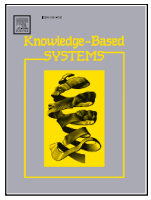
Knowledge-Based Systems 338 (2026) 115427

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys



The impact of fine-tuning on entity resolution: An experimental evaluation

Dimitrios Karapiperis, Leonidas Akritidis, Panayiotis Bozanis

International Hellenic University, School of Science and Technology, 14th km Thessaloniki - N. Moudania, 57001, Thermi, Thessaloniki, Greece

ARTICLE INFO

Keywords:
Entity resolution; deep learning; blocking; matching; embeddings; fine-tuning

ABSTRACT

Fine-tuning pre-trained language models has become the state-of-the-art approach for Entity Resolution (ER), but this has created a divide between two dominant architectures: fast-but-less-accurate bi-encoders and accurate-but-slow cross-encoders. However, a concrete gap in prior ER benchmarking remains unresolved: existing studies often evaluate architectures in isolation or on limited datasets. It remains unclear which base models and architectures are best suited for the diverse range of real-world ER datasets, each with unique characteristics and performance bottlenecks. This paper bridges this gap through an extensive empirical evaluation. We systematically compare three popular pre-trained models (MiniLM, MPNet, and BGE) across three distinct architectural paradigms: a pre-trained bi-encoder, a fine-tuned bi-encoder, and a fine-tuned cross-encoder. We tested these combinations on eight diverse real-world and semi-synthetic datasets, analyzing their performance, training costs, and final resolution times. Our results reveal a clear accuracy-vs-efficiency trade-off, identifying the fine-tuned bi-encoder as the optimal balance between performance and practical resolution speed. More importantly, we demonstrate that fine-tuning is not a universal solution. Its effectiveness is highly contingent on the dataset: it provides substantial gains on specialized domains by fixing pre-existing performance gaps but is detrimental to performance on datasets where pre-trained models are already well-aligned. These findings provide a practical guide for practitioners on selecting the optimal model and architecture based on their specific data and application requirements.

1. Introduction

Entity Resolution (ER) is a fundamental and long-standing challenge in data management, aiming to identify records across one or more datasets that refer to the same real-world entity. As a critical step in data integration and cleaning pipelines, ER enables the creation of unified, high-quality knowledge bases from disparate and often noisy data sources. The core difficulty of ER lies in accurately matching entities despite variations in their textual descriptions, such as abbreviations, misspellings, and different data formats. To address the inherent quadratic complexity of comparing all possible record pairs, the ER process is typically split into two main phases: a computationally efficient *blocking* phase to generate a small set of candidate pairs, followed by a more sophisticated and expensive *matching* phase to make the final classification on these candidates.

The advent of deep learning, and particularly the rise of large pre-trained language models like BERT [1], has marked a significant paradigm shift in the field. The state-of-the-art is no longer defined by handcrafted features but by *fine-tuning* these powerful, general-purpose models to become specialists on the ER task. This process adapts a model's vast linguistic knowledge to the specific vocabulary and semantic nuances of a given dataset, leading to substantial improvements in accuracy.

Within this fine-tuning paradigm, two dominant architectural philosophies have emerged, presenting a fundamental trade-off between computational efficiency and predictive accuracy. The first is the bi-encoder architecture, utilized in systems like BERT-ER [2], LSBlock [3] and for blocking in SC-Block [4]. This approach operates by passing each record independently through a language pre-trained model to generate a fixed-size vector embedding. Because each embedding is pre-computed in an offline step, the entire collection can be indexed using a high-speed Approximate Nearest Neighbor Search (ANNS) index. At resolution time, this allows for a highly scalable "search-then-filter" process where candidate pairs are generated in sub-linear time, making the bi-encoder the preferred architecture for large-scale blocking and efficient end-to-end pipelines.

In contrast, the cross-encoder architecture, popularized by Ditto [5] and experimentally studied in [6], prioritizes maximum accuracy. Instead of processing records independently, it concatenates the plain text of a record pair into a single input sequence for the language model. This allows the model's self-attention mechanism to perform a deep, token-level interaction, enabling it to learn nuanced, contextual relationships

E-mail addresses: dkarapiperis@ihu.edu.gr (D. Karapiperis), lakritids@ihu.gr (L. Akritidis), pbozanis@ihu.gr (P. Bozanis).

https://doi.org/10.1016/j.knosys.2026.115427
Received 2 December 2025; Received in revised form 8 January 2026; Accepted 28 January 2026
Available online 2 February 2026

(e.g., that "apple" in one record refers to the company because "iphone" is in the other). While this deep comparison typically yields superior accuracy, it comes at a prohibitively high computational cost. The cross-encoder cannot pre-compute embeddings for indexing and must perform a full forward pass of a large model for every candidate pair, resulting in a quadratic complexity that makes it infeasible for large-scale applications without a preceding blocking step.

While the recent surge of generative Large Language Models (LLMs) has transformed the data management landscape, encoder-based models remain the indispensable standard for scalable, low-latency ER pipelines. However, a concrete gap in prior benchmarking remains unresolved. Existing evidence is often derived from studies that evaluate architectures in isolation or on a narrow set of datasets , leaving it unclear how fine-tuning interacts with specific data characteristics such as schema heterogeneity and noise levels. Consequently, practitioners often lack clear guidance on which approach–a pre-trained bi-encoder, a fine-tuned bi-encoder, or a fine-tuned cross-encoder–is optimal for their specific requirements.

To bridge this gap, we conduct an extensive experimental evaluation across eight real-world and semi-synthetic datasets. We systematically compare three popular pre-trained models (MiniLM, MPNet, and BGE) across these three architectural paradigms. Going beyond the well-established trade-off between the speed of bi-encoders and the accuracy of cross-encoders, our contributions are as follows:

- Scope of Fine-Tuning: We demonstrate that fine-tuning is not a universal solution. We identify specific scenarios where it acts as a necessary precision-fixer or recall-booster, and conversely, scenarios where it is detrimental to performance on already-aligned datasets.
- Model Efficiency: We challenge the assumption that larger models are always superior by showing that smaller, fine-tuned models (e.g., MiniLM) can outperform larger baselines, offering a resource-efficient alternative.
- Practical Framework: We provide a detailed analysis of the balance between performance and computational cost, serving as a practical guide for researchers and practitioners to select the optimal ER methodology based on their specific data constraints.

The remainder of this paper is structured as follows. Section 2 reviews related work in deep learning for ER, situating our contributions within the current research landscape. Section 3 formally presents the problem definition, followed by Section 4 which details the pre-trained embedding models employed. Section 5 describes the architectural implementations and fine-tuning strategies for the bi-encoder and cross-encoder pipelines. Section 6 outlines the experimental setup, specifying the datasets, performance metrics, and implementation hyperparameters used for evaluation. Section 7 provides a comprehensive discussion and analysis of the results, examining the performance, efficiency, and qualitative behaviors of each approach. Finally, Section 8 concludes the paper by summarizing our key findings and discussing their implications for future research.

2. Related work

The application of deep learning to ER has evolved significantly, moving from sequential models to the now-dominant paradigm of fine-tuning large, pre-trained language models. Early pioneering works, such as DeepMatcher [7], established the viability of deep learning for this task by exploring a design space that included RNNs and attention mechanisms to learn representations of entity pairs. Building on this foundation, subsequent research developed more sophisticated attention-based architectures. For instance, the Multi-Context Attention model [8] proposed an integrated framework that captures self-attention within each entity, pair-attention between entities, and global-attention across the dataset to enhance matching accuracy. Other works like HierMatcher [9] introduced hierarchical matching networks to specifically address the challenges of heterogeneous and noisy data by jointly matching entities at the token, attribute, and entity levels. Concurrently, graph-based approaches have also gained traction. Systems like GNEM [10] and FlexER [11] leverage Graph Neural Networks to collectively model the relationships between multiple candidate pairs, allowing matching decisions to be influenced by the broader context of the dataset.

Large pre-trained language models, most notably BERT [1], based on the Transformer architecture [12], fundamentally altered the research landscape of the field. The state-of-the-art in ER is now largely defined by approaches that fine-tune these powerful models for the specific task of entity matching. A seminal work in this area is Ditto [5], which demonstrated that casting ER as a sequence-pair classification task and fine-tuning a pre-trained model could significantly outperform previous methods. Ditto uses a *cross-encoder* architecture, which concatenates the text of both records into a single long sequence (separated by a [SEP] token) and feeds this combined input into the language model. The model can perform deep, token-level attention between the two records. The final decision is made by a very simple linear layer on top of the model's output. Ditto further introduced key optimizations, such as injecting domain knowledge through special markers and using data augmentation to generate difficult training examples, thereby forcing the model to learn a more robust decision boundary.

Similarly, the experimental work of Brunner and Stockinger [6] established the dominance of the new paradigm. By systematically comparing transformer architectures to older deep learning methods, they proved that using fine-tuned cross-encoders for sequence-pair classification yields superior results.

The significant computational expense of these deep models prompted further research into improving their efficiency. For example, BERT-ER [2] introduced a siamese network structure that independently encodes entity records, enabling the use of a dedicated blocking module to drastically reduce the number of pairs that require expensive, deep interaction. More recently, the principle of fine-tuning has been successfully applied not only to the matching phase but also to the blocking phase of the ER pipeline. For instance, the authors in [4] present a method that fine-tunes a pre-trained bi-encoder model using supervised contrastive learning to create a highly effective embedding space specifically for the purpose of candidate generation, demonstrating the versatility of the fine-tuning approach across the entire ER workflow.

A significant body of research in ER has been dedicated to improving the efficiency of the pipeline through sophisticated blocking techniques. The advent of pre-trained language models has also heavily influenced modern blocking research. For instance, AutoBlock [13] introduced a hands-off framework that uses similarity-preserving representation learning and nearest neighbor search to generate candidate pairs without requiring extensive feature engineering. Similarly, LSBlock [3] proposed a hybrid system that combines lexical similarity with semantic similarity from dense embeddings to improve recall while maintaining high precision. The dense embeddings are generated using the bi-encoder architecture.

The authors in [14] provide a comprehensive design space exploration of how deep learning can be applied to blocking, analyzing various architectures from simple autoencoders to more complex transformer-based models. A prominent example of applying fine-tuning to blocking is found in BLINK [15], which uses a fine-tuned bi-encoder to create dense embeddings for entity linking. This bi-encoder is used for fast and scalable candidate retrieval. Another notable work, EmbER [16], leverages task-specific embeddings learned via transformer-based techniques to automate keyless joins, which implicitly involves a high-quality blocking or candidate generation step.

Moving beyond supervised techniques, Sudowoodo [17] proposes a multi-purpose data integration framework based on contrastive self-supervised learning. It aims to learn powerful, similarity-aware data representations from unlabeled data, which can then be applied to a wide variety of tasks including entity matching, error correction, and semantic type detection, thereby reducing the dependency on large, labeled gold standards.

The challenge of handling large datasets with limited computational budgets has also led to the development of progressive and active ER methodologies. To reduce labeling costs, research efforts like [18–21] proposed active learning frameworks that iteratively select the most informative examples for human annotation, optimizing the trade-off between labeling effort and model accuracy. The work on pBlocking [22] introduces a dynamic feedback loop, in which the blocking stage is iteratively adjusted using early feedback from the matcher to find the best balance between speed and accuracy. Broadening this concept, the framework for progressive entity matching [23] provides a comprehensive design space, organizing the task into four distinct steps: filtering, weighting, scheduling, and matching, which allows for a more structured approach to pay-as-you-go entity resolution.

More recently, the focus has shifted to adapting these efficiency principles to modern architectures. For instance, [24] combines active learning with in-context learning, demonstrating how Large Language Models (LLMs) can effectively tackle cross-domain ER tasks with minimal supervision. Similarly, AvengER [25] explores the potential of generative LLMs by combining fine-tuning with ensemble methods for prompt-based resolution, specifically targeting the *select* prompting strategy to mitigate position bias.

Finally, the experimental works by Zeakis et al. [26] and Karapiperis et al. [27] provide a thorough experimental analysis comparing various pre-trained embeddings, using the bi-encoder architecture, offering valuable insights into which models are most effective for both blocking and matching tasks.

3. Problem definition

Let $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ be two sets of entity records, where each record consists of a sequence of tokens. The goal of Entity Resolution is to identify the set of matching pairs $M = \{(a_i, b_j) \in A \times B\}$ such that a_i and b_j refer to the same real-world entity.

This is typically formulated as a binary classification problem where a parameterized function $f(a, b) \to [0, 1]$ predicts the probability that two records match. To clarify the structural differences examined in this study, we formally define the scoring function f for the two competing architectures:

- Bi-Encoder Formulation: The bi-encoder decouples the encoding of the two records. It maps each record independently to a dense vector space $\mathbb{R}^d$ using an encoder function ϕ (e.g., an embedding model). The similarity is then computed using a function g (a neural network classifier) over the separate embeddings: $f(a, b) = g(\phi(a), \phi(b))$. Critically, the encoding $\phi(a)$ is conditional only on the tokens of a, independent of b.
- Cross-Encoder Formulation: The cross-encoder processes the pair jointly. It employs an encoder function ψ that accepts the concatenated sequence of both records. The similarity is derived from the contextualized representation of the entire pair: $f(a, b) = g(\psi(a \oplus b))$. Here, the encoding is conditional on the interaction between tokens of a and tokens of b simultaneously.

4. Embedding models

The central goal of this work is to investigate the impact of task-specific fine-tuning on pre-trained language models for the ER matching task. To provide a comprehensive analysis, we selected three representative transformer-based models that span the spectrum of efficiency, historical performance, and the current state-of-the-art: MiniLM [28], MPNet [29], and BGE [30]. This selection allows for a nuanced evaluation of how fine-tuning affects models with different architectural features and baseline capabilities.

4.0.0.1. MiniLM (`all-MiniLM-L6-v2`). This model was chosen to represent a class of highly efficient, compact language models designed for scenarios where computational resources are constrained or high throughput is a priority. MiniLM is a product of knowledge distillation, a process where a smaller "student" model is trained to mimic the behavior of a larger "teacher" model. Specifically, it utilizes deep self-attention distillation to transfer knowledge from the final layers of a larger transformer. As a result, MiniLM is a lightweight 6-layer Transformer with only 22.7 million parameters, which maps text to a 384-dimensional vector space. Its small size and high speed–approximately five times faster than larger models like MPNet–make it an ideal baseline for efficient semantic encoding. Its inclusion allows us to investigate the performance ceiling achievable through fine-tuning a model that is explicitly built for speed.

4.0.0.2. MPNet (`all-mpnet-base-v2`). This model was selected to represent a previous generation's state-of-the-art in sentence embedding quality. MPNet introduced a novel pre-training method that unifies masked language modeling, as used in BERT, and permuted language modeling, as used in XLNet [31]. By combining these approaches, MPNet addresses the limitations of both: it leverages the dependency among predicted tokens, while also incorporating auxiliary position information to see the full sentence context, reducing the position discrepancy issue found in permuted language modeling. MPNet, which maps sentences to a 768-dimensional vector space, demonstrated superior performance on semantic textual similarity benchmarks compared to its predecessors like BERT [1] and RoBERTa [32]. It serves as our high-performance baseline, enabling an analysis of fine-tuning on a more powerful, though more computationally expensive, base model.

4.0.0.3. BGE (`bge-base-en-v1.5`). This model was chosen to represent the current state-of-the-art in general-purpose text embeddings, consistently ranking at the top of comprehensive benchmarks like the Massive Text Embedding Benchmark (MTEB). The BGE (BAAI General Embedding) model is built on a BERT-style dual-encoder but incorporates more advanced fine-tuning techniques that have proven highly effective for retrieval tasks. A key innovation is its use of contrastive training with hard negatives, a process that sharpens the model's ability to discern subtle differences by training it to distinguish a correct document from other semantically similar but incorrect documents. Furthermore, BGE employs instruction-based tuning, allowing it to adjust its embedding strategy for queries versus documents by prepending a specific instruction to the input text. The `v1.5` release also specifically addresses the issue of similarity score distribution, providing better-calibrated outputs. Including BGE provides a contemporary, high-performance benchmark against which the impact of fine-tuning on the other models can be measured, offering a more forward-looking perspective on optimal ER strategies.

5. Encoding architectures

This section details the two fundamental architectural paradigms governing deep learning-based ER. We first define the **bi-encoder**, designed for scalable retrieval via independent processing, and subsequently describe the **cross-encoder**, which leverages joint attention for high-precision matching at the cost of increased computational complexity.

5.1. The bi-encoder architecture

In this paradigm, a sentence transformer model is used to encode each record from datasets A and B into a fixed-size, high-dimensional embedding, independently of any other record. This independence is the primary advantage of this approach because it favors scalability; because each record has its own pre-computed embedding, the computationally expensive task of comparing all possible pairs can be replaced by an efficient, sub-linear similarity search using an ANNS index. This architecture is well-established in the domain, serving as the foundational

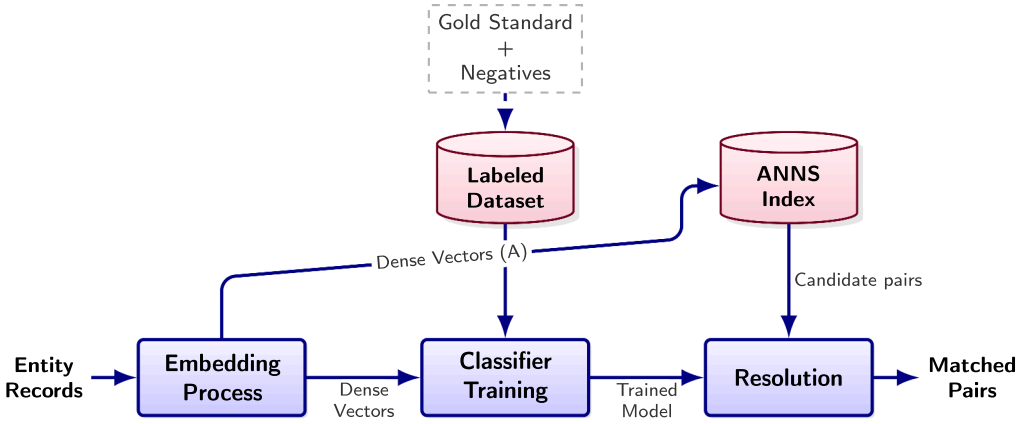


Fig. 1. The pre-trained bi-encoder pipeline.

mechanism for efficient ER systems such as BERT-ER [2], BLINK [15], SC-Block [4], Autoblock [13], Sudowoodo [17], and LSBlock [3].

Our experimental pipeline consists of several sequential stages, but it is important to note that the bi-encoder pipeline can be used in two modes: one leveraging the general knowledge of pre-trained models, and another using fine-tuned, specialized models:

1. **Embedding Process**: All entity records from both datasets are first converted into dense vector embeddings using an off-the-shelf pre-trained language model.
2. **Model Fine-Tuning**: To create a specialized model, the initial embeddings and the dataset's gold standard are used to fine-tune the pre-trained model, typically with a contrastive loss objective like triplet loss.
3. **Re-embedding Process**: After fine-tuning, this new specialized model is used to re-embed all records, resulting in a set of high-quality, domain-specific embeddings for the fine-tuned approach.
4. **Classifier Training**: for both the pre-trained and fine-tuned approaches, the respective embeddings are used as input features to train a downstream neural network classifier using labeled data.
5. **Resolution**: In the final resolution step, an ANNS index is built using the embeddings from dataset A. The embeddings from dataset B are then used as queries to retrieve candidate pairs, which are subsequently passed to the trained classifier for the final match decision.

Steps 2, which is further detailed in Section 5.1.1, and 3 are executed only during the fine-tuned approach. The pre-trained bi-encoder is presented in Fig. 1.

5.1.1. Model fine-tuning

To adapt the general-purpose, pre-trained embedding models to the specific semantic domains of each dataset, we employed a fine-tuning process based on a *triplet loss* objective function using the gold standard. This approach is designed to restructure the embedding space such that the distances between matching entity pairs (positives) are minimized while the distances between non-matching pairs (negatives) are maximized. The training examples for this process are generated as *triplets*, each consisting of an *anchor* record from dataset A, a *positive* match from B, and a *negative* non-match, which is also selected from B. This structure trains the model to pull the anchor and positive closer together while pushing the negative instance away.

A crucial component of our methodology was a sophisticated negative mining strategy. We identified *hard negatives* by building an ANNS index on the pre-trained embeddings to retrieve the top-k nearest non-matching neighbors for each anchor; these represent the most confusing examples for the model. These were complemented by *easy negatives*, obtained via random sampling, to ensure the model could still distinguish broadly dissimilar entities.

Specifically, in our setup, we generated a total of 10 training triplets for each positive pair. These were comprised of $k = 5$ hard negatives, retrieved via the ANNS index to target the most confusing non-matches, and 5 easy negatives, sampled randomly to ensure robust generalization. This strict 1:1 ratio (50% hard negatives) was determined through sensitivity analysis by varying the proportion of hard negatives from 0% (purely random sampling) to 100% (purely hard mining) on the Abt-Buy dataset, which is representative of trends observed across our benchmarks. As illustrated in Fig. 3, we observed a distinct performance trade-off: models trained exclusively on easy negatives failed to learn fine-grained decision boundaries, resulting in low precision, while those trained solely on hard negatives became overly sensitive to minor variations, degrading recall. The 1:1 ratio consistently yielded the highest F1-score, effectively balancing the discrimination of subtle differences with the generalization needed for broad non-matches.

Consequently, the total volume of training data for each dataset corresponds to ten times the size of the positive training set (calculated as 60% of the total matches reported in Table 1), ensuring that the model is exposed to a diverse and statistically significant number of difficult counter-examples during optimization. For example, on the Abt-Buy dataset, which contains 1097 matching pairs, the 60% training split yields approximately 658 positive pairs. By generating 10 negatives for each, the model is fine-tuned on a total of 6580 triplets, ensuring robust exposure to both hard and easy non-matches.

We trained each model for 2 epochs, employing the held-out validation set (20%) for periodic performance monitoring via a triplet evaluator. The resulting fine-tuned model, specialized for the specific dataset, was then saved for use in the downstream resolution task. The fine-tuned bi-encoder architecture is illustrated in Fig. 2.

5.1.2. Hybrid neural network classifier

To effectively leverage the different signals available for matching, we designed a hybrid, multi-branch neural network binary classifier using a binary cross-entropy loss function. This architecture allows for the specialized processing of different feature types before they are integrated for a final decision. The neural network consists of three parallel input branches, a feature fusion layer, and a classification head.

1. **Input Streams:** The model is designed to process three distinct types of input vectors for each candidate pair:
   - **Semantic Embeddings:** The primary input is the vector representation of the entity pair derived from the bi-encoder language model. Let $u, v \in \mathbb{R}^d$ denote the dense embedding vectors for the corresponding records from A and B, respectively. These vectors capture the deep, contextual semantic relationship between the two entities.
   - **Embedding Interactions:** A secondary input vector is explicitly engineered to capture the geometric relationship between the

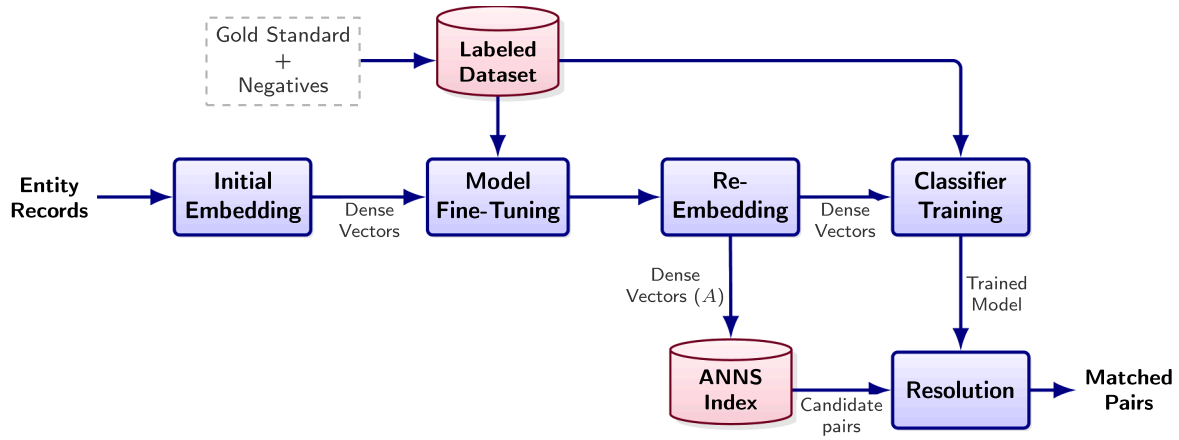


Fig. 2. The fine-tuned bi-encoder pipeline.

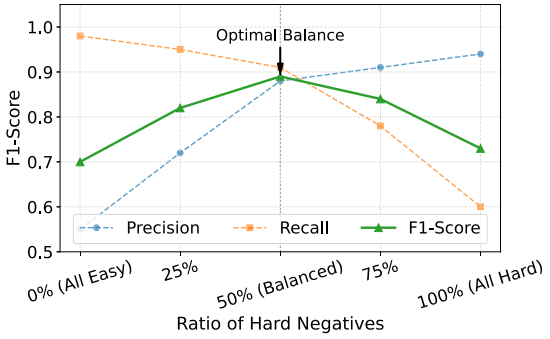


Fig. 3. Sensitivity analysis of the negative sampling strategy using the Abt-Buy dataset. The F1-score peaks at a 1:1 ratio (50% hard negatives), effectively balancing the trade-off between precision and recall.

two embeddings. We compute the element-wise absolute difference $|u - v|$. This provides the network with a direct, learnable signal of vector-space distance, effectively highlighting specific dimensions where the entities differ.

- **Engineered Features:** The third input is a low-dimensional vector f_{eng} of handcrafted features. We specifically compute two similarity metrics for the discriminative attributes (e.g., product titles, author names) of each pair: (A) **Bigram Jaccard Similarity:** We calculate the Jaccard index over the sets of character bigrams extracted from each string. This measures the overlap of shared 2-character sequences, providing robustness against minor token variations. (B) **Bigram Cosine Similarity:** We compute the cosine similarity between the vectorized representations of these character bigrams. This captures the morphological proximity of the entities, effectively handling typos or slight misspellings (e.g., "iPhone" vs. "iPhon"). This stream provides the model with valuable, explicit textual signals that complement the latent semantic understanding derived from the language model embeddings.

2. **Processing Branches:** Each input stream is fed into a dedicated processing branch composed of a sequence of fully connected (Dense) layers with Rectified Linear Unit (ReLU) activation functions.
   - The Semantic Embedding and Interaction branches are designed to process the high-dimensional inputs. Each consists of a dense layer reducing the dimensionality to 256, followed by a dropout layer with a rate of 0.2 for regularization, and a final dense layer projecting to a 128-dimensional space.
   - The Engineered Features branch is shallower, reflecting its lower input dimensionality. It consists of a dense layer projecting to 64 dimensions, followed by a second dense layer projecting to a 32-dimensional space.
3. **Feature Fusion and Classification:** The processed feature vectors from the three branches are concatenated into a single, unified representation $[u; v; |u - v|; f_{\text{eng}}]$. This fused vector is then passed through a final classification head, which comprises a dense layer of 128 neurons (ReLU activation), a dropout layer (rate of 0.2), and a final output neuron with a sigmoid activation function. The output is a scalar probability score between 0 and 1, indicating the likelihood of a match.

A fixed probability threshold of 0.5 is often suboptimal for imbalanced classification tasks. Therefore, the optimal decision threshold is empirically determined on the validation set. We plot the precision-recall curve and select the threshold that maximizes the F1-score. This optimized threshold is both used to classify the predictions on the held-out test set for the final performance evaluation as well as to classify the formulated pairs during the resolution task.

5.2. The cross-encoder architecture

The cross-encoder architecture represents a higher-accuracy, though more computationally intensive, approach for ER. This architecture follows the methodology established by the state-of-the-art Ditto system [5]. Unlike the bi-encoder which processes records independently, the cross-encoder is designed to perform a deep, contextual comparison between a pair of records. The plain text from both records is concatenated into a single sequence, separated by a special [SEP] token, and fed into a pre-trained language model. This allows the model's self-attention mechanism to directly compare and relate the specific tokens from both records, capturing nuanced semantic relationships that are often missed by a simple vector comparison.

1. **Embedding & Blocking Process**: Our experimental pipeline for the cross-encoder begins with the embedding of records of A and B using a *baseline pre-trained model*. The embeddings of A are then indexed using an ANNS structure to facilitate efficient blocking.
2. **Model Fine-Tuning**: The core of this pipeline is the fine-tuning of the pre-trained model as a binary classifier. We construct the input sequence: S = [CLS]RecordA[SEP]RecordB[SEP]. The model is trained on labeled pairs (matches and hard negatives) using a binary cross-entropy loss function. The objective is to optimize the weights such that the embedding of the special [CLS] token captures the semantic similarity of the pair.

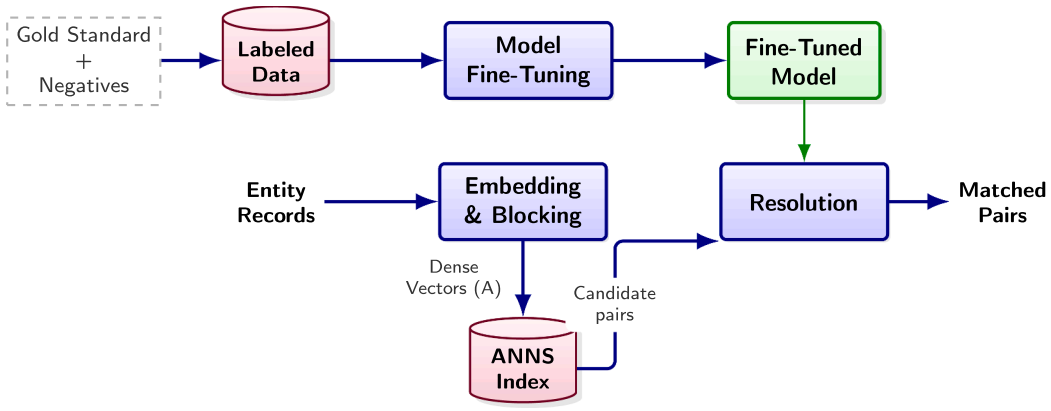


Fig. 4. The fine-tuned cross-encoder pipeline.

3. **Resolution**: In the final resolution phase, the fine-tuned cross-encoder operates strictly as a re-ranker on the candidates generated in Step 1. For each candidate pair, the model processes the joint sequence S. The input to the final classification head is the contextualized embedding of the [CLS] token. This single vector, which aggregates the deep token-level interactions learned by the network, is fed into a linear layer to output a final match probability. This probability is compared against an optimal threshold to make the match/no-match decision.

Fig. 4 illustrates the workflow of the cross-encoder architecture.

5.2.1. Model fine-tuning

The cross-encoder architecture was fine-tuned to serve as a high-accuracy matcher. This architecture processes a concatenated pair of records, allowing the self-attention mechanism of the underlying language model to perform a deep, token-level comparison between the two inputs. The fine-tuning process begins by constructing a comprehensive set of labeled training pairs from the gold standard. For each true positive pair, a corresponding set of negative pairs is generated using a hybrid mining strategy. *Hard negatives* are sourced by building an ANNS index on pre-computed bi-encoder embeddings and retrieving the nearest non-matching neighbors, which represent the most semantically confusing examples for the model. These are supplemented with *easy negatives* that are randomly sampled from the dataset to ensure robust generalization.

Similar to the bi-encoder approach, we maintained a 1:1 ratio of hard to easy negatives to ensure a fair comparison. Specifically, for each positive pair, we generated 10 negative pairs, comprising 5 hard negatives retrieved via ANNS and 5 randomly sampled easy negatives. This consistent sampling strategy ensures that both architectures are optimized against an identical set of difficult counter-examples.

The model is fine-tuned exclusively on the training set for 2 epochs. Upon completion, the model's performance on the validation set is used to determine the optimal classification threshold.

6. Experimental setup

Drawing upon the architectural paradigms reviewed in Section 2, this section details the datasets, the performance metrics, and the implementation details and hyperparameters that were used throughout the evaluation process.

6.1. Datasets

To provide a comprehensive and robust evaluation, we designed a series of experiments across eight diverse datasets, encompassing real-world benchmarks and large-scale semi-synthetic challenges. Our analysis covers a wide spectrum of ER tasks, including product matching (Abt-Buy,[1] Amazon-Walmart,[2] and Amazon-Google1), bibliographic record linkage (ACM-DBLP1 and Scholar-DBLP1), specific domain matching (IMDB-DBPEDIA[3] and Voters[4]), and large-scale scenarios (DBLP[5]). DBLP and Voters are semi-synthetic created by perturbing the records of a real dataset to generate its counterpart. The DBLP dataset simulates a high-volume, real-world bibliographic matching scenario. Its primary challenge is the sheer scale of the data, which serves as a stress test for the computational efficiency and memory footprint of any resolution method. However, we acknowledge that the perturbation strategy used for these semi-synthetic datasets may create "regular" noise patterns that are easier for models to learn compared to organic errors, potentially affecting the impact of hard negative mining in these specific cases.

Table 1
Number of entities and matches of each paired dataset.

Dataset	$\|A\|$	$\|B\|$	#Matches
Abt-Buy	1093	1082	1097
Amazon-Walmart	22 075	2555	1154
Amazon-Google	3227	1364	1300
IMDB-DBPEDIA	27 616	23 183	22 863
Scholar-DBLP	64 264	2617	5346
ACM-DBLP	2617	2295	2224
DBLP	1M	1M	500K
Voters	26 317	3000	3000

The selected datasets, which are quoted in Table 1, are all well-established benchmarks for ER in the literature [3,5–7,13,17,23,26,33–39].

6.2. Data serialization

Given the heterogeneity of schemas across datasets (e.g., product specifications vs. bibliographic citations), a consistent input format is crucial. We utilized a schema-agnostic serialization template to ensure fair comparison across diverse domains. For each record, all attribute values are concatenated into a single string, separated by a special [SEP] token (or spaces where appropriate for the specific tokenizer). This approach allows the model to treat the record as a continuous semantic

[1] https://old.dbs.uni-leipzig.de/research/projects/object_matching/benchmark_datasets_for_entity_resolution.
[2] https://hpi.de/naumann/projects/repeatability/datasets/amazon-walmart-dataset.html.
[3] https://zenodo.org/record/8433873/files/data_ea.tar.gz.
[4] https://www.ncsbe.gov/results-data/voter-registration-data.
[5] https://dblp.org/xml.

sequence, while excluding attribute names prevents the model from overfitting to schema-specific headers that may vary between datasets.

6.3. Performance metrics

Performance is measured using a holistic set of metrics covering both effectiveness (F1-Score, Precision, Recall, and Blocking Recall) and computational cost (fine-tuning time, classifier training time, embedding time, and resolution time). Recall quantifies the proportion of true matches found by the final classifier; Precision measures the proportion of retrieved pairs that are matches; and the F1-Score combines both into a single, balanced metric. Additionally, we report **Blocking Recall (Recall@k)**, which measures the proportion of true matches successfully retrieved in the top-k candidate set generated by the ANNS index. This metric serves as a crucial upper bound, verifying that the downstream matching models are not artificially bottlenecked by candidates missed during the initial blocking phase.

To ensure a rigorous and statistically valid comparison, we implemented a unified evaluation protocol across all investigated architectures. For each dataset, the labeled record pairs were partitioned into three disjoint subsets using stratified sampling: a training set (60%), used for all parameter optimization steps (including embedding fine-tuning and classifier training); a validation set (20%), used for hyperparameter tuning and classification threshold selection; and a test set (20%), strictly held out for the final performance reporting. Crucially, the exact same test set was utilized to evaluate every model. This ensures that all reported metrics reflect a direct, head-to-head comparison on identical, unseen data, guaranteeing that performance differences are attributable solely to the architecture rather than data variance.

To validate that the reported performance improvements were not due to random chance, we performed statistical significance testing. Since our experiments involve comparing the performance of two classifiers (e.g., pre-trained vs. fine-tuned) on the same test set, we utilized McNemar's Test, which is the standard non-parametric test for paired nominal data. We constructed a 2×2 contingency table for each comparison to track the number of samples misclassified by one model but correctly classified by the other. Differences in performance are reported as significant only if the calculated p-value is less than 0.05.

6.4. Implementation details and hyperparameters

We utilized a unified hyperparameter configuration for the blocking and retrieval stages. We employed the FAISS library [40] to construct a Hierarchical Navigable Small World (HNSW) index [41] for the embedding sets. We utilized the default graph construction parameters, setting the maximum number of bi-directional links per node to $M = 32$ and the size of the dynamic candidate list during index construction to $ef_{\text{construction}} = 40$. For the candidate generation (blocking) phase, we retrieved the top-k (where $k = 5$) nearest neighbors for each query record. This strict cutoff was chosen to evaluate the models' ability to place true matches in the immediate vicinity of the query, thereby testing the quality of the embedding space without relying on excessively large candidate sets. To ensure high retrieval accuracy, we performed searches using an expanded candidate list size of $ef_{\text{search}} = 64$ and utilized inner product similarity on normalized vectors (equivalent to cosine similarity). This configuration was kept constant across all eight datasets to prevent dataset-specific overfitting.

Our experimental environment was a system configured with 55 GB of RAM and an NVIDIA GPU (23 GB VRAM). The Sentence-Transformers [6] and Keras[7] Python libraries were used to perform GPU-accelerated fine-tuning and classifier training tasks, respectively.

[6] https://pypi.org/project/sentence-transformers/.
[7] https://keras.io/.

Table 2
Blocking recall (Recall@k in %) using pre-trained bi-encoders (MiniLM, MPNet, BGE).

Dataset	MiniLM	MPNet	BGE
Abt-Buy	85.4	92.7	92.8
Amazon-Walmart	90.1	93.4	89.5
Amazon-Google	92.7	94.5	93.2
IMDB-DBPEDIA	60.5	61.4	69.6
Scholar-DBLP	86.9	85.6	87.7
ACM-DBLP	72.2	60.5	78.6
DBLP	98.5	99.1	99.3
Voters	93.8	93.5	90.5

7. Experimental results

In this section, we present a detailed analysis of the performance and efficiency of the evaluated architectures across the eight benchmark datasets. We structure our findings to highlight key patterns in effectiveness (F1-score), the specific dynamics of precision and recall improvement, and the trade-offs between computational cost and resolution speed.

7.1. Impact on effectiveness (F1-Score)

Our evaluation across eight diverse datasets reveals that no single architecture is universally superior; rather, the optimal choice depends heavily on the domain characteristics and the pre-existing alignment of the base model.

The Dominance of Fine-Tuned Bi-Encoders on Product and Cross-Domain Data. On e-commerce and specific cross-domain datasets, the fine-tuned bi-encoder consistently delivered the best balance of performance, often outperforming the computationally heavier cross-encoder. On the **Abt-Buy** dataset (Fig. 5), fine-tuning the MPNet bi-encoder surged the F1-score by 88.2% (from 0.34 to 0.64), whereas the cross-encoder failed to improve performance, yielding a significantly lower score of 0.38 for BGE. Similarly, on **Amazon-Google** (Fig. 6), fine-tuning led to significant gains across all models, with MPNet improving from 0.43 to 0.65, while the cross-encoder architecture was largely ineffective. On **Amazon-Walmart** (Fig. 7), the fine-tuned MiniLM bi-encoder achieved the highest F1-score of 0.76, and on the cross-domain **IMDB-DBPEDIA** dataset (Fig. 8), the same model saw a 35% improvement to reach a top score of 0.73.

Since the cross-encoder re-ranks candidates produced by the pre-trained bi-encoder, its end-to-end performance is strictly upper-bounded by the blocking stage's Recall@k. The initial retrieval step effectively establishes a hard ceiling on the maximum possible effectiveness of the cross-encoder pipeline. To assess this limitation, we calculated the Recall@k of the blocking phase using all three pre-trained bi-encoders. As shown in Table 2, the blocking performance varies significantly across benchmarks and architectures. For well-aligned or semi-synthetic datasets like DBLP and Voters, the blocking recall is consistently high across all models (e.g., $> 98\%$ on DBLP), ensuring the cross-encoder processes nearly all relevant pairs. However, for structurally heterogeneous datasets like IMDB-DBPEDIA, the retrieval bottleneck persists regardless of model capacity. While the larger BGE model improves recall to 69.6% (compared to MiniLM's 60.5%), approximately 30% of true matches remain irretrievable by the pre-trained embeddings. We also observe that model size does not guarantee performance stability; for instance, while MPNet and BGE significantly boost recall on Abt-Buy (92.7% and 92.8% vs. 85.4% for MiniLM), MPNet struggles on ACM-DBLP, dropping to 60.5%. This highlights that while a superior base model generally provides a more robust foundation–critical for effective hard negative mining–the retrieval bottleneck on heterogeneous data remains a structural challenge for pre-trained bi-encoders.

In contrast, the fine-tuned bi-encoder overcomes this retrieval limitation by restructuring the vector space. By optimizing the embeddings via

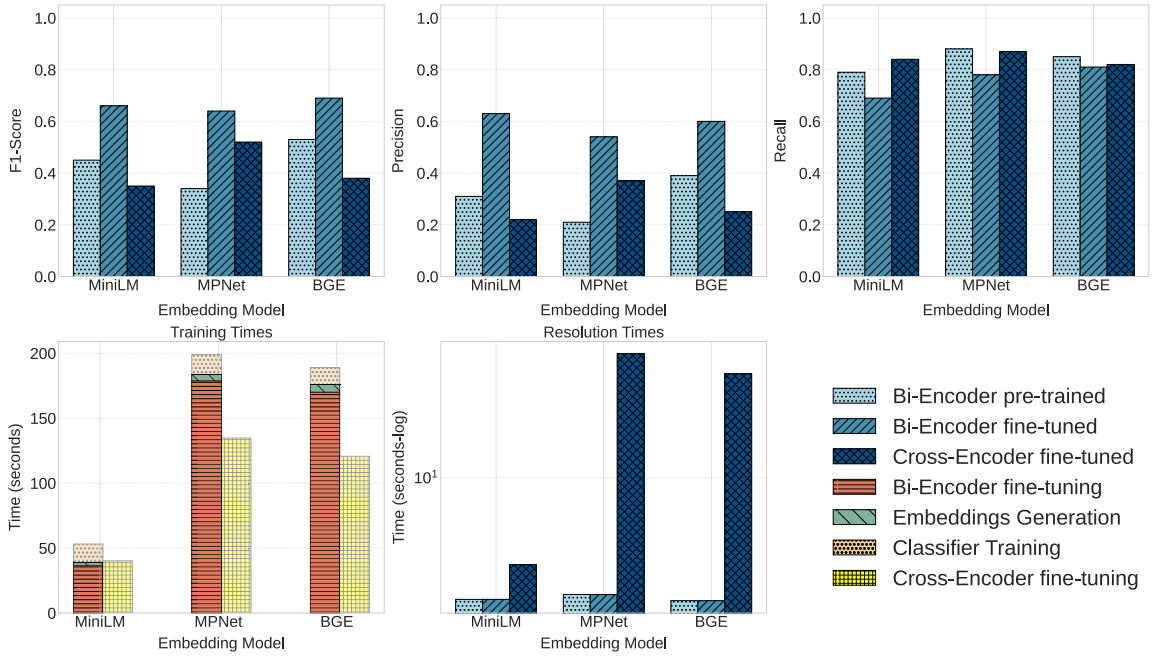


Fig. 5. Results on the Abt-Buy dataset.

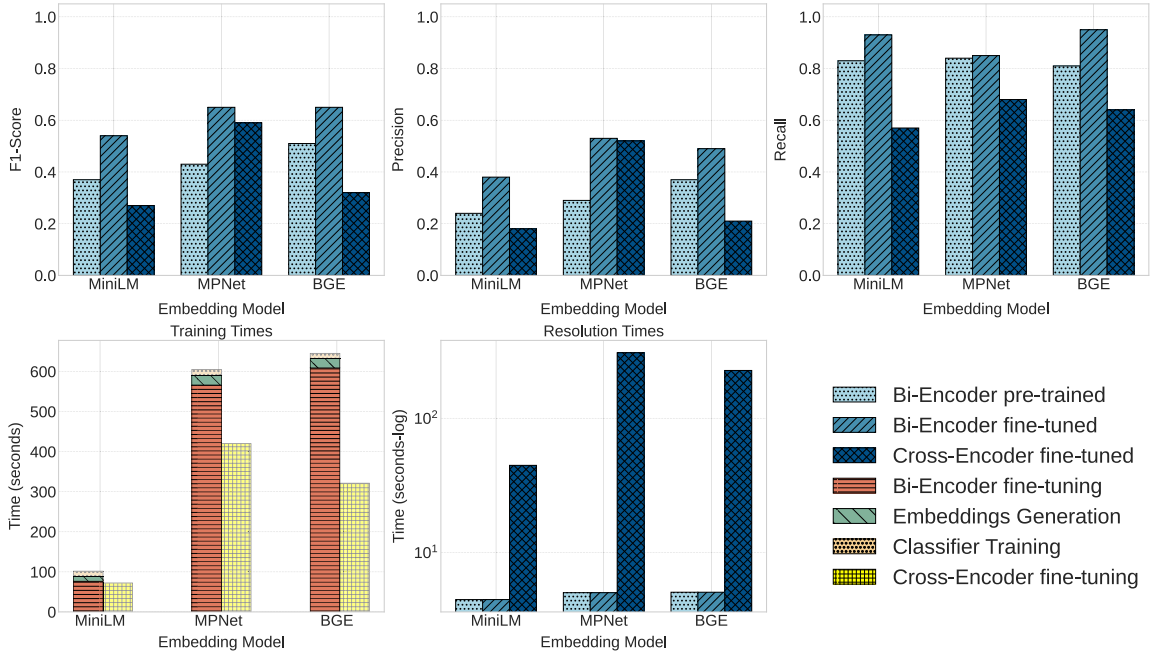


Fig. 6. Results on the Amazon-Google dataset.

triplet loss, it pulls matching entities closer together, effectively repairing the blocking recall. This allows the fine-tuned bi-encoder to retrieve and identify true matches that were geometrically inaccessible to the pre-trained model.

The Superiority of Cross-Encoders on Heterogeneous and Noisy Data. Conversely, on datasets with significant structural heterogeneity or noise, the Cross-Encoder achieved the highest accuracy. On **ACM-DBLP** (Fig. 9), the fine-tuned BGE cross-encoder reached the top F1-score of 0.98, an 18% increase over the baseline. On the **Voters** dataset (Fig. 12), which contains high levels of noise, the MPNet cross-encoder achieved the top performance of 0.82, a 9.3% improvement over the pre-trained model.

Cases of Diminishing Returns and Overfitting. On well-aligned academic datasets, fine-tuning the bi-encoder was often detrimental. On **Scholar-DBLP** (Fig. 10), the MPNet bi-encoder's F1-score dropped by nearly 10% (from 0.81 to 0.73) after fine-tuning. A similar trend was observed on the large-scale **DBLP** dataset (Fig. 11), where the MiniLM bi-encoder's score plummeted by over 16% from 0.93 to 0.78. However,

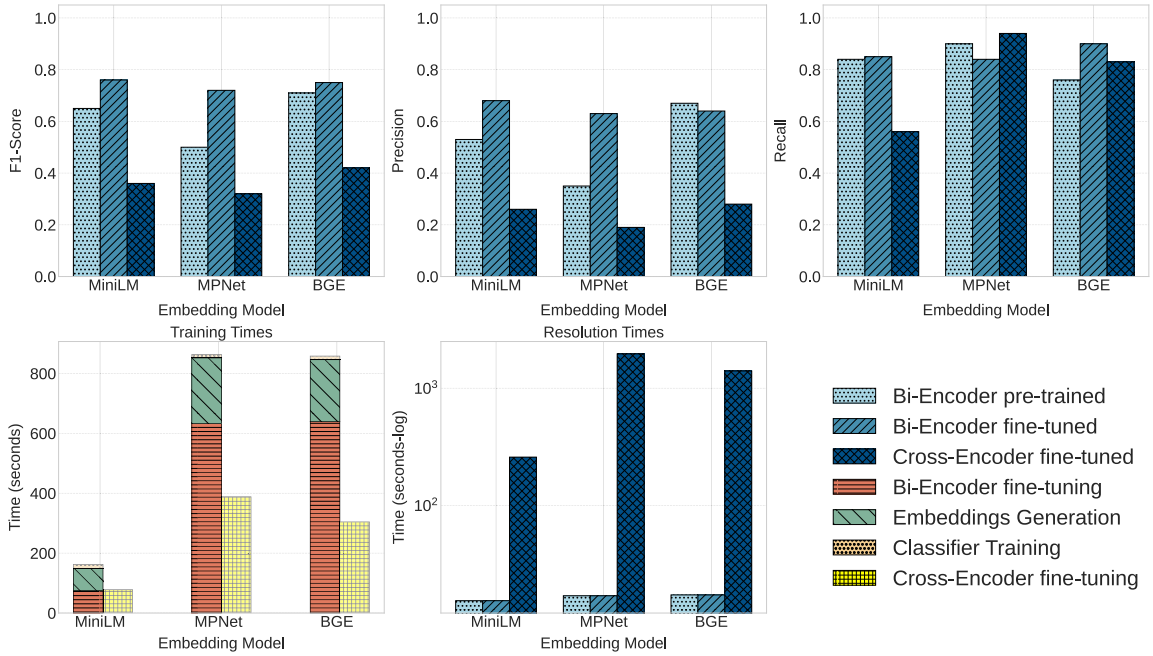


Fig. 7. Results on the Amazon-Walmart dataset.

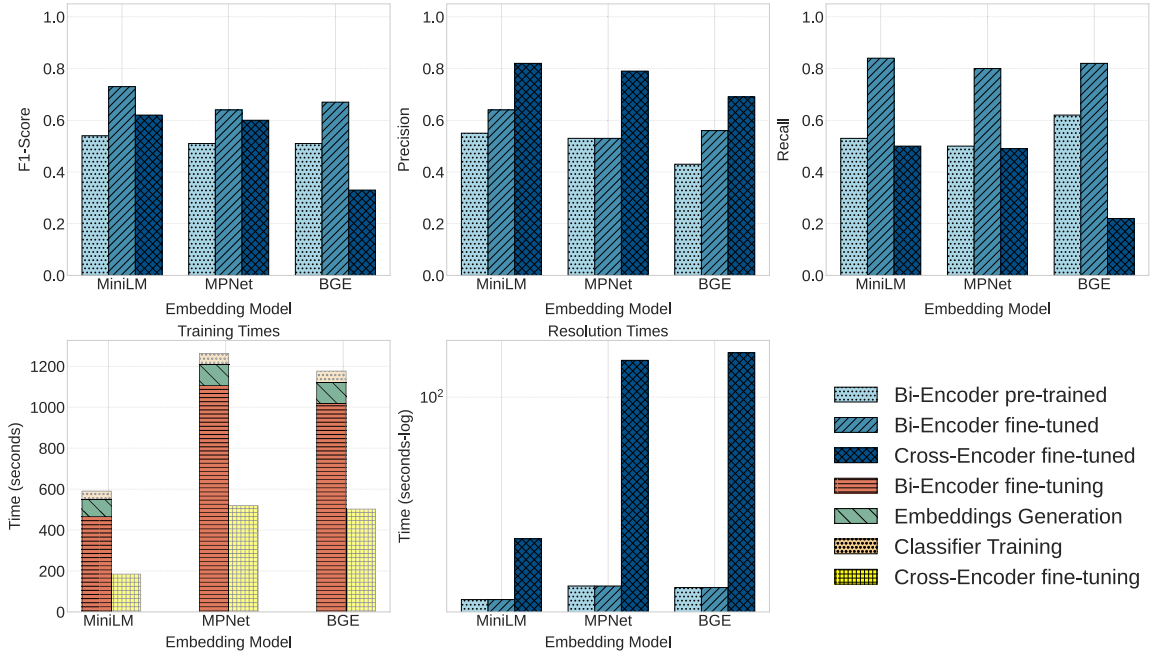


Fig. 8. Results on the IMDB-DBPEDIA dataset.

the cross-encoder was still able to extract marginal gains in these high-precision domains, with the MPNet cross-encoder reaching near-perfect F1-scores of 0.90 on Scholar-DBLP and 0.94 on DBLP.

7.2. Precision vs. recall dynamics

By analyzing precision and recall, we identified distinct patterns in how fine-tuning improves performance across the different benchmarks.

Fine-Tuning as a *Precision Fixer*. On precision-limited datasets, particularly in the e-commerce domain, the pre-trained models suffered from high false-positive rates. Fine-tuning effectively tightened the decision boundaries. For example, on **Abt-Buy** (Fig. 5), the MPNet model's precision increased by a remarkable 157% (from 0.21 to 0.54) after fine-tuning. Similarly, on **Amazon-Google** (Fig. 6), MPNet's precision improved by over 80% (from 0.29 to 0.53), and on **Amazon-Walmart**, fine-tuning corrected MPNet's precision deficit, raising it from 0.35 to 0.63.

Fine-Tuning as a *Recall Booster*. On recall-limited datasets, the pre-trained models were overly cautious. Here, fine-tuning significantly improved the identification of true matches. On **ACM-DBLP** (Fig. 9), the recall for the MPNet bi-encoder surged from 0.54 to 0.97. On

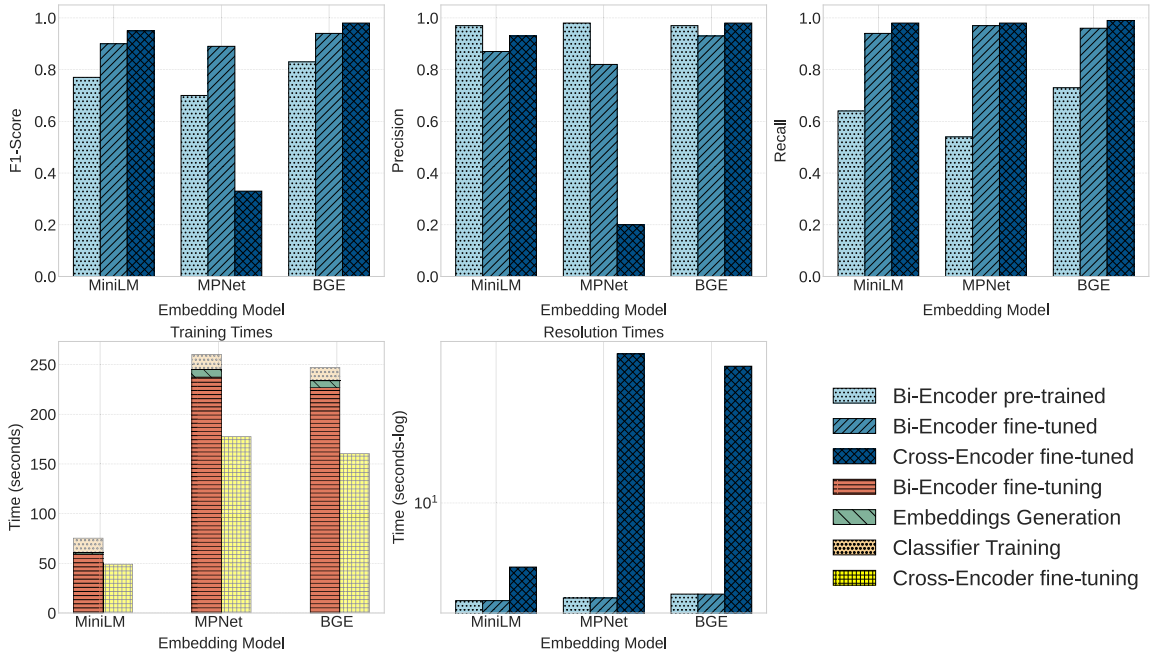


Fig. 9. Results on the ACM-DBLP dataset.

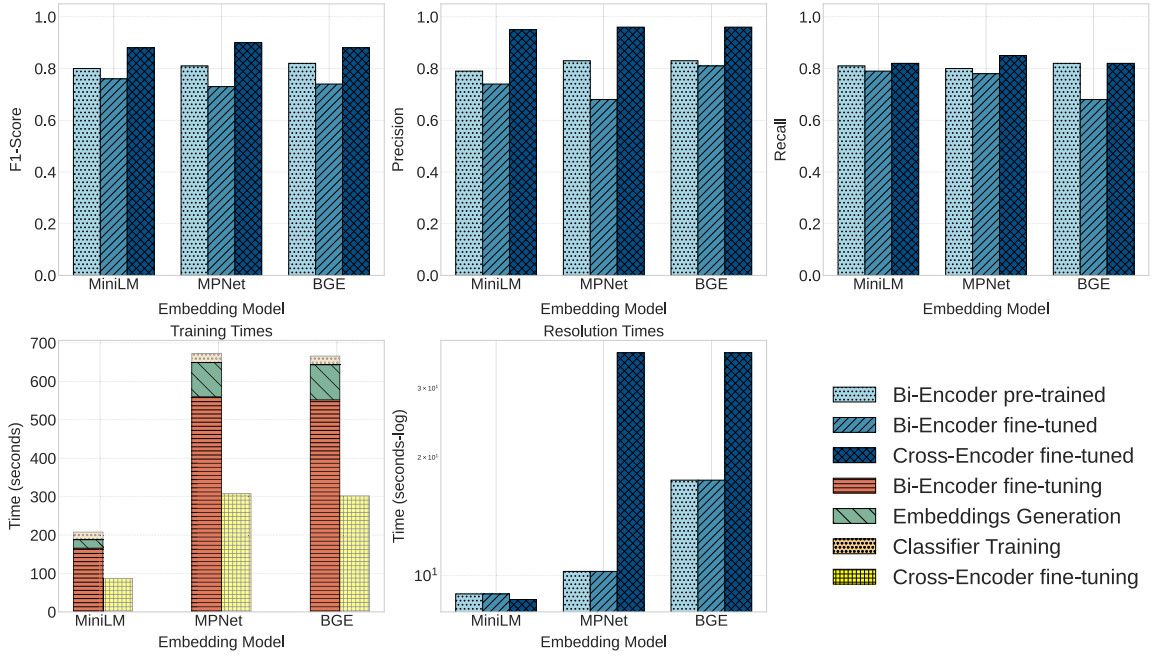


Fig. 10. Results on the Scholar-DBLP dataset.

IMDB-DBPEDIA (Fig. 8), fine-tuning the MiniLM bi-encoder increased recall from 0.53 to 0.84. Crucially, this overcomes the 60.5% retrieval bottleneck of the pre-trained model (Table 2), demonstrating that fine-tuning successfully restructured the embedding space to retrieve previously inaccessible matches. In specific cases like **Amazon-Walmart** (Fig. 7), the mechanism varied by model, with BGE's improvement driven specifically by a boost in recall from 0.76 to 0.90.

Simultaneous Improvements. On complex datasets like **Voters** (Fig. 12), fine-tuning improved both metrics simultaneously. The BGE cross-encoder saw precision increase from 0.53 to 0.70 and recall from 0.81 to 0.85, indicating the creation of a fundamentally more discriminative embedding space. Conversely, on **Scholar-DBLP** (Fig. 10) and **DBLP** (Fig. 11), the degradation in the bi-encoder performance was primarily due to a collapse in recall (e.g., MiniLM on DBLP dropping from 0.97 to 0.65), suggesting the model overfitted to the training negatives.

7.3. Efficiency analysis

A consistent trade-off between training cost and resolution speed was observed across all datasets.

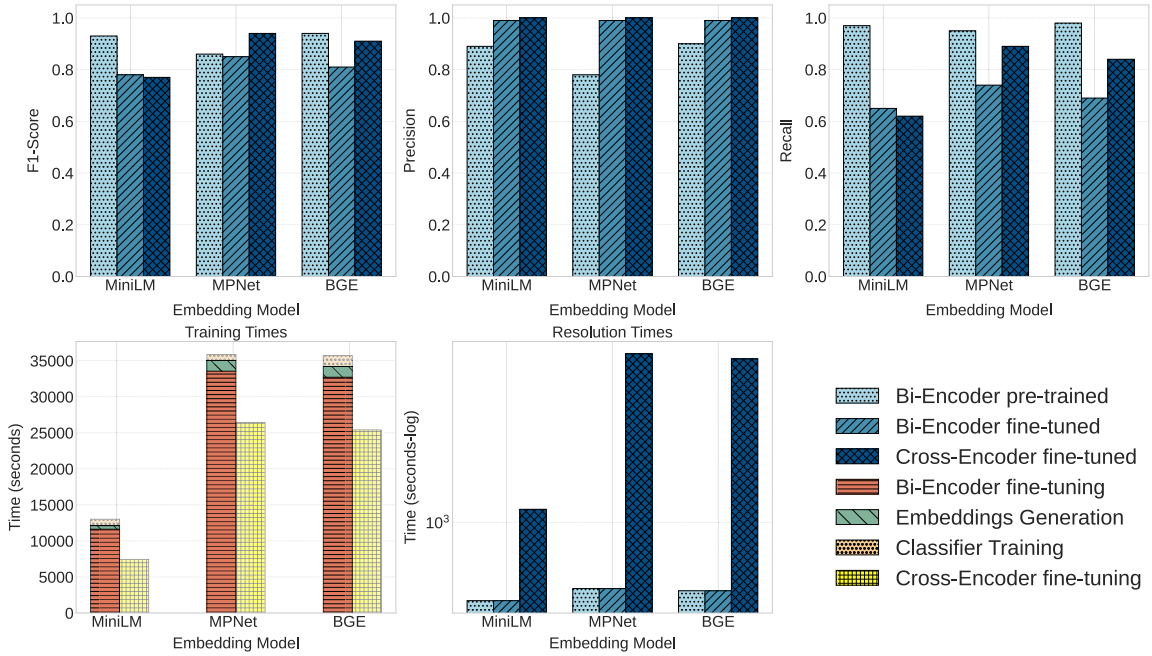


Fig. 11. Results on the DBLP dataset.

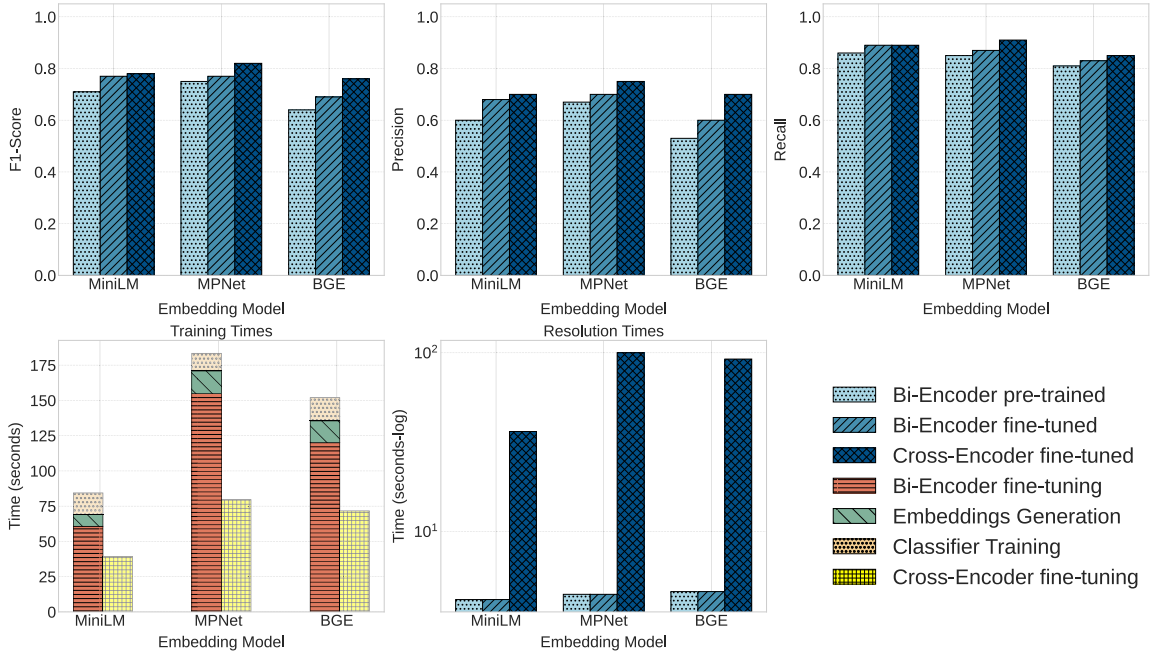


Fig. 12. Results on the Voters dataset.

Training Costs. Fine-tuning the bi-encoder is generally the most computationally intensive workflow due to the multi-stage process of triplet mining and re-embedding. On the large-scale **DBLP** dataset, the BGE bi-encoder required over 9 hours (34 720.46 seconds) to fine-tune, compared to approximately 7 hours for the cross-encoder. Similarly, on **Abt-Buy**, the MPNet bi-encoder training took approximately 200 seconds, while the cross-encoder required only 134 seconds. On **Voters**, the bi-encoder workflow (183.39 seconds for MPNet) was more than double the time of the cross-encoder (79.64s).

Resolution (Inference) Speeds. The most dramatic difference lies in resolution time. Bi-encoder architectures maintained sub-linear, highly efficient resolution times across all experiments, typically resolving datasets in seconds. For instance, on **ACM-DBLP**, resolution took roughly 4.5 seconds for all models. In stark contrast, cross-encoders were orders of magnitude slower. The MPNet cross-encoder took 35.54 seconds on **ACM-DBLP** (nearly 8 times slower), 137.83 seconds on **IMDB-DBPEDIA** (7 times slower), and over 2 hours (7227 seconds) on the large **DBLP** dataset (15 times slower). Even on smaller datasets like **Abt-Buy**,

Table 3
Ablation Study: F1-scores comparison (%) between the *cosine similarity* baseline and the *hybrid neural network classifier* using pre-trained bi-encoders and fine-tuned bi-encoders (MiniLM).

Dataset	Bi-Enc.		Bi-Enc. FT	
	Cos.	Class.	Cos.	Class.
Abt-Buy	34.2	45.5	49.5	66.1
Amazon-Google	26.1	37.3	43.4	54.2
Amazon-Walmart	43.5	65.8	58.1	76.9
ACM-DBLP	73.2	77.5	81.1	90.4
Scholar-DBLP	75.1	80.2	69.0	77.5
DBLP	84.5	93.1	71.2	78.5
IMDB-DBPEDIA	42.5	54.1	63.2	73.8
Voters	64.4	71.2	68.1	77.5

the cross-encoder required thousands of seconds compared to just $3 - 4$ seconds for the bi-encoder. This confirms that while cross-encoders offer peak accuracy on specific noisy datasets, they are computationally prohibitive for large-scale blocking tasks.

Notably, the resolution times for the pre-trained and fine-tuned bi-encoders are effectively identical. This is because fine-tuning updates the model's internal weights without altering its architecture or the dimensionality of the output embeddings. Consequently, the computational cost of the forward pass (inference) and the complexity of the subsequent vector similarity search remain constant, regardless of the model's training state.

7.4. Impact of the classification head: Ablation study

To quantify the specific contribution of the hybrid neural network classifier to the overall performance of the bi-encoder architecture, we conducted an ablation study comparing it against a standard cosine similarity baseline using MiniLM. In this baseline configuration, the embedding pairs (from both the pre-trained and fine-tuned models) were directly evaluated using cosine similarity, with the optimal threshold determined via grid search on the validation set.

Table 3 presents the comparison, where our results indicate that the hybrid neural network classifier yields a consistent performance improvement over the simple cosine similarity threshold. This performance disparity is most pronounced in the e-commerce domains–Abt-Buy, Amazon-Google, and Walmart-Amazon–where the cosine threshold approach fails to distinguish between hard negatives. In these datasets, items often share significant lexical overlap (e.g., matching brand names, categories, and technical specs) but differ by a single attribute (e.g., "Model X1" vs "Model X2"). A simple angular distance (cosine) is often insufficient to separate these vectors linearly. In contrast, the classifier, which consumes the embeddings interactions and engineered features, discussed in Section 5.1.2, effectively learns a non-linear decision boundary. Importantly, the inclusion of the difference vector allows the model to weigh specific feature dimensions differently, "learning" that discrepancies in version numbers or model codes are fatal to a match, even if the overall vector similarity is high. On Abt-Buy and Amazon-Walmart, for instance, replacing the classifier with a cosine threshold, on the fine-tuned bi-encoder, resulted in a performance drop of over 16% and 18%, respectively. This confirms that the classifier is not merely an appendage but a critical component for resolving ambiguity in high-overlap domains.

Regarding the computational cost raised by the complexity of the hybrid classifier, the time analysis (Figs. 5 – 12) reveals that the classifier training and feature generation phase accounts for less than 2% of the total pipeline duration. Given the substantial F1-score improvements observed in Table 3 (e.g., +16.6% on Abt-Buy), the inclusion of the interaction and engineered feature branches offers a highly favorable cost-benefit ratio.

7.5. Discussion and analysis

To investigate the underlying mechanisms driving the observed performance, we proceed with a detailed quantitative, qualitative, and efficiency analysis of the competing architectures.

7.5.1. Quantitative performance assessment

Our comprehensive experimental evaluation across eight diverse datasets reveals a nuanced relationship between fine-tuning, model architecture, and dataset characteristics. The results consistently demonstrate that while fine-tuning is a powerful technique for adapting pre-trained models, its effectiveness depends heavily on the nature of the *performance gap* between the general-purpose model and the specific dataset.

A primary takeaway from our experiments is the distinct trade-off between the bi-encoder and cross-encoder architectures. While the fine-tuned cross-encoder achieves state-of-the-art accuracy on structurally heterogeneous datasets (e.g., ACM-DBLP), it exhibits significant volatility on precision-sensitive tasks. As detailed later in our analysis, this architecture often regresses on e-commerce datasets compared to the bi-encoder baseline. Furthermore, its superior accuracy in specific domains comes at the cost of a resolution time that is orders of magnitude higher than that of the bi-encoder pipelines. In contrast, the fine-tuned bi-encoder consistently offers the best balance, providing a substantial and robust performance uplift over the pre-trained baseline while maintaining an exceptionally efficient resolution speed.

Our findings further suggest that the benefit of fine-tuning is directly related to the specific weakness a pre-trained model exhibits on a given dataset. We identify two primary scenarios where specialization is highly effective. On precision-limited datasets like Abt-Buy and Amazon-Google, the pre-trained models suffered from a high rate of false positives. Here, fine-tuning acted as a precision-fixer, teaching the models to be more discerning. Conversely, on recall-limited datasets like ACM-DBLP and IMDB-DBPEDIA, the pre-trained models were overly cautious. On these, fine-tuning acted as a recall-booster, teaching the models to identify a wider range of true matches.

Conversely, our experiments identify specific conditions where fine-tuning is unwarranted. On datasets like Scholar-DBLP, and the semi-synthetic Voters and DBLP datasets, the pre-trained models already exhibited strong, balanced performance. In these scenarios, the fine-tuned bi-encoder offered no benefit and, in the case of Scholar-DBLP, was even detrimental to performance. This suggests a form of overfitting, where adapting the model to a specific subset of the data caused it to lose the robust, general knowledge that made it effective in the first place. This confirms that fine-tuning is not a universally required step but a targeted intervention that is most valuable when a clear performance gap exists.

We further validated the robustness of these findings using McNemar's test. The performance gains of the fine-tuned bi-encoder over the pre-trained baseline were found to be statistically significant ($p < .05$) across all datasets where an F1-score improvement was reported (e.g., Abt-Buy, Amazon-Walmart). Conversely, on datasets like Scholar-DBLP where fine-tuning degraded performance, the drop in accuracy was also statistically significant, confirming that the negative impact was systematic rather than random noise.

The choice of the base model also plays a role, with the larger BGE and MPNet models often providing a strong baseline, but with the smaller MiniLM model frequently emerging as a surprisingly powerful and resource-efficient option after fine-tuning. These insights provide a practical guide for practitioners, suggesting that the optimal ER strategy requires a careful consideration of the specific accuracy, efficiency, and data characteristics of the task at hand.

Table 4 provides a comprehensive summary of the impact of fine-tuning by presenting the percentage difference in F1-score relative to the pre-trained baseline. The table clearly illustrates that while dataset characteristics matter, the bi-encoder architecture offers superior stability. The fine-tuned bi-encoder consistently provides a substantial

Table 4
Percentage differences (%) in F1-scores for the fine-tuned bi-encoder and cross-encoder architectures compared to the pre-trained bi-encoder baseline. The highest performance gain for each dataset is highlighted in bold. The **Average** rows summarize the performance across the eight datasets.

Dataset	Base Model	Fine-Tuned Bi-Encoder (%)	Fine-Tuned Cross-Encoder (%)
Abt-Buy	MiniLM	+46.7%	-22.2%
	MPNet	**+88.2%**	+52.9%
	BGE	+30.2%	-28.3%
ACM-DBLP	MiniLM	+16.9%	**+23.4%**
	MPNet	+27.1%	-52.9%
	BGE	+13.3%	+18.1%
Amazon-Google	MiniLM	+45.9%	-27.0%
	MPNet	**+51.2%**	+37.2%
	BGE	+27.5%	-37.3%
Amazon-Walmart	MiniLM	+16.9%	-44.6%
	MPNet	**+44.0%**	-36.0%
	BGE	+5.6%	-40.8%
DBLP	MiniLM	-16.1%	-17.2%
	MPNet	-1.2%	**+9.3%**
	BGE	-13.8%	-3.2%
IMDB-DBPEDIA	MiniLM	**+35.2%**	+14.8%
	MPNet	+25.5%	+17.6%
	BGE	+31.4%	-35.3%
Scholar-DBLP	MiniLM	-5.0%	**+10.0%**
	MPNet	-9.9%	+11.1%
	BGE	-9.8%	+7.3%
Voters	MiniLM	+8.5%	+9.9%
	MPNet	+2.7%	+9.3%
	BGE	+7.8%	**+18.8%**
Aggregated Performance (Average across 8 datasets)			
Average	MiniLM (Base F1: 65.3)	**+18.6%** (Avg F1: 73.8)	-6.6% (Avg F1: 62.3)
	MPNet (Base F1: 61.3)	**+28.5%** (Avg F1: 73.6)	+6.1% (Avg F1: 62.8)
	BGE (Base F1: 68.6)	**+11.5%** (Avg F1: 74.3)	-12.6% (Avg F1: 62.3)

performance uplift on product-matching datasets, yielding an F1-score increase of up to 88.2% on Abt-Buy. While it showed some degradation on already-aligned academic datasets, its average performance gain across all eight benchmarks was consistently high (e.g., +28.5% for MPNet). In contrast, the cross-encoder exhibits a high-risk, high-reward behavior. While it achieves top-tier gains on specific heterogeneous tasks (e.g., +23.4% on ACM-DBLP for MiniLM), it suffers from severe volatility. As shown in the Aggregated Performance rows of Table 4, the fine-tuned cross-encoders for MiniLM and BGE actually yielded a negative average improvement (−6.6% and −12.6% respectively) due to significant regressions on high-precision e-commerce datasets. Even the robust MPNet cross-encoder only achieved a modest +6.1% average gain, far below the +28.5% of its bi-encoder counterpart.In summary, our experimental quantitative analysis across eight diverse datasets reveals three key insights:

1. **Identifying a Winner:** Contrary to the assumption that cross-encoders are always the accuracy leaders, our aggregated results identify the **fine-tuned MPNet bi-encoder** as the optimal default architecture. It delivers the highest average stability (Avg F1: 73.6) and the largest consistent performance boost (+28.5%) without the extreme volatility or computational overhead of the cross-encoder.
2. **Targeted Intervention:** We confirm that fine-tuning is most effective as a targeted intervention. On datasets where pre-trained models suffered from low precision (e.g., Abt-Buy) or low recall (e.g., ACM-DBLP), fine-tuning yielded significant gains. However, on datasets where the pre-trained models were already strong (e.g., Scholar-DBLP), it proved to be an unnecessary step, occasionally leading to overfitting.
3. **Model Efficiency:** Our results challenge the assumption that larger models are always superior. The smaller **MiniLM** bi-encoder, when properly fine-tuned, achieved an average F1-score of 73.8, virtually identical to the larger MPNet (73.6) and BGE (74.3) models. This presents a highly resource-efficient option for practitioners, offering state-of-the-art performance at a fraction of the inference cost.

7.5.2. *Qualitative error analysis*

To better understand the performance characteristics driving the quantitative results, we conducted a manual inspection of false positives and false negatives across three representative datasets: Abt-Buy, DBLP-ACM, and Voters.

The primary failure mode for bi-encoders was the inability to distinguish between entities that share high lexical overlap but differ in a single, critical token. For instance, in the Abt-Buy dataset, the bi-encoder frequently failed to distinguish between product variations such as “HP ProBook 450 G3” and “HP ProBook 450 G4.” The pooling operation in bi-encoders tends to "smooth out" these small but critical token differences, pushing their vectors too close together.

Conversely, the cross-encoder performance was highly inconsistent. While it successfully resolved complex heterogeneous matches (e.g., in ACM-DBLP), it exhibited a tendency toward *overfitting* on product data. In datasets like Abt-Buy, the cross-encoder frequently penalized valid matches due to minor, irrelevant discrepancies in descriptions (e.g., shipping details included in one record but not the other), leading to the significant drops in recall observed in Table 4. Furthermore, in the Voters dataset, it occasionally predicted matches based on generic shared tokens (e.g., matching "Smith" and "Main St") when discriminative fields were empty, highlighting a susceptibility to noise that the bi-encoder’s structured representation avoided.

On structured datasets like DBLP-ACM, errors were rare and often due to ambiguous abbreviations (e.g., "Proc. of VLDB" vs "PVLDB"). The fine-tuned models showed a remarkable ability to learn these specific patterns compared to pre-trained baselines, confirming that gains in these domains stem from learning specific vocabulary mismatches.

7.5.3. Efficiency and scalability considerations

While the cross-encoder architecture theoretically offers a more expressive attention mechanism, our results demonstrate that its practical utility is severely limited by its computational cost and volatility. The latency of the Cross-Encoder is primarily driven by the full self-attention mechanism over the concatenated input sequence, which scales quadratically with sequence length. Unlike bi-encoders, which allow for the offline pre-computation of embeddings, cross-encoders must process every candidate pair online, making the cost proportional to the number of candidates generated by the blocking phase.

To mitigate these costs in production environments, several optimization strategies can be employed. Batching is the most immediate optimization; by processing multiple candidate pairs in parallel, the overhead of memory access is amortized, significantly increasing throughput on GPU hardware. Further acceleration can be achieved through quantization (reducing model weights from FP32 to INT8), which typically yields $2-4x$ speedups with negligible accuracy loss, and pruning, which removes redundant attention heads. Additionally, Knowledge Distillation offers a promising avenue, where a high-performing cross-encoder effectively transfers complex matching logic into a faster architecture.

In terms of practical thresholds, our findings suggest that cross-encoders are computationally prohibitive for the initial blocking stage of ER on datasets exceeding a few thousand records. They are most effectively deployed in a re-ranking capacity, applied only to a small set of high-confidence candidates (e.g., top-50) retrieved by a bi-encoder. However, given our finding that the **fine-tuned bi-encoder** achieves comparable or superior accuracy on most datasets (Table 4), the utility of the cross-encoder re-ranking step is questionable for general-purpose applications compared to a well-optimized bi-encoder pipeline.

7.5.4. Impact of domain characteristics and data alignment

Our experimental results reveal a sharp dichotomy in how different datasets respond to fine-tuning, a phenomenon largely explained by the distributional alignment between the target domain and the corpora used to pre-train the underlying language models.

Datasets such as DBLP and Scholar-DBLP consist primarily of academic citations, titles, and author names. These records are structurally consistent and lexically similar to the massive text corpora on which models like MPNet were pre-trained (e.g., Wikipedia, BooksCorpus, and academic subsets of Common Crawl). Consequently, the pre-trained models already possess a high-quality semantic representation of this domain "out-of-the-box." In these scenarios, the manifold of the target data is already well-covered by the base model, meaning that fine-tuning offers diminishing returns–or, as observed in Scholar-DBLP, may even degrade performance by overfitting to specific blocking artifacts rather than general matching logic.

In contrast, datasets like Abt-Buy (e-commerce) and Voters (public records) exhibit significant domain shift. E-commerce data is characterized by short, ungrammatical segments, brand-specific abbreviations, and dense clusters of alphanumeric product codes (e.g., "Sams. Gal. S4" vs "Samsung Galaxy S4"). Similarly, the Voters dataset contains high levels of noise, including OCR errors, missing fields, and non-standard formatting. These patterns are under-represented in standard pre-training corpora. Therefore, the pre-trained embeddings are often scattered or misaligned for these specific entities. Fine-tuning becomes essential here not merely to learn a matching function, but to fundamentally restructure the embedding space, pulling these domain-specific variations into proximity. This explains why the largest performance gains from fine-tuning the bi-encoder are consistently observed in these noisy or highly specialized domains, where the model must learn to bridge the gap between noisy surface forms and latent semantic meaning.

8. Conclusions and future work

This paper presented a comprehensive experimental evaluation of deep learning architectures for Entity Resolution, specifically investigating the trade-offs between pre-trained baselines, fine-tuned bi-encoders, and fine-tuned cross-encoders across eight diverse datasets.

Our findings challenge the prevailing assumption that the computationally expensive cross-encoder architecture is the universal gold standard for accuracy. While effective on structurally heterogeneous data, we demonstrated that it exhibits significant volatility, yielding a *negative average performance improvement* on efficient models due to overfitting on precision-sensitive e-commerce tasks. Furthermore, its prohibitive computational cost–orders of magnitude higher than the bi-encoder–renders it impractical for large-scale blocking.

In contrast, we identified the fine-tuned bi-encoder (specifically utilizing MPNet) as the robust, optimal default for practitioners. It delivers the highest consistent performance gain (Average F1 increase of +28.5%) and superior stability, all while maintaining sub-linear retrieval speeds suitable for real-time applications. Additionally, we demonstrated that model size is not the sole determinant of success; the compact MiniLM bi-encoder, when properly fine-tuned with hard negatives, frequently rivals or outperforms larger state-of-the-art models, offering a highly resource-efficient alternative.

Crucially, we conclude that fine-tuning is not a panacea but a targeted intervention. It acts as a necessary "precision-fixer" for noisy, unaligned domains (e.g., Abt-Buy), but can be detrimental to performance on already-aligned academic datasets (e.g., Scholar-DBLP), where it may induce overfitting.

Future research should investigate: (1) hybrid architectures that dynamically offload only low-confidence pairs from a bi-encoder to a cross-encoder re-ranker to optimize the accuracy-efficiency frontier; (2) The potential of generative LLMs for zero-shot cleaning and data augmentation to bootstrap training sets; and (3) The development of noise-robust loss functions for noisy real-world data.

CRediT authorship contribution statement

Dimitrios Karapiperis: Writing – original draft, Software, Conceptualization; **Leonidas Akritidis:** Writing – original draft, Methodology; **Panayiotis Bozanis:** Supervision.

Data availability

The source code and datasets used in this experimental evaluation are available at the following repository: https://github.com/dimkar121/er_classifier.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: Association for Computational Linguistics: Human Language Technologies, 2019, pp. 4171–4186.
[2] B. Li, Y. Miao, Y. Wang, Y. Sun, W. Wang, Improving the efficiency and effectiveness for BERT-based entity resolution, in: AAAI, 2021.
[3] D. Karapiperis, C. Tjortjis, V. Verykios, LSBlock: A hybrid blocking system combining lexical and semantic similarity search for record linkage, in: ADBIS, 2025, pp. 131–146.
[4] A. Brinkmann, R. Shraga, C. Bizer, SC-Block: Supervised Contrastive Blocking within Entity Resolution Pipelines, 2023.
[5] Y. Li, J. Li, Y. Suhara, A. Doan, W.C. Tan, Deep entity matching with pre-Trained language models, PVLDB 14 (1) (2020) 50–60.
[6] U. Brunner, K. Stockinger, Entity matching with transformer architectures - A step forward in data integration, in: EDBT, 2020, pp. 463–473.
[7] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, V. Raghavendra, Deep learning for entity matching: a design space exploration, in: SIGMOD, 2018, pp. 19–34.
[8] D. Zhang, Y. Nie, S. Wu, Y. Shen, K.L. Tan, Multi-Context attention for entity matching, in: WWW, 2020, pp. 2634–2640.

[9] C. Fu, X. Han, J. He, L. Sun, Hierarchical matching network for heterogeneous entity resolution, in: IJCAI, 2020.
[10] R. Chen, Y. Shen, D. Zhang, GNEM: A generic one-to-Set neural entity matching framework, in: WWW, 2021.
[11] B. Genossar, R. Shraga, A. Gal, FlexER: flexible entity resolution for multiple intents, SIGMOD 1 (1) (2023).
[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, I. Polosukhin, 2017. arXiv:1706.03762
[13] W. Zhang, H. Wei, B. Sisman, X.L. Dong, C. Faloutsos, D. Page, Autoblock: a hands-off blocking framework for entity matching, in: WSDM, 2020.
[14] S. Thirumuruganathan, H. Li, N. Tang, M. Ouzzani, Y. Govind, D. Paulsen, G. Fung, A. Doan, Deep learning for blocking in entity matching: a design space exploration, PVLDB 14 (11) (2021) 2459–2472.
[15] L. Wu, F. Petroni, M. Josifoski, S. Riedel, L. Zettlemoyer, Scalable Zero-shot Entity Linking with Dense Entity Retrieval, 2019.
[16] S. Suri, I.F. Ilyas, C. Ré, T. Rekatsinas, EMBER: No-Code context enrichment via similarity-Based keyless joins, PVLDB 15 (3) (2022) 699–712.
[17] R. Wang, Y. Li, J. Wang, Sudowoodo: contrastive self-supervised learning for multi-purpose data integration and preparation, in: ICDE, 2023, pp. 1502–1515.
[18] K. Qian, L. Popa, P. , Sen, Active Learning for Large-Scale Entity Resolution, in: CIKM, 2017, pp. 1379–1388.
[19] R. Wu, S. Chaba, S. Sawlani, X. Chu, S. Thirumuruganathan, ZeroER: entity resolution using zero labeled examples, in: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data, 2020, p. 1149–1164.
[20] Y. Nafa, Q. Chen, Z. Chen, X. Lu, H. He, T. Duan, Z. Li, Active deep learning on entity resolution by risk sampling, Know.-Based Syst. 236 (2022).
[21] A. Jain, S. Sarawagi, P. Sen, Deep indexed active learning for matching heterogeneous entity representations, Proceed. VLDB Endowment (PVLDB) 15 (1) (2021) 31–45.
[22] S. Galhotra, D. Firmani, B. Saha, D. Srivastava, Efficient and effective ER with progressive blocking, VLDB J. 30 (2021) 537–557.
[23] J. Maciejewski, K. Nikoletos, G. Papadakis, Y. Velegrakis, Progressive entity matching: a design space exploration, in: SIGMOD, 3, 2025, pp. 1–25.
[24] Z. Zhang, W. Zeng, J. Tang, H. Huang, X. Zhao, Active in-context learning for cross-domain entity resolution, Inf. Fusion 117 (2025).
[25] A. Zeakis, G. Papadakis, D. Skoutas, M. Koubarakis, AvengER: ensembling and fine-Tuning LLMs for SELECT prompts in entity resolution, in: ESWC, 2025, p. 301–320.
[26] A. Zeakis, G. Papadakis, D. Skoutas, M. Koubarakis, Pre-trained embeddings for entity resolution: an experimental analysis, PVLDB 16 (9) (2023) 2225–2238.
[27] D. Karapiperis, G. Feretzakis, V.S. Verykios, An experimental evaluation of pre-trained models for efficient and accurate record linkage, in: IEEE IISA, 2025.
[28] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, M. Zhou, Deep self-attention distillation for task-agnostic compression of pre-trained transformers, in: Neurips, 2020.
[29] K. Song, X. Tan, T. Qin, J. Lu, T.Y. Liu, MPNet: Masked and permuted pre-training for language understanding, in: NeurIPS, 2020.
[30] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, C-Pack: Packaged Resources To Advance General Chinese Embedding, 2023. arXiv:2309.07597
[31] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R.R. Salakhutdinov, Q.V. Le, XLNet: Generalized autoregressive pretraining for language understanding, in: NeuIPS, 2019.
[32] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, RoBERTa: A Robustly Optimized BERT Pretraining Approach, 2019.
[33] V. Christophides, V. Efthymiou, K. Stefanidis, Entity Resolution in the Web of Data, Morgan & Claypool Publishers, 2015.
[34] G. Papadakis, J. Svirsky, A. Gal, T. Palpanas, Comparative analysis of approximate blocking techniques for entity resolution, PVLDB (2016) 684–695.
[35] D. Karapiperis, C. Tjortjis, V.S. Verykios, A suite of efficent randomized algorithms for streaming record linkage, IEEE TKDE 36 (7) (2024) 2803–2813.
[36] L. Gagliardelli, G. Papadakis, G. Simonini, S. Bergamaschi, T. Palpanas, GSM: A generalized approach to supervised meta-blocking for scalable entity resolution, Inf. Systems 120 (2024).
[37] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani, N. Tang, Distributed representations of tuples for entity resolution, PVLDB 11 (11) (2018) 1454–1467.
[38] D. Karapiperis, C. Tjortjis, V. Verykios, A randomized blocking structure for streaming record linkage, in: PVLDB, 2023, p. 2783–2791.
[39] D. Karapiperis, V.S. Verykios, An LSH-based blocking approach with a homomorphic matching technique for privacy-Preserving record linkage, IEEE TKDE 27 (4) (2015) 909–921.
[40] M. Douze, A. Guzhva, C.-H. Deng, J. Johnson, G. Szilvasy, P. Mazaré, M. Lomeli, L. Hosseini, H. Jégou, The Faiss library, 2024.
[41] Y.A. Malkov, D.A. Yashunin, Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, IEEE TPAMI 42 (4) (2018) 824–836.