Balancing Efficiency and Quality in LLM-Based Entity Resolution on Structured Data

Navapat Nananukul and Mayank Kekriwal

Information Sciences Institute. University of Southern California. CA, United states 90292

Abstract. Entity Resolution (ER) is the problem of automatically determining when two or more entities refer to the same underlying entity. ER has been researched for over fifty years across multiple domains (including healthcare, e-commerce, and census data). In graph-based applications, such as deduplicating identities across (or even within) social media platforms, as well as knowledge graphs, ER can be particularly important. Traditionally, ER was a difficult problem both within Artificial Intelligence (AI) and in databases, owing to the quadratic $O(n^2)$ complexity of comparing n entities to each other, given one or more graphs with n total nodes. However, recent emergence of large language models (LLMs) allow us to address the challenges of ER as an AI problem, but a clear framework for applying LLMs in a cost-effective way remains an open issue. In this paper, we present such a framework and validate it through early experiments on real-world ER benchmarks. The framework is LLM-agnostic and is premised on assumptions that resemble pragmatic real-world requirements.

Keywords: Entity resolution, identity matching, knowledge graphs, blocking, efficiency, large language models.

1 Background

Entity resolution (ER) is the algorithmic problem of determining when two or more entities refer to the same underlying entity. With increased growth of social media and user data on the Web, and with users typically having accounts (sometimes, more than one) across multiple internet platforms, ER has become central in applications ranging from customer analytics to data integration. Figure 1 provides an intuitive illustration of the problem. Many classes of ER solutions [4,5,7,11,14] have been proposed over the decades, with deep learning [6,16], and most recently, large language models (LLMs) yielding improved performance on challenging benchmarks [18, 19, 21].

ER is a problem with inherently quadratic complexity in the worst case. The simplest case (and one that is theoretically sufficient for analyzing other variants) arises when a single network with n nodes needs to be *deduplicated*. Given a vertex-set V with n nodes, suppose that a perfect constant-time oracle $f : V \times V \rightarrow \{True, False\}$ exists that, given two nodes from V, outputs a

decision on whether two input entities are semantic duplicates[1] (True) or not (False). Even given such a *similarity* function, which in practice is imperfect and based on some form of machine learning, all pairs of entities would have to be compared in the worse case, yielding $O(n^2)$ complexity. Conducting so many comparisons is wasteful in real-world scenarios because most nodes either don't have a duplicate, and those that do tend to have some small number of duplicates. For this reason, ER has previously been referred to as a *micro-clustering* problem [3].

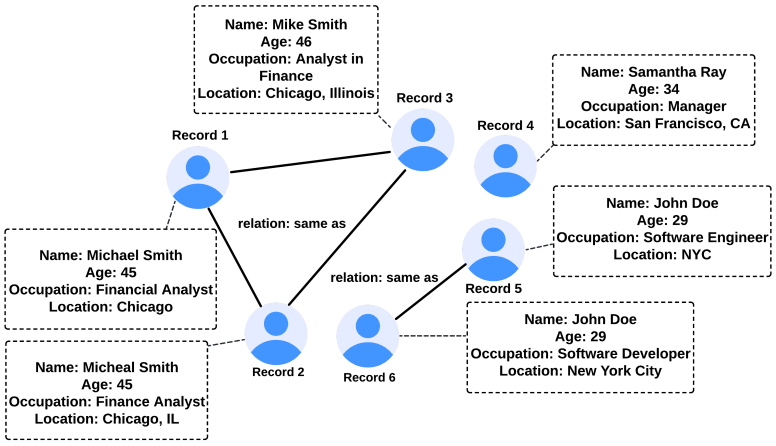

Fig. 1. Illustration of the ER problem where the same user (indicated by the *same as* edge) has multiple records with different semantic representation.

To mitigate the quadratic $O(n^2)$ complexity of comparing n entities to each other, a typical ER architecture comprises two steps: *blocking* and *similarity*. Blocking is the process of using an inexpensive algorithm to cluster entities into 'blocks', such that entities sharing a block have higher probability of being duplicates. Figure 2 illustrates a simple blocking scenario where blocks are created based on matching last names. Following blocking, the similarity step only compares pairs of entities within the *same* blocks. As we noted earlier, LLMs like GPT-3.5 and GPT-4 have emerged as a viable similarity step, but are too expensive to invoke for hundreds of thousands of entity pairs, which is not uncommon (even post-blocking) for reasonably-sized datasets.

Motivated by this challenge, we propose a novel cost-sensitive approach for balancing efficiency and quality in LLM-based ER on structured data. An important aspect of the framework is that, rather than only consider the LLM as the

[1] We clarify, as also suggested by the figure, that the entities are not *syntactic* duplicates, otherwise the problem would neither require AI, nor a sophisticated approach to reduce quadratic computation. Deduplication and ER are *semantic* processes at their core i.e., entities that are actually the same 'look' different.

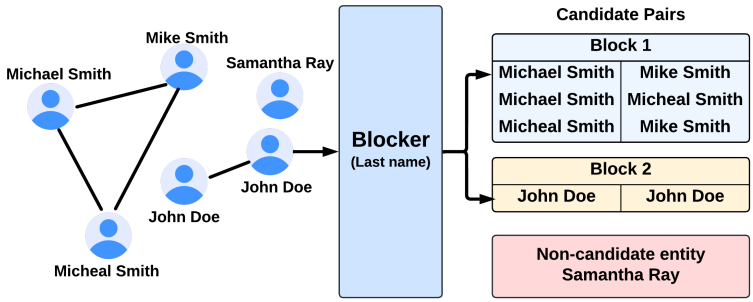

Fig. 2. An illustration of the blocking process in entity resolution, where blocks are created based on matching last names. Entities are grouped into blocks according to their last names, resulting in Block 1 containing a permutation of candidate pairs with the last name "Smith" and Block 2 containing candidate pairs with "Doe." The entity "Samantha Ray," which does not share a last name with any other entity, is classified as a non-candidate entity and is excluded from further comparisons.

similarity function, we consider using *two* similarity functions: one high-quality but expensive (like an LLM, or in the extreme case, an oracle), and the other, moderate quality but much cheaper. The question now becomes: which candidate pairs (obtained after blocking) should we prioritize sending into the former, rather than accepting the output of the 'default' latter method? Intuitively, the second case will apply to the vast majority of pairs in limited-budget settings, which any prioritization function must be sensitive to.

The proposed framework is designed to take a budget as input and, rather than treat all blocks equally, uses an *entropy-based* methodology for block prioritization. Next, the feedback from the LLM is used to further refine the prioritization score to better navigate the cost-benefit tradeoff. We conduct detailed experiments across three ER benchmarks, using two blocking baselines, and show that the framework significantly outperforms two competitive prioritization baselines by achieving competitive ER performance 30% faster than traditional blocking methods. Incorporating feedback from the LLM is further found to boost efficiency by 5% compared to the normal entropy-based approach on some ER benchmarks.

2 Related Work

Blocking is a crucial step in reducing the quadratic complexity of entity resolution (ER) and has been approached through various methodologies [27]. Originally developed in the late 1960s [9], blocking workflows involve block building [9, 24, 42], block cleaning [23, 25], and comparison cleaning [26], organizing records into blocks based on derived signatures. However, ER datasets' growing complexity and size have introduced new challenges and limitations [10] where budget constraints have become important. This leads to a new entity resolution

technique called *progressive ER*. Progressive ER integrates additional components into existing blocking techniques to enhance early recall performance [30]. Recent works [1, 2] explored methods for handling large datasets with limited time and computational resources. [31] proposed a system to minimize missed duplicates by estimating the matching likelihood of attributes in unresolved candidate pairs. These studies have opened new avenues for improving ER performance within constrained budgets.

The recent emergence of large language models (LLMs) has demonstrated state-of-the-art performance in many fields [8, 15, 35]. LLMs such as ChatGPT represent a significant milestone in the field. Recently, several notable LLMs have been developed and released, including OpenAI's ChatGPT [22], Meta's LLaMA [36], and Google's Gemini [34]. LLMs have been applied in various domains, including natural language understanding, machine translation, question answering, and text summarization [20, 40, 41]. They are also being used in specialized applications such as legal document analysis, medical research, and educational tools [13, 33].

In ER, evidence suggests that LLMs can offer promising results in unsupervised ER. LLMs can determine whether two records refer to the same entity [28] using their pre-trained knowledge of products across the internet. This has led to a growing body of research integrating LLMs' capability with ER [18]. Moreover, prompt engineering [12,38,43] also shows that LLMs' performance is significantly influenced by how different information or examples are given in the prompt. Recent work [21] illustrated that prompt engineering techniques influence ER performance at different costs depending on the prompts' complexity. However, one obvious trade-off when using state-of-the-art commercial LLMs still persists. Creating or resolving thousands of blocks can be expensive and impractical in a research setup. Existing works related to using LLMs-ER field focus on finding methods to integrate LLMs with ER or maximize ER performance. To our knowledge, no existing work has explored cost-efficient methods of employing ER with LLMs for entity matching. Our proposed methods aim to achieve cost-efficient ER by combining a progressive ER technique with a token-based entropy sorting approach to maximize ER performance under a budget.

3 Framework

Figure 3 illustrates the schematic of the entropy-based block prioritization approach. We integrate the entropy-based approach into the traditional blocking process to have high recall early when having a limited entity-matching budget. First, this approach tokenized attributes in an ER dataset, yielding a probability distribution of tokens in the ER dataset. Second, entropy scores are calculated at an entity level using the probability distribution from the first step. Third, candidate pairs are generated and sorted based on their entropy scores, from lowest to highest, to prioritize more homogeneous pairs, expected to have a higher potential for matching. Lastly, the probability distribution can be updated while entity matching is performed. We implemented a feedback loop to recalculate the

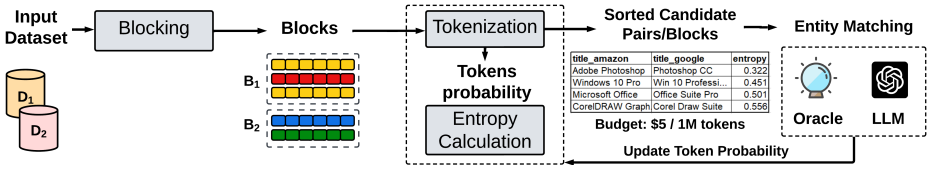

Fig. 3. An illustrative example of how the entropy-based block prioritization approach fits in an ER workflow. This approach begins by tokenizing the attributes of entities within blocks to establish a probability distribution. Next, it calculates entropy scores for all entities, sorting them from lowest to highest. Using this sorted list of entities, candidate pairs are formed according to a predetermined blocking method. After all entropy scores are assigned to all entities, blocks are sorted using the average scores of the candidate pairs. Lastly, candidate pairs are sent to a theoretically constructed Oracle or an LLM to resolve. The token probability pool is then updated based on the remaining candidate pairs.

probability distribution and the entropy scores using the remaining candidate pairs. Next, we provide specific details, using a practical example.

3.1 Prioritization using Entropy

As a first step, entropy scores are computed for all entities in blocks to quantify the 'information' in each entity in the block. We use the attributes of entities (e.g., names and descriptions) to calculate entropy scores. The entropy score E is based on the token probability distributions and is given by the formula:

$$E = -\sum_{\text{token} \in \text{tokens}} p(\text{token}) \log_2 p(\text{token})$$

where $p(\text{token})$ is the probability of occurrence of each token, calculated as the frequency of the token divided by the total token count across inside a block.

The entropy scores are subsequently utilized to reorder the entities within each block. Additionally, they serve as a key variable for traditional blocking methods, such as Progressive Sorted Neighborhood (PSN) and Hierarchical Record Partition (HRP), in generating the final list of candidate pairs. The following subsection provides a detailed explanation of these selected blocking methods and illustrates how entropy scores are integrated using a practical example.

3.2 Blocking Methods

We selected established blocking techniques commonly utilized in progressive entity resolution frameworks. Notable methods, including the Progressive Sorted Neighborhood (PSN) and Hierarchical Record Partition (HRP), systematically reorder candidate pairs using sorting keys derived from dataset attributes. This

makes them excellent blocking benchmarks for integrating the entropy score and using it as a sorting key.

Progressive Sorted Neighborhood (PSN) [37,39] initially sorts the data using a predefined sorting key and subsequently compares records that fall within a specified sliding window. The underlying rationale for this method is that records positioned closely in the sorted order are more likely to be duplicates than those that are distantly placed.

Hierarchical Record Partition (HRP) [37] organizes records into a structured hierarchy of partitions, where each level represents groups of potentially matching entities at varying degrees of granularity. The most granular partition at the hierarchy's base applies the strictest criteria for grouping, resulting in tightly clustered entities. The closer to the root of the hierarchy, the coarser each partition becomes. For example, at level 1, entities might be grouped based on a single matching token; at level 2, the criteria might require two matching tokens to align entities as potentially similar.

Entropy Score Application To integrate entropy scores into a blocking method, we illustrate the application using PSN as an example. Assuming we have a block $X = \{e_1, e_2, e_3, e_4\}$, where the entities must be progressively resolved. Initially, entities within X are not sorted and are randomly placed within a block. Suppose the entities, after being sorted by entropy, are listed as $[e_3, e_2, e_4, e_1]$ with corresponding entropy values indicating $[e_3, e_2]$ as having lower entropy, suggesting closer similarity or higher match potential. The $\text{Rank}(e_i)$ of each entity e_i in the sorted list is determined, and the distance between any two entities e_i and e_j, $\text{distance}(e_i, e_j)$, is calculated as $|\text{Rank}(e_i) - \text{Rank}(e_j)|$. For example, if $\text{Rank}(e_2) = 2$ and $\text{Rank}(e_4) = 3$, then $\text{distance}(e_2, e_4) = |2 - 3| = 1$.

The sorting mechanism starts resolving the pairs in X in an ascending order based on their distance measures within window size w. For instance, it will resolve the pair (e_3, e_2) before (e_3, e_4) because the former pair has a smaller distance and potentially lower combined entropy, implying a higher likelihood of being duplicates. This process continues until a stopping condition is met, such as when the rate of newly discovered duplicate pairs drops below a certain threshold or until all pairs whose distance value is less than a specified window size w have been compared.

3.3 Feedback Mechanism

We developed a feedback mechanism that dynamically updates entropy scores during the entity resolution (ER) process. Initial entropy scores are computed using all tokens from unresolved candidate pairs at the start of the ER step. Subsequently, Oracle or LLMs perform entity matching based on the entropy-sorted list of candidate pairs. After resolving a certain number of pairs, entropy scores are recalibrated based on the updated token pool, excluding resolved pairs. The number of pairs resolved before recalibration is a hyperparameter that can be adjusted in our implementation, with the default value set to every 10% of the total number of pairs in a block. This recalibration aims to reorder candidate

pairs using the current token pool from unresolved entities, ensuring more accurate and efficient matching. The setup and results are reported separately for (1) entropy without feedback and (2) entropy with feedback in Section 5.

4 Experiments

In this section, we detail the experimental setup used to evaluate the efficacy of the entropy-based prioritization approach. We describe the datasets, Large Language Models (LLMs), baselines, and experimental outcomes, including insights from the results. All experiments, including blocking, tokenization, entropy calculation, and subsequent evaluations, were conducted using Python version 3.9.13.

We used three ER datasets to evaluate our approach. Specifically, we used two e-commerce datasets—WDC Products and Amazon-Google—and an academic dataset, DBLP-ACM.

1. **Amazon-Google:** [17] This dataset primarily contains software product listings featuring various versions of the Windows operating system alongside popular image and video editing applications. Compiled from both Amazon and Google, the dataset provides a rich environment for entity resolution with its overlap of product types and differing naming standards.
2. **WDC Products:** [29] This dataset contains various products offered across categories such as electronics and software, clothing, and tools. We chose the WDC computer products (small) dataset to conduct the experiment.
3. **DBLP-ACM:** [32] This dataset contains bibliographic entries from two prominent academic sources, DBLP and ACM. It includes titles, authors, publication venues, and other metadata. Although generally considered less challenging than the other two datasets, DBLP-ACM presents unique challenges due to the diversity in conventions and formatting styles between the sources.

We used GPT-3.5 turbo (specifically, the gpt-3.5-turbo-0125 model) developed by OpenAI as the expensive entity matching method. It is a commercial LLM that is known to have excellent performance on a range of tasks.

Blocking Benchmarks and Baselines: We compared the results of candidate pairs ordered using an entropy-based approach with standard progressive entity resolution techniques. As explained in Section 3.2, we used two blocking methods commonly utilized in progressive entity resolution frameworks: PSN and HRP. Each blocking method is applied to each dataset before serving as input for the sorting step, where the baselines are compared against the entropy-based approach. We report the dataset statistics and the Pair Completeness (PC) of the initial blocking step for each dataset in Table 1. PC measures the proportion of true matches retained after blocking, which is the maximum recall performance we can get for each dataset (in the overall ER pipeline). The details on how we compare the entropy-based approach with the baseline are provided below.

Table 1. Dataset size and pair completeness (PC) for PSN and HRP blocking Methods across three datasets

Dataset	# Rows	PC (PSN)	PC (HRP)
Amazon-Google	1363, 3226	92.3%	93.6%
WDC Products	1100, 1100	89.3%	89.7%
DBLP-ACM	2216, 2294	87.9%	88.4%

PSN Baseline. The Progressive Sorted Neighborhood (PSN) method employs heuristic sorting keys and a fixed window slide value to generate candidate pairs with a high probability of matching. In the Amazon-Google and WDC Product datasets, we utilized *product_title* as the baseline sorting key with a window size of 3. For the DBLP-ACM dataset, *biography_title* was used as the sorting key. We compared the baseline performance against two entropy-based approaches: (1) PSN using entropy as the sorting key without any feedback update and (2) PSN with entropy as the sorting key with feedback update.

HRP Baseline. We used traditional HRP to generate a hierarchy of blocks where we set the level of hierarchy = 3 and resolve blocks from lowest level to highest. Each individual block at a different level of hierarchy is sorted using the same baseline sorting key we used in PSN, which are *product_title* for Amazon-Google and WDC Product datasets, and *biography_title* for DBLM-ACM dataset. The baseline sorted blocks are then compared with (1) HRP using entropy as the sorting key without any feedback update and (2) PSN with entropy as the sorting key with feedback update.

5 Results and Discussion

We evaluate the effectiveness of the entropy-based approach by measuring progressive recall relative to token costs. Specifically, we analyze two variations: entropy without feedback and entropy with feedback. This analysis demonstrates how entropy-based strategies can effectively prioritize candidate pairs with a high probability of matching within a constrained budget. Additionally, to ensure a comprehensive assessment of entity resolution (ER) performance at any budget level, we implement a cost-effective method using TF-IDF and cosine similarity with a defined threshold of 0.8. This cost-effectiveness similarity method is designed to resolve any remaining pairs at specific budget points, allowing us to gauge the overall ER performance even when the budget does not extend to all candidate pairs. For instance, consider a scenario with 1000 total candidate pairs and a limited budget that only allows for resolving 500 pairs. In this case, we prioritize the top 500 pairs based on entropy using language model lookups (LLMs) and address the remaining 500 pairs using a more economical similarity-based approach. This dual strategy enables us to maintain performance insights across the full dataset under budget constraints.

Performance using PSN as blocking method: Figure 4 and 5 show progressive recall performance when Oracle and GPT-3.5 are used for entity match-

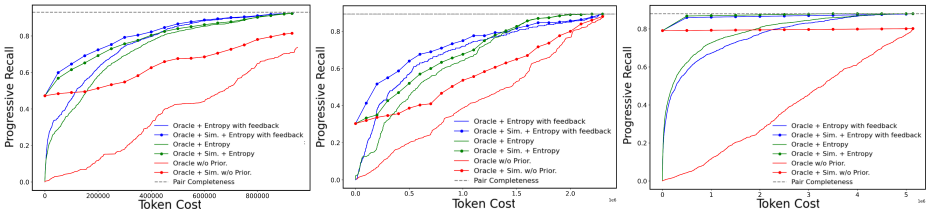

Fig. 4. Progressive recall performance of PSN blocking from three datasets: Amazon-Google (left), WDC Products (middle), and DBLP-ACM (right) when using Oracle. Baselines are presented using red lines, while entropy-based variations are presented in green (without feedback) and blue (with feedback).

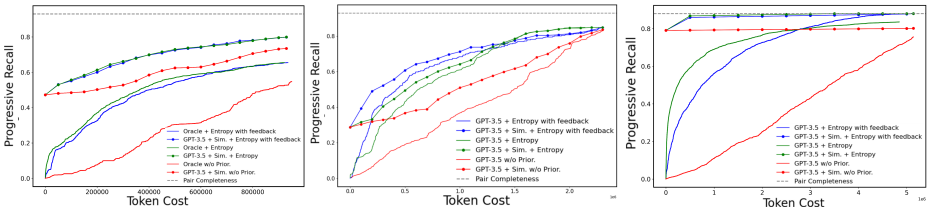

Fig. 5. Progressive recall performance of PSN blocking from three datasets: Amazon-Google (left), WDC Products (middle), and DBLP-ACM (right) when using GPT-3.5. Baselines are presented using red lines, while entropy-based variations are presented in green (without feedback) and blue (with feedback).

ing, respectively. We found that entropy-based approaches outperformed all default baselines. We further analyze the token cost cutoff where entropy-based approaches (with and without similarity method) performed relatively well. As shown in Figure 4 (left), the performance from Amazon-Google shows that both entropy-based approaches outperformed the default PSN. When employing entropy scores as the sorting key within the PSN framework, recall performance significantly improves, exceeding 0.5 from token cost between 0 to 200k (25% of total cost). Table 2 (Oracle columns) also shows that at 44.2% and 42.6% of the total cost, both variations of entropy-based approaches achieved a recall of 0.8. Moreover, the maximum performance of both variations of the entropy-based approaches is higher than the default approach, where both entropy-based methods beat the maximum baseline performance at 44.8% and 43.0% of the total cost. This is due to the different order of initial candidate pairs within blocks resulting from different sorting keys. PSN uses a fixed sliding window to generate the final candidate pairs. Therefore, different sorting keys make the list of candidate pairs different when using entropy scores as a sorting key compared to the default PSN.

For the WDC Products dataset, as depicted in Figure 4 (middle), the recall plot illustrates that entropy-based approaches consistently outperform the default Progressive Sorted Neighborhood (PSN). Specifically, the recall for entropy-

Metric	Amazon-Google		WDC Products		DBLP-ACM	
	Oracle	GPT-3.5	Oracle	GPT-3.5	Oracle	GPT-3.5
% of total cost achieving 0.8 recall (E)	44.2%	-%	50.6%	59.2%	45.2%	51.9%
% of total cost achieving 0.8 recall (E+F)	42.6%	-%	53.7%	68.5%	35.6%	55.7%
% of total cost achieving maximum baseline perf. (E)	44.8%	44.6%	100%	100%	45.2%	35.7%
% of total cost achieving maximum baseline perf. (E+F)	43.0%	54.3%	100%	100%	35.6%	42.3%

Table 2. The percentage out of total cost required to achieve competitive entity resolution (ER) performance using the entropy-based approach with the PSN blocking method. We report two scenarios from the ER results: (1) percentage out of the total cost to achieve 80% recall and (2) percentage out of the total cost to achieve the maximum recall of the baseline approach. The dash (-%) indicates that the ER performance did not reach 0.8 recall. The notation "E" refers to the normal entropy approach, while "E+F" refers to the entropy with feedback approach.

based methods increased approximately 25% faster than the default PSN over the initial 1M token cost range. Towards the end of this range, the recall rates of both entropy-based approaches begin to converge, ultimately reaching the same maximum recall. As shown in Table 2 (Oracle columns), the entropy-based approaches used 50.6% and 53.7% of the total cost to achieve 0.8 recall. However, they did not exceed the maximum performance of the baseline, as the recall rates converged to the same value in the end.

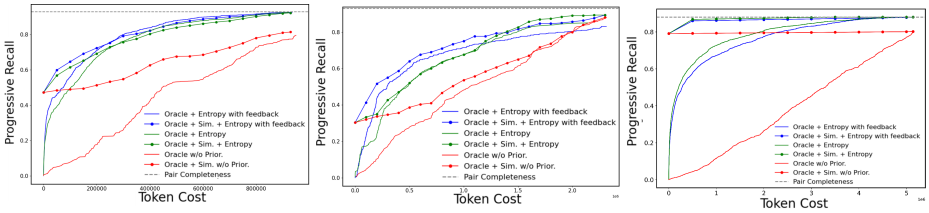

Fig. 6. Progressive recall performance of HRP blocking from three datasets: Amazon-Google (left), WDC Products (middle), and DBLP-ACM (right) when using Oracle. Baselines are presented using red lines, while entropy-based variations are presented in green (without feedback) and blue (with feedback).

For the DBLP-ACM dataset, which is the simplest among the three, even basic similarity methods achieve high performance without the need for prioritization. However, when employing Oracle as an entity matcher, entropy-based approaches still outperform the default Progressive Sorted Neighborhood (PSN), as shown in Figure 4 (right). This advantage is particularly noticeable in the early stages of entity matching, where entropy-based methods achieve a recall exceeding 0.8 at approximately 45.2% and 35.6% of the total token cost. Since the baseline method did not reach 0.8 recall, both entropy-based methods beat the baseline approach at the same budget points.

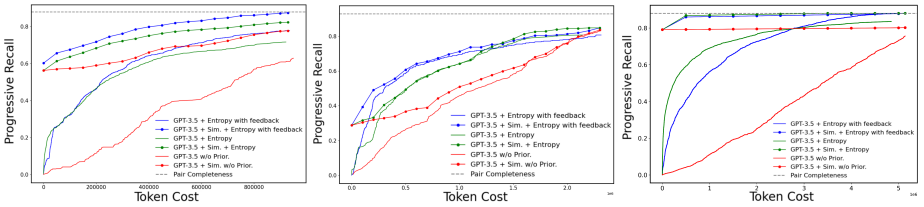

Fig. 7. Progressive recall performance of HRP blocking from three datasets: Amazon-Google (left), WDC Products (middle), and DBLP-ACM (right) when using GPT-3.5. Baselines are presented using red lines, while entropy-based variations are presented in green (without feedback) and blue (with feedback).

Metric	Amazon-Google		WDC Products		DBLP-ACM	
	Oracle	GPT-3.5	Oracle	GPT-3.5	Oracle	GPT-3.5
% of total cost achieving 0.8 recall (E)	32.6%	-%	55.2%	68.5%	45.7%	52.0%
% of total cost achieving 0.8 recall (E+F)	33.6%	-%	64.8%	92.6%	35.8%	55.8%
% of total cost achieving maximum baseline perf. (E)	33.1%	45.6%	100%	100%	46.7%	35.9%
% of total cost achieving maximum baseline perf. (E+F)	34.2%	41.3%	100%	100%	35.9%	42.4%

Table 3. The percentage out of total cost required to achieve competitive entity resolution (ER) performance using the entropy-based approach with the HRP blocking method. We report two scenarios from the ER results: (1) percentage out of the total cost to achieve 80% recall and (2) percentage out of the total cost to achieve the maximum recall of the baseline approach. The dash (-%) indicates that the ER performance did not reach 0.8 recall. The notation "E" refers to the normal entropy approach, while "E+F" refers to the entropy with feedback approach.

When using GPT-3.5 as an entity matcher, the performance is shown in Figure 5 . The overall trend of recall performance mirrors that of using Oracle, with both entropy-based variations outperforming the baseline at the same rate. However, the average recall performance decreased by approximately 10-20% due to GPT-3.5 incorrectly resolving some true positive (TP) pairs as negatives. Consequently, the budget percentage required to achieve 80% recall increased by 10-20%, as shown in Table 2 (GPT-3.5 columns). This outcome aligns more closely with typical entity-matching scenarios, where errors are expected.

Performance using HRP as blocking method: Figures 6 and 7 illustrate the recall performance with Oracle and GPT-3.5 as entity matchers, respectively. The overall performance is similar to that obtained with PSN as the blocking method across all datasets and entity matchers. Both entropy-based methods consistently outperformed the baseline HRP method, accelerating early recall performance across all datasets. Notably, the early recall acceleration rate is slightly higher with HRP than with PSN for the Amazon-Google and DBLP-ACM datasets.

For the Amazon-Google dataset, HRP with entropy-based approaches achieved 80% recall using only 32.6% and 33.6% of the total budget, as shown in Table 3 (Oracle). This indicates that HRP with entropy-based methods accelerates early

Table 4. Qualitative Examples from the ground truth of three ER datasets including the Amazon-Google, WDC Products, and DBLP-ACM Datasets.

Dataset	Source	Title
Amazon-Google	A	adobe premiere pro cs3 upgrade
	G	premiere pro cs3 mac upgrade
	A	mia's math adventure: just in time
	G	kutoka interactive 61208 mia's math adventure: just in time!
	A	noah's ark activity center (jewel case ages 3-8)
	G	the beginners bible: noah's ark activity center: activity center
WDC Products	P1	Benq ZOWIE RL2455 24 Full HD TN Grey computer monitor 4718755065736) Product data Benq monitor monitors
	P2	Zowie RL2455 E-Sports 24 Full HD LED Monitor 24 inch 1ms Monitor - HDMI 9H.LF4LB.DBE
	P1	Matrox G550 Low Profile Graphics Card
	P2	MATROX G55MDDE32LPDF 32MB Millenium G550 LP PCI-E Video Card with Cable
	P1	WD Black 2TB Performance Desktop Hard Disk Drive - 7200 RPM SATA 6 Gb/s 64MB Cache 3.5 Inch
	P2	HDD Black 2TB 3.5 SATA 6Gbs 64MB@es Cables
DBLP-ACM	DBLP	Database Systems: The Complete Book
	ACM	Database Systems: The Complete Book (2nd Edition)
	DBLP	Introduction to Algorithms
	ACM	Introduction to Algorithms, 3rd Edition
	DBLP	Artificial Intelligence: A Modern Approach
	ACM	Artificial Intelligence: A Modern Approach (3rd Edition)

recall performance by about 10-11% faster than PSN for this dataset. A similar improvement is seen in Table 3 (GPT-3.5) when using GPT-3.5 as the entity matcher where GPT-3.5 used about 7-12% more budget to achieve the same result as Oracle.

However, for the WDC Products dataset, HRP with entropy-based approaches achieved 80% recall using 55.2% and 64.8% of the total budget, which is 5-11% slower than PSN. Even with a slower pace compared to PSN, the entropy-based approaches still consistently outperformed the baselines by accelerating approximately 25% faster than the default HRP. This trend is also observed when using GPT-3.5 as an entity matcher.

Due to the simplicity of the DBLP-ACM dataset, the results from both Oracle and GPT-3.5 with HRP are almost identical to those with PSN across all sorting methods (baseline and entropy-based approaches). Both entropy-based approaches consistently outperformed the baseline at similar rates and budget points. The maximum recall performance is also influenced by the difficulty of the dataset. As shown in Table 4, the DBLP-ACM dataset is the easiest, as

illustrated by the candidate pair "Database Systems: The Complete Book" from DBLP and "Database Systems: The Complete Book (2nd Edition)" from ACM, where the differences are minimal and they share multiple identical tokens. This makes entity matching straightforward for GPT-3.5. In contrast, the Amazon-Google dataset is more complex, exemplified by the pair "mia's math adventure: just in time" versus "kutoka interactive 61208 mia's math adventure: just in time!" which includes differences in formatting and additional model information, making entity resolution challenging. The WDC Products dataset also shows high difficulty, exemplified by pairs "WD Black 2TB Performance Desktop Hard Disk Drive" versus "HDD Black 2TB 3.5 SATA," where the brand name and description are omitted in one of the entities. These variations led to mismatches and performance drop when using GPT-3.5 as an entity matcher.

6 Conclusion and Future Work

Entropy-based block prioritization methods are able to effectively prioritize high-probability candidate pairs, achieving superior performance at a reduced cost. On average, our approach accelerates the expected ER performance by 30% compared to traditional methods on both Oracle and GPT-3.5. This demonstrates that our approach effectively prioritizes pairs that LLMs can resolve efficiently. The cost of resolving candidate pairs is correlated with the size of the datasets. For example, two datasets with 2,000 records each would typically require evaluating over 4 million candidate pairs without blocking. With blocking, this number might still exceed 40,000 pairs, assuming a 99% reduction ratio. The proposed method mitigates this cost by easily integrating with existing blocking techniques and significantly enhances early matching performance through budgeted use of LLMs. Future work should expand the scope of datasets to include ER benchmarks from different fields, such as the Cora, FEBRL, and MusicBrainz20K datasets, to further validate the robustness of entropy-based approaches. Additionally, improving feedback mechanisms through different methods for dynamically updating entropy scores could enhance performance. Finally, as large language models evolve, it is crucial to compare our entropy-based approaches with more LLMs, such as Gemini, LLaMA, GPT-4, and beyond, to ensure competitive results across multiple LLMs.

References

1. Altowim, Y., Kalashnikov, D.V., Mehrotra, S.: Progresser: adaptive progressive approach to relational entity resolution. ACM Transactions on Knowledge Discovery from Data (TKDD) **12**(3), 1–45 (2018)
2. Altowim, Y., Mehrotra, S.: Parallel progressive approach to entity resolution using mapreduce. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE). pp. 909–920. IEEE (2017)
3. Betancourt, B., Zanella, G., Miller, J.W., Wallach, H., Zaidi, A., Steorts, R.C.: Flexible models for microclustering with application to entity resolution. Advances in neural information processing systems **29** (2016)

4. Christophides, V., Efthymiou, V., Palpanas, T., Papadakis, G., Stefanidis, K.: An overview of end-to-end entity resolution for big data. ACM Computing Surveys (CSUR) **53**(6), 1–42 (2020)
5. Christophides, V., Efthymiou, V., Stefanidis, K.: Entity resolution in the web of data, vol. 5. Springer (2015)
6. Ebraheem, M., Thirumuruganathan, S., Joty, S., Ouzzani, M., Tang, N.: Deeper–deep entity resolution. arXiv preprint arXiv:1710.00597 (2017)
7. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on knowledge and data engineering **19**(1), 1–16 (2006)
8. Fan, L., Li, L., Ma, Z., Lee, S., Yu, H., Hemphill, L.: A bibliometric review of large language models research from 2017 to 2023 (2023)
9. Fellegi, I.P., Sunter, A.B.: A theory for record linkage. Journal of the American Statistical Association **64**(328), 1183–1210 (1969)
10. Galhotra, S., Firmani, D., Saha, B., Srivastava, D.: Efficient and effective er with progressive blocking. The VLDB Journal **30**(4), 537–557 (2021)
11. Getoor, L., Machanavajjhala, A.: Entity resolution for big data. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1527–1527 (2013)
12. Giray, L.: Prompt engineering with chatgpt: a guide for academic writers. Annals of biomedical engineering **51**(12), 2629–2633 (2023)
13. Kasneci, E., Seßler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., et al.: Chatgpt for good? on opportunities and challenges of large language models for education. Learning and individual differences **103**, 102274 (2023)
14. Kejriwal, M.: Populating entity name systems for big data integration. In: The Semantic Web–ISWC 2014: 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II 13. pp. 521–528. Springer (2014)
15. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. In: Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., Oh, A. (eds.) Advances in Neural Information Processing Systems. vol. 35, pp. 22199–22213. Curran Associates, Inc. (2022), https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf
16. Kooli, N., Allesiardo, R., Pigneul, E.: Deep learning based approach for entity resolution in databases. In: Asian conference on intelligent information and database systems. pp. 3–12. Springer (2018)
17. Köpcke, H., Thor, A., Rahm, E.: Evaluation of entity resolution approaches on real-world match problems (09 2010). https://doi.org/10.14778/1920841.1920904
18. Li, H., Feng, L., Li, S., Hao, F., Zhang, C.J., Song, Y., Chen, L.: On leveraging large language models for enhancing entity resolution. arXiv preprint arXiv:2401.03426 (2024)
19. Li, H., Li, S., Hao, F., Zhang, C.J., Song, Y., Chen, L.: Booster: Leveraging large language models for enhancing entity resolution. arXiv preprint arXiv:2403.06434 (2024)
20. Lyu, C., Xu, J., Wang, L.: New trends in machine translation using large language models: Case examples with chatgpt. arXiv preprint arXiv:2305.01181 (2023)
21. Nananukul, N., Sisaengsuwanchai, K., Kejriwal, M.: Cost-efficient prompt engineering for unsupervised entity resolution (2024)
22. OpenAI: Gpt-4 technical report (2024)

23. Papadakis, G., Alexiou, G., Papastefanatos, G., Koutrika, G.: Schema-agnostic vs schema-based configurations for blocking methods on homogeneous data. Proceedings of the VLDB Endowment **9**(4), 312–323 (2015)
24. Papadakis, G., Ioannou, E., Niederée, C., Palpanas, T., Nejdl, W.: Eliminating the redundancy in blocking-based entity resolution methods. In: Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries. pp. 85–94 (2011)
25. Papadakis, G., Ioannou, E., Palpanas, T., Niederée, C., Nejdl, W.: A blocking framework for entity resolution in highly heterogeneous information spaces. IEEE Transactions on Knowledge and Data Engineering **25**(12), 2665–2682 (2012)
26. Papadakis, G., Koutrika, G., Palpanas, T., Nejdl, W.: Meta-blocking: Taking entity resolutionto the next level. IEEE Transactions on Knowledge and Data Engineering **26**(8), 1946–1960 (2013)
27. Papadakis, G., Skoutas, D., Thanos, E., Palpanas, T.: Blocking and filtering techniques for entity resolution: A survey. ACM Comput. Surv. **53**(2) (mar 2020). https://doi.org/10.1145/3377455, `https://doi.org/10.1145/3377455`
28. Peeters, R., Bizer, C.: Using chatgpt for entity matching. In: European Conference on Advances in Databases and Information Systems. pp. 221–230. Springer (2023)
29. Primpeli, A., Peeters, R., Bizer, C.: The wdc training dataset and gold standard for large-scale product matching (2019)
30. Simonini, G., Papadakis, G., Palpanas, T., Bergamaschi, S.: Schema-agnostic progressive entity resolution. IEEE Transactions on Knowledge and Data Engineering **31**(6), 1208–1221 (Jun 2019). https://doi.org/10.1109/tkde.2018.2852763, `http://dx.doi.org/10.1109/TKDE.2018.2852763`
31. Stefanidis, K., Christophides, V., Efthymiou, V.: Web-scale blocking, iterative and progressive entity resolution. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE). pp. 1459–1462 (2017). https://doi.org/10.1109/ICDE.2017.214
32. Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., Su, Z.: Arnetminer: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 990–998 (2008)
33. Tang, L., Sun, Z., Idnay, B., Nestor, J.G., Soroush, A., Elias, P.A., Xu, Z., Ding, Y., Durrett, G., Rousseau, J.F., et al.: Evaluating large language models on medical evidence summarization. npj Digital Medicine **6**(1), 158 (2023)
34. Team, G.: Gemini: A family of highly capable multimodal models (2023)
35. Thirunavukarasu, A.J., Ting, D.S.J., Elangovan, K., Gutierrez, L., Tan, T.F., Ting, D.S.W.: Large language models in medicine. Nature medicine **29**(8), 1930–1940 (2023)
36. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
37. Whang, S.E., Marmaros, D., Garcia-Molina, H.: Pay-as-you-go entity resolution. IEEE Transactions on Knowledge and Data Engineering **25**(5), 1111–1124 (2012)
38. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., Schmidt, D.C.: A prompt pattern catalog to enhance prompt engineering with chatgpt. arXiv preprint arXiv:2302.11382 (2023)
39. Yan, S., Lee, D., Kan, M.Y., Giles, L.C.: Adaptive sorted neighborhood methods for efficient record linkage. In: Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries. pp. 185–194 (2007)

40. Yasunaga, M., Ren, H., Bosselut, A., Liang, P., Leskovec, J.: Qa-gnn: Reasoning with language models and knowledge graphs for question answering. arXiv preprint arXiv:2104.06378 (2021)
41. Zhang, T., Ladhak, F., Durmus, E., Liang, P., McKeown, K., Hashimoto, T.B.: Benchmarking large language models for news summarization. Transactions of the Association for Computational Linguistics **12**, 39–57 (2024)
42. Zhang, W., Wei, H., Sisman, B., Dong, X.L., Faloutsos, C., Page, D.: Autoblock: A hands-off blocking framework for entity matching. In: Proceedings of the 13th International Conference on Web Search and Data Mining. pp. 744–752 (2020)
43. Zhou, Y., Muresanu, A.I., Han, Z., Paster, K., Pitis, S., Chan, H., Ba, J.: Large language models are human-level prompt engineers. arXiv preprint arXiv:2211.01910 (2022)