1.3]	−0.1 [−0.3, 0.0]	+0.5 [−0.3, 1.2]	+0.3 [0.0, 0.9]
Adaptive-k	**+0.5** **[0.1, 0.9]**	**+0.4** **[0.0, 0.8]**	**+1.2** **[0.5, 2.0]**	+0.3 [0.0, 1.0]	+0.7 [−2.0, 3.4]	+0.3 [−0.5, 1.1]	+0.0 [−0.3, 0.3]	+0.8 [0.0, 1.5]	+0.3 [0.0, 0.9]
Candidate set size: $K = 100$									
No-Reranking	+0.6 [−0.1, 1.3]	+0.5 [−0.3, 1.3]	−1.1 [−2.6, 0.4]	+0.0 [−1.7, 1.7]	+1.4 [−2.0, 4.7]	+1.2 [−0.1, 2.6]	+0.2 [−0.7, 1.1]	+0.6 [−1.1, 2.3]	+1.8 [−0.3, 3.8]
No-Pruning	**+0.3** **[0.2, 0.6]**	**+0.5** **[0.2, 0.7]**	+0.4 [0.0, 0.9]	+0.0 [0.0, 0.0]	+0.7 [0.0, 2.0]	**+0.8** **[0.2, 1.4]**	+0.2 [0.0, 0.5]	+0.2 [−0.3, 0.8]	+0.0 [0.0, 0.0]
Tuned-Fixed-k	+0.0 [−0.2, 0.3]	**+0.3** **[0.0, 0.6]**	+0.4 [0.0, 0.9]	−0.3 [−1.0, 0.0]	+0.0 [−2.0, 2.0]	+0.1 [−0.3, 0.6]	−0.1 [−0.3, 0.0]	+0.0 [−0.6, 0.6]	+0.0 [0.0, 0.0]
Adaptive-k	−0.1 [−0.6, 0.4]	−0.1 [−0.4, 0.2]	+0.1 [0.0, 0.4]	−0.3 [−1.0, 0.0]	−0.7 [−4.1, 2.7]	+0.0 [−0.7, 0.6]	+0.0 [−0.3, 0.3]	+0.3 [−0.5, 1.1]	+0.0 [0.0, 0.0]

candidate set with the distributions of candidate set sizes produced by Adaptive-k and the proposed method. Table 13 reports both the number and percentage of test mentions in each range.

The candidate generator ranks the gold entity in the first position for a large portion of mentions. Nevertheless, the distribution has a meaningful tail. For many mentions, the gold entity appears below the first position, occasionally at substantially deeper ranks. A system that directly returns only the top-ranked candidate cannot select the correct entity for mentions in this tail, even when the gold entity is present in the candidate set. Therefore, a candidate reranker is necessary to compare lower-ranked candidates and recover these cases. At the same time, the strong concentration at rank 1 indicates that applying the candidate reranker to all 30 candidates for every mention is unnecessary.

The two adaptive pruning methods respond to this variation differently. Adaptive-k retains no candidate sets of size 1–3 because its development-selected buffer is $B = 3$. More importantly, its candidate sets are highly concentrated at sizes of 10 or fewer, with very few extending into the deeper ranges. Thus, although Adaptive-k selects a mention-specific cutoff, its candidate set size distribution does not capture the long tail of the gold entity rank distribution. In contrast, the proposed method produces candidate sets ranging from singletons for most mentions to substantially larger sets, including sets in the 21–30 range. The concentration at size 1 mirrors the large portion of mentions for which the gold entity is ranked first, while the spread toward larger sizes provides more candidates for ambiguous mentions. These distributions explain the gap measurements reported in Section 6.3: the proposed method achieves a smaller gap than Adaptive-k.

Table 12: Paired bootstrap accuracy differences between the proposed method and each baseline with the E5 pipeline evaluated on GERBIL with the KILT KB for different initial candidate set sizes. Metric definitions and formatting conventions are the same as in Table 9.

Baseline	Acc.	AIDA-B	MSNBC	DER	KORE50	N3-R128	N3-R500	OKE-2015	OKE-2016
Candidate set size: $K = 10$									
No-Reranking	+0.6 [0.0, 1.3]	**+2.1 [1.2, 3.0]**	**−2.4 [−3.8, −0.9]**	−1.0 [−2.4, 0.3]	+2.7 [−0.7, 6.8]	**+1.4 [0.2, 2.5]**	+0.0 [−0.8, 0.8]	**+1.4 [0.2, 2.7]**	+0.9 [−1.5, 3.2]
No-Pruning	**+0.2 [0.1, 0.3]**	**+0.2 [0.0, 0.3]**	**+1.1 [0.4, 1.9]**	+0.0 [0.0, 0.0]	+0.0 [0.0, 0.0]	+0.2 [0.0, 0.6]	+0.0 [0.0, 0.0]	+0.0 [0.0, 0.0]	+0.0 [0.0, 0.0]
Tuned-Fixed-k	**+0.2 [0.1, 0.3]**	**+0.2 [0.0, 0.3]**	**+0.9 [0.3, 1.7]**	+0.0 [0.0, 0.0]	+0.0 [0.0, 0.0]	+0.2 [0.0, 0.6]	+0.1 [0.0, 0.3]	+0.0 [0.0, 0.0]	+0.0 [0.0, 0.0]
Adaptive-k	+0.2 [−0.1, 0.4]	+0.0 [−0.2, 0.3]	+0.5 [0.0, 1.2]	−0.3 [−1.0, 0.0]	+0.7 [0.0, 2.0]	+0.2 [0.0, 0.6]	−0.2 [−0.6, 0.2]	+0.5 [0.0, 1.1]	+0.0 [−1.2, 1.2]
Candidate set size: $K = 30$									
No-Reranking	**+0.8 [0.2, 1.5]**	**+2.3 [1.4, 3.2]**	**−1.9 [−3.3, −0.5]**	−0.3 [−1.4, 0.7]	+3.4 [−0.7, 7.4]	**+1.2 [0.2, 2.4]**	−0.1 [−0.9, 0.7]	**+1.5 [0.3, 2.9]**	+0.6 [−1.8, 2.9]
No-Pruning	**+0.5 [0.2, 0.8]**	**+0.4 [0.1, 0.8]**	**+1.7 [0.8, 2.6]**	+0.7 [0.0, 1.7]	+1.4 [0.0, 3.4]	+0.3 [0.0, 0.8]	−0.2 [−0.5, 0.0]	−0.3 [−0.8, 0.0]	+0.0 [0.0, 0.0]
Tuned-Fixed-k	**+0.4 [0.1, 0.7]**	**+0.4 [0.1, 0.7]**	**+1.5 [0.7, 2.4]**	+0.7 [0.0, 1.7]	+0.7 [0.0, 2.0]	+0.1 [−0.3, 0.7]	+0.0 [0.0, 0.0]	+0.2 [−0.5, 0.8]	−0.3 [−0.9, 0.0]
Adaptive-k	**+0.4 [0.0, 0.8]**	+0.2 [−0.1, 0.5]	**+1.1 [0.3, 1.9]**	+0.3 [−0.7, 1.4]	+1.4 [0.0, 3.4]	+0.1 [−0.3, 0.6]	−0.3 [−0.7, 0.0]	+0.6 [0.0, 1.4]	−0.3 [−1.5, 0.9]
Candidate set size: $K = 50$									
No-Reranking	**+0.9 [0.2, 1.6]**	**+2.4 [1.5, 3.3]**	**−1.9 [−3.3, −0.5]**	−0.3 [−1.4, 0.7]	+3.4 [−0.7, 7.4]	**+1.4 [0.2, 2.5]**	−0.1 [−0.9, 0.7]	**+1.7 [0.3, 3.0]**	+0.6 [−1.8, 2.9]
No-Pruning	**+0.6 [0.3, 0.9]**	**+0.5 [0.2, 0.9]**	**+1.9 [0.9, 2.9]**	+0.7 [0.0, 1.7]	+1.4 [0.0, 3.4]	**+0.5 [0.1, 0.9]**	−0.1 [−0.5, 0.2]	−0.3 [−0.8, 0.0]	+0.0 [0.0, 0.0]
Tuned-Fixed-k	**+0.4 [0.2, 0.7]**	**+0.5 [0.1, 0.8]**	**+1.5 [0.7, 2.4]**	+0.7 [0.0, 1.7]	+0.7 [0.0, 2.0]	+0.2 [−0.3, 0.8]	+0.0 [0.0, 0.0]	+0.3 [−0.5, 1.1]	−0.3 [−0.9, 0.0]
Adaptive-k	**+0.4 [0.1, 0.8]**	+0.3 [−0.1, 0.6]	**+0.9 [0.1, 1.7]**	+0.3 [−0.7, 1.4]	+1.4 [0.0, 3.4]	+0.2 [−0.3, 0.8]	−0.3 [−0.7, 0.0]	+0.8 [0.0, 1.5]	−0.3 [−1.5, 0.9]
Candidate set size: $K = 100$									
No-Reranking	**+0.9 [0.3, 1.6]**	**+2.3 [1.4, 3.2]**	**−1.9 [−3.3, −0.5]**	−0.3 [−1.4, 0.7]	+3.4 [−0.7, 7.4]	**+1.4 [0.2, 2.5]**	+0.0 [−0.8, 0.8]	**+1.8 [0.5, 3.2]**	+0.9 [−1.5, 3.5]
No-Pruning	**+0.7 [0.4, 1.1]**	**+0.7 [0.3, 1.0]**	**+2.1 [1.2, 3.2]**	+0.7 [0.0, 1.7]	+2.0 [0.0, 4.7]	+0.2 [−0.3, 0.8]	−0.1 [−0.5, 0.2]	−0.3 [−0.8, 0.0]	+0.0 [0.0, 0.0]
Tuned-Fixed-k	**+0.5 [0.2, 0.8]**	**+0.4 [0.1, 0.8]**	**+1.5 [0.7, 2.4]**	+0.7 [0.0, 1.7]	+0.7 [0.0, 2.0]	+0.2 [−0.3, 0.8]	+0.1 [0.0, 0.3]	+0.5 [−0.3, 1.2]	+0.0 [−0.9, 0.9]
Adaptive-k	**+0.5 [0.1, 0.9]**	+0.2 [−0.2, 0.6]	**+0.9 [0.3, 1.7]**	+0.3 [−0.7, 1.7]	+1.4 [0.0, 3.4]	+0.2 [−0.3, 0.8]	−0.2 [−0.6, 0.2]	**+0.9 [0.2, 1.8]**	+0.0 [−1.5, 1.5]

D.2 Alternative Nonconformity Scores and Additional Baseline

Our method defines nonconformity as the raw score gap between the top-ranked entity and each candidate, as shown in Equation 9. We investigate whether alternative definitions of the nonconformity score yield better results. We consider four nonconformity score variants, i.e., **Adjacent Gap**, **Normalized Gap**, **Standardized Gap**, and **Z-Normalized Score**, on the dual-encoder–cross-encoder pipeline while keeping the other experimental settings unchanged. We also evaluate an **ARDS**-based baseline adapted from Gonzalez-Gamella et al. (2026).

Adjacent Gap is inspired by the local gap signal used by Adaptive-k and incorporates this signal into our threshold selection procedure. Like Adaptive-k, it treats a sharp drop between consecutive ranked candidates as evidence for a pruning boundary. For a gold entity at rank r, its nonconformity score is the largest adjacent difference $s_{\mathrm{de}}(m, e_{(j)}) - s_{\mathrm{de}}(m, e_{(j+1)})$ for $j < r$. At inference time, Adjacent Gap retains the ranked prefix ending immediately before the first adjacent gap that exceeds the calibrated threshold. We deliberately omit the fixed buffer of size B used by Adaptive-k to avoid introducing an additional hyperparameter and to keep the threshold selection and candidate pruning procedure otherwise unchanged from the proposed adaptive candidate pruning method. This controlled design isolates whether replacing the top-candidate gap with adjacent-score information provides a more effective nonconformity score.

Because the scale and variance of candidate generator scores can vary across mentions and candidate sets, we consider Normalized Gap and Standardized Gap, which rescale the raw score gap from the top candidate for each candidate set. Normalized Gap divides the raw score gap by the score

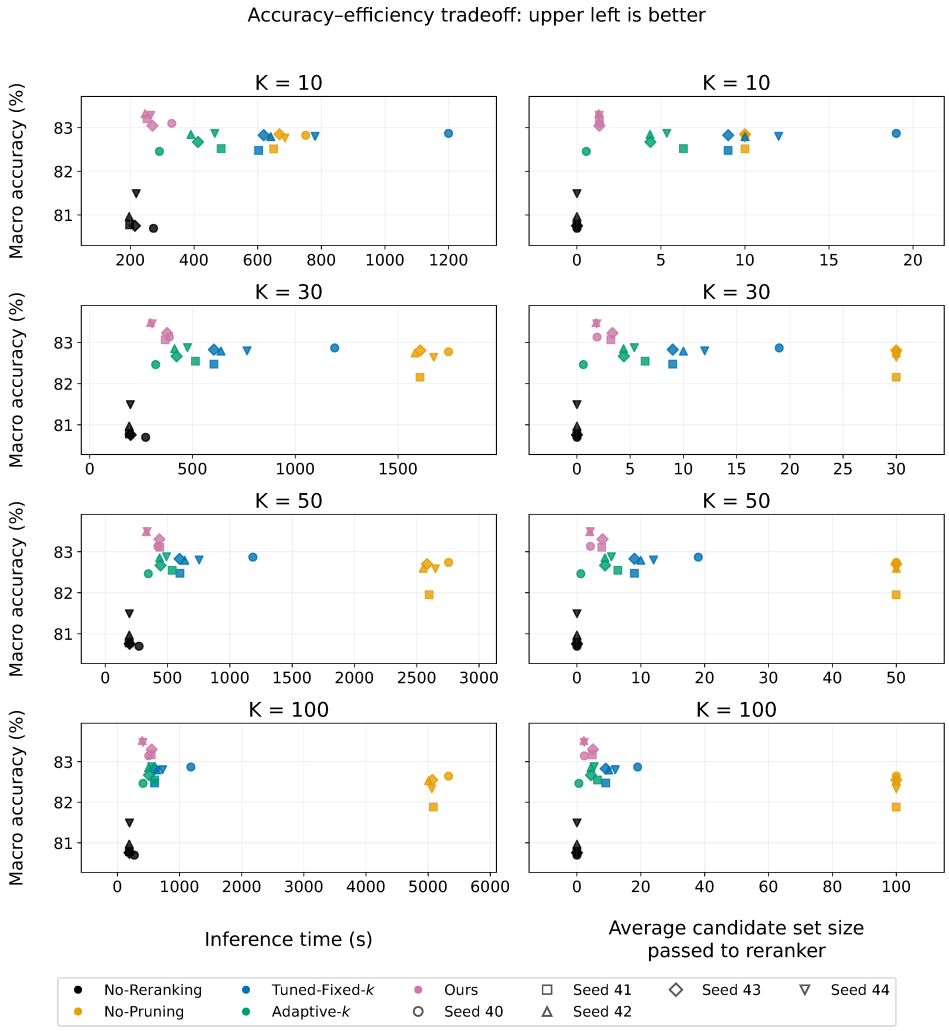

Figure 4: Accuracy–efficiency trade-offs across five training seeds for the dual-encoder pipeline on ZELDA, with $K \in \{10, 30, 50, 100\}$. Each point represents one method and training seed; colors distinguish methods and marker shapes distinguish seeds. Macro-average accuracy is plotted against (i) inference time for the complete test set, and (ii) the average number of candidates passed to the cross-encoder. Higher accuracy and lower cost correspond to the upper-left region of each panel.

range within the candidate set. Let $R(m) = s_{\text{de}}(m, e_{(1)}) - s_{\text{de}}(m, e_{(K)})$ denote this range. The normalized score is

$$A_{\text{norm}}(m, e_{(r)}) = \frac{s_{\text{de}}(m, e_{(1)}) - s_{\text{de}}(m, e_{(r)})}{\max\{R(m), \epsilon\}}, \tag{21}$$

where ϵ is a small positive constant. Let σ_m denote the standard deviation of the scores of the candidate set for mention m. Standardized Gap is then

$$A_{\text{std}}(m, e_{(r)}) = \frac{s_{\text{de}}(m, e_{(1)}) - s_{\text{de}}(m, e_{(r)})}{\sigma_m + \epsilon}. \tag{22}$$

These two variants test whether accounting for mention-specific score scale improves the comparability of nonconformity scores across examples.

Z-Normalized Score tests whether the mean and standard deviation of all candidate scores provide useful pruning information. While the gap-based variants compare each candidate with the top-ranked entity, it compares each candidate score with the mean score of the candidate set. Let $\mu_m =$

Table 13: Gold position and candidate set size distributions for the dual-encoder pipeline with initial candidate set size $K = 30$ on the ZELDA test datasets. Gold position is the rank of the gold entity in the original candidate set. For example, if the gold entity is ranked fourth for a mention, that mention is counted in column 4 for Gold Position. The 31– column means that the gold entity is not included in the original candidate set. The Adaptive-k and Ours rows instead group mentions by the number of candidates retained after pruning. For instance, if Adaptive-k retains six candidates for the same mention, it is counted in column 6–10 for the Adaptive-k row. For each method, the upper row reports counts and the lower row reports the percentage.

	1	2	3	4	5	6–10	11–20	21–30	31–
Gold Position	21608	1691	590	300	196	439	379	170	1902
	79.2%	6.2%	2.2%	1.1%	0.7%	1.6%	1.4%	0.6%	7.0%
Adaptive-k	0	0	0	21841	3266	1999	163	6	0
	0.0%	0.0%	0.0%	80.1%	12.0%	7.3%	0.6%	0.0%	0.0%
Ours	20270	2729	1051	589	389	913	622	712	0
	74.3%	10.0%	3.9%	2.2%	1.4%	3.3%	2.3%	2.6%	0.0%

$\frac{1}{K}\sum_{j=1}^{K} s_{\mathrm{de}}(m, e_{(j)})$ denote the mean candidate score for mention m. We define the Z-Normalized Score as:

$$A_{\mathrm{z}}(m, e_{(r)}) = \frac{\mu_m - s_{\mathrm{de}}(m, e_{(r)})}{\sigma_m + \epsilon}. \tag{23}$$

The candidate's nonconformity score is larger when its generator score is farther below the mean of the candidate set. Dividing by σ_m accounts for differences in score variation across candidate sets.

The additional ARDS-based baseline is motivated by the observation in Appendix D.1 that the dual-encoder ranks the gold entity first for 79.2% of ZELDA test mentions. Thus, we test whether it is sufficient to choose between sending the entire initial candidate set to the cross-encoder and using the top-ranked candidate in the initial set as the final prediction. To make this decision adaptively, we use ARDS (Gonzalez-Gamella et al., 2026), which treats the decision to invoke the reranker as a binary classification problem based on signals available after retrieval. It trains an XGBoost (Chen & Guestrin, 2016) classifier to predict the probability that the gold entity will be ranked higher when the candidate set is ordered by cross-encoder scores rather than dual-encoder scores. Following the original work, ARDS uses embedding-based features such as the similarities between a mention context and its candidate entities and the variation in pairwise distances among the candidate entity embeddings. It also divides the training examples according to whether reranking improves or worsens the gold-entity rank and fits a separate topic model to the mention contexts in each group. Similarities to the resulting topic centroids then indicate whether a mention and its candidates resemble cases in which reranking tends to help or hurt.

We split the development examples in a ratio of 8:2. The larger portion is used to train the XGBoost classifier and topic models, while the held-out portion selects a probability threshold that maximizes macro-average top-1 accuracy. At inference time, ARDS reranks all $K = 30$ candidates when the predicted probability from the XGBoost classifier reaches the selected threshold; otherwise, it outputs the top candidate from the dual-encoder as the final prediction. The decision uses neither gold labels nor cross-encoder scores. We also compare its pruning overhead with those of the other methods, measuring the time required for ARDS feature computation, additional mention-context encoding, and classifier prediction while excluding retrieval, reranking, model loading, and warmup.

Table 14 reports the results. Adjacent Gap, Normalized Gap, Standardized Gap, and Z-Normalized Score all achieve accuracy higher than or comparable to that of No-Pruning, Tuned-Fixed-k, and Adaptive-k while passing fewer candidates to the cross-encoder. This consistency indicates that the overall framework of our adaptive candidate pruning is effective across several reasonable definitions of nonconformity. However, all four variants achieve lower accuracy and retain more candidates than the proposed method. Adjacent Gap increases the average candidate set size while slightly reducing accuracy. This suggests that measuring each candidate relative to the top-ranked entity yields more reliable pruning information than measuring gaps between neighboring candidates. Normalized Gap and Standardized Gap produce candidate sets close in size to the proposed method, but degrade

Table 14: Comparison of nonconformity score functions and ARDS on the ZELDA test datasets. Hyperparameter α is selected independently for each score function using development accuracy. Acc. is the macro-average test accuracy, Avg. Cands is the micro-average number of candidates passed to the cross-encoder, and inference time covers the complete test set. Pruning time is reported as mean $\pm$ standard deviation, measured on the complete test set.

	Acc.	Inference Time (s) ↓	Pruning Time (s) ↓	Avg. Cands ↓	AIDA-B	TWEEKI	REDDIT-POSTS	REDDIT-COMM.	WNED-CWEB	WNED-WIKI	SLINKS-TAIL	SLINKS-SHADOW	SLINKS-TOP
No-Reranking	81.0	$196_{\pm 2.6}$	0	0	83.5	77.9	92.3	87.2	70.5	90.2	99.0	62.0	66.2
No-Pruning	82.7	$1732_{\pm 16.3}$	0	30.0	83.2	79.1	93.0	89.8	71.9	90.8	99.3	69.7	67.8
Tuned-Fixed-k	82.8	$747_{\pm 12.5}$	$0.2_{\pm 0.3}$	10.0	83.8	79.2	93.0	90.0	71.7	90.9	99.3	69.0	68.3
Adaptive-k	82.9	$508_{\pm 1.7}$	$0.3_{\pm 0.2}$	4.38	84.3	80.0	93.2	90.3	71.7	90.8	99.4	68.4	67.6
Ours	83.5	$380_{\pm 3.3}$	$0.3_{\pm 0.2}$	1.79	85.5	80.2	93.2	90.8	72.6	91.2	99.4	68.1	70.4
Adjacent Gap	83.4	$482_{\pm 0.5}$	$0.3_{\pm 0.2}$	3.73	85.3	80.1	93.3	90.1	72.5	91.2	99.4	68.7	69.8
Normalized Gap	83.1	$386_{\pm 1.9}$	$0.4_{\pm 0.2}$	1.91	85.1	80.2	93.0	90.3	71.9	90.8	99.4	68.1	68.6
Standardized Gap	83.1	$386_{\pm 2.4}$	$0.7_{\pm 0.2}$	1.87	85.1	80.1	93.0	90.3	71.9	91.0	99.4	68.4	69.0
Z-Normalized Score	82.9	$423_{\pm 2.9}$	$0.8_{\pm 0.2}$	2.68	84.6	79.8	93.2	90.4	71.7	90.8	99.4	68.0	68.3
ARDS	82.9	$1036_{\pm 8.3}$	$87_{\pm 0.3}$	11.9	84.1	79.7	93.2	89.7	72.1	91.0	99.3	68.1	68.6

accuracy. This indicates that normalizing the gap separately for each mention offers no additional benefit. Z-Normalized Score also achieves lower accuracy and efficiency than the proposed method. This suggests that the distance of a candidate from the top score is more informative for identifying the pruning boundary than its distance from the candidate-set mean. Because the mean reflects all the candidates from the candidate generator, including low-ranked ones, it may provide a less focused reference for distinguishing plausible candidates near the top of the ranking.

ARDS achieves accuracy comparable to Tuned-Fixed-k and Adaptive-k, but passes more candidates to the cross-encoder than either method. Relative to the proposed method, ARDS is both less accurate and substantially more expensive in terms of cross-encoder computation. These results show that a binary choice between reranking the entire candidate set and directly accepting the top-ranked candidate is insufficient, even when the gold entity is ranked first for most mentions. In particular, our method can retain only a few top-ranked candidates for such straightforward cases while moving the pruning boundary deeper when the score distribution suggests that the gold entity may appear at a lower rank. This mention-specific pruning boundary adaptation yields a more favorable accuracy–efficiency trade-off than the binary reranking decision of ARDS. Moreover, the pruning process takes substantially longer for ARDS than for the other methods. This overhead highlights the computational advantage of our simple score-threshold pruning method over introducing an additional classifier and its associated feature computation.

D.3 Alternative Candidate Reranker

Section 5 and Appendix C.1 show that our adaptive candidate pruning method improves both efficiency and effectiveness in the dual-encoder and E5 pipelines. The rescue–harm analysis in Section 6.2 shows that pruning removes incorrect candidates that the reranker would otherwise select as final predictions. This corrects more reranking errors than it introduces, leading to a net gain in accuracy. We investigate whether these benefits persist with a stronger candidate reranker.

We construct an ED pipeline that retains the BERT-based candidate generator trained in the main experiment and replaces the BERT-based cross-encoder with a cross-encoder reranker initialized from DeBERTa V3 base (184M)[3]. DeBERTa V3 has been reported to outperform encoder-based language models, including BERT, across a range of tasks (He et al., 2023), motivating its use as an alternative reranker. We train this reranker on ZELDA using candidate sets produced by the existing candidate generator trained in the main experiment. In evaluation, we measure the inference time of the pipeline using a batch size of 4 for the candidate reranker, instead of 8 in the main experiment, due to GPU memory constraints. Apart from the base model and the evaluation batch size of the reranker, we follow the same training, threshold selection, and evaluation settings as in the preceding experiments (Section 4 and Appendix C.1). Keeping the candidate generator fixed allows us to examine the effect of candidate pruning when the same retrieved candidates are scored

[3] https://huggingface.co/microsoft/deberta-v3-base

Table 15: Results of the pipeline comprising a BERT-based dual-encoder and a DeBERTa-based cross-encoder on the ZELDA test datasets for different initial candidate set sizes. Metric definitions and formatting conventions are the same as in Table 5.

	Acc.	Inference Time (s) ↓	Avg. Cands ↓	AIDA-B	TWEEKI	REDDIT-POSTS	REDDIT-COMM.	WNED-CWEB	WNED-WIKI	SLINKS-TAIL	SLINKS-SHADOW	SLINKS-TOP
Candidate set size: $K = 10$												
Oracle	87.7	$383_{\pm 3.6}$	1.29	91.1	84.9	96.2	93.4	76.0	93.2	99.6	76.7	78.2
No-Reranking	81.0	$297_{\pm 1.9}$	0.00	83.5	77.9	92.3	87.1	70.5	90.2	99.0	61.9	66.2
No-Pruning	83.7	$939_{\pm 3.2}$	10.0	87.2	**82.0**	**94.2**	**91.2**	72.4	90.2	98.9	67.1	69.7
Tuned-Fixed-k	83.7	$939_{\pm 3.2}$	10.0	87.2	**82.0**	**94.2**	**91.2**	72.4	90.2	98.9	67.1	69.7
Adaptive-k	83.6	$720_{\pm 4.7}$	6.33	87.2	**82.0**	93.9	91.1	72.5	90.2	99.0	67.1	69.6
Ours	**84.1**	$\mathbf{383}_{\pm 2.8}$	**1.30**	**87.3**	81.5	93.8	90.9	**73.1**	**91.0**	**99.4**	**68.4**	**71.7**
Candidate set size: $K = 30$												
Oracle	88.4	$515_{\pm 1.8}$	2.84	92.1	85.6	96.6	94.0	76.9	93.5	99.6	77.8	79.6
No-Reranking	81.0	$297_{\pm 1.9}$	0.00	83.5	77.9	92.3	87.1	70.5	90.2	99.0	61.9	66.2
No-Pruning	83.7	$2227_{\pm 6.9}$	30.0	87.4	81.9	**94.5**	**91.5**	72.5	90.1	98.8	66.9	69.7
Tuned-Fixed-k	83.7	$1189_{\pm 5.6}$	14.0	87.5	82.2	94.2	91.2	72.3	90.3	98.9	67.0	69.4
Adaptive-k	83.6	$751_{\pm 1.0}$	6.38	87.3	82.1	93.9	91.1	72.4	90.2	99.0	67.1	69.6
Ours	**84.5**	$\mathbf{542}_{\pm 2.1}$	**3.28**	**87.9**	**82.6**	**94.5**	**91.5**	**73.3**	**90.9**	**99.3**	**69.0**	**71.1**
Candidate set size: $K = 50$												
Oracle	88.7	$634_{\pm 3.8}$	4.15	92.6	85.9	96.6	94.2	77.2	93.6	99.6	78.0	80.3
No-Reranking	81.0	$297_{\pm 1.9}$	0.00	83.5	77.9	92.3	87.1	70.5	90.2	99.0	61.9	66.2
No-Pruning	83.7	$3543_{\pm 24.8}$	50.0	87.6	81.9	**94.5**	**91.5**	72.6	89.9	98.8	66.5	70.0
Tuned-Fixed-k	83.7	$1189_{\pm 5.6}$	14.0	87.5	82.2	94.2	91.2	72.3	90.3	98.9	67.0	69.4
Adaptive-k	83.6	$778_{\pm 4.8}$	6.38	87.3	82.1	93.9	91.1	72.4	90.2	99.0	67.1	69.6
Ours	**84.5**	$\mathbf{617}_{\pm 5.2}$	**3.98**	**87.8**	**82.6**	94.3	**91.5**	**73.4**	**90.8**	**99.3**	**69.1**	**71.1**
Candidate set size: $K = 100$												
Oracle	89.0	$904_{\pm 4.4}$	6.96	92.8	86.3	96.9	94.8	77.7	93.6	99.6	78.2	81.0
No-Reranking	81.0	$297_{\pm 1.9}$	0.00	83.5	77.9	92.3	87.1	70.5	90.2	99.0	61.9	66.2
No-Pruning	83.6	$6929_{\pm 52.5}$	100	87.4	81.9	**94.5**	**92.0**	72.6	89.7	98.6	65.8	70.4
Tuned-Fixed-k	83.7	$1189_{\pm 5.6}$	14.0	87.5	82.2	94.2	91.2	72.3	90.3	98.9	67.0	69.4
Adaptive-k	83.6	$845_{\pm 2.3}$	6.39	87.3	82.1	93.9	91.1	72.4	90.2	99.0	67.1	69.6
Ours	**84.5**	$\mathbf{767}_{\pm 1.4}$	**5.01**	**87.8**	**82.4**	**94.5**	91.8	**73.4**	**90.8**	**99.4**	**69.4**	**71.1**

Table 16: Candidate set coverage and end-to-end prediction changes relative to No-Pruning with a DeBERTa-based cross-encoder. Cov., R/H, and † are defined as in Table 2.

	Macro		Micro			
	Cov.	R/H	Cov.	R/H	Rescue	Harm
Oracle	95.1	–	93.0	–	1187	0
No-Reranking	81.0	0.68	79.2	0.64	980	1522
No-Pruning	95.1	–	93.0	–	0.00	0
Tuned-Fixed-k	94.1	$1.29^{\dagger}$	91.7	0.96	117	122
Adaptive-k	92.7	$0.87^{\dagger}$	90.2	0.97	243	251
Ours	91.3	$1.86^{\dagger}$	88.9	2.01	406	202

by a different reranker. For initial candidate set sizes $K \in \{10, 30, 50, 100\}$, we report end-to-end accuracy on each ZELDA evaluation dataset and the macro-average across datasets, together with inference time and the average number of candidates passed to the cross-encoder. We also apply the rescue–harm analysis from Section 6.2 to this pipeline to examine how replacing the reranker affects the balance between errors corrected and introduced by candidate pruning.

Table 15 reports the results with the DeBERTa-based cross-encoder. Across all evaluated initial candidate set sizes, the proposed method achieves the highest macro-average accuracy among the non-oracle systems while passing fewer candidates to the cross-encoder and requiring less inference time than No-Pruning, Tuned-Fixed-k, and Adaptive-k. As the candidate set size increases, the cost of No-Pruning grows substantially without a corresponding accuracy improvement, whereas our

method maintains higher accuracy while reranking only a small fraction of the initial candidates. These results mirror the findings with the BERT-based cross-encoder and show that the improvements in both efficiency and effectiveness persist across the two candidate rerankers.

Table 16 reports the rescue–harm analysis at the initial candidate set size $K = 30$. The proposed method again achieves the highest macro and micro R/H ratios among the evaluated pruning methods, correcting approximately twice as many predictions as it degrades. Compared with the BERT-based reranker in Table 2, its micro R/H increases from 1.73 to 2.01, while its reported macro R/H remains similar (1.93 versus 1.86). This indicates that the benefit is not specific to errors made by the BERT-based cross-encoder. Even with the stronger DeBERTa-based reranker, removing candidates that would otherwise lead to incorrect predictions remains useful, and the proposed method retains a favorable balance between correcting such errors and introducing new ones.