# Fine-tuning Large Language Models for Entity Matching

Aaron Steiner
University of Mannheim
Mannheim, Germany
aasteine@mail.uni-mannheim.de

Ralph Peeters
University of Mannheim
Mannheim, Germany
ralph.peeters@uni-mannheim.de

Christian Bizer
University of Mannheim
Mannheim, Germany
christian.bizer@uni-mannheim.de

#### **ABSTRACT**

Generative large language models (LLMs) are a promising alternative to pre-trained language models for entity matching due to their high zero-shot performance and their ability to generalize to unseen entities. Existing research on using LLMs for entity matching has focused on prompt engineering and in-context learning. This paper explores the potential of fine-tuning LLMs for entity matching. We analyze fine-tuning along two dimensions: 1) The representation of training examples, where we experiment with adding different types of LLM-generated explanations to the training set, and 2) the selection and generation of training examples using LLMs. In addition to the matching performance on the source dataset, we investigate how fine-tuning affects the model's ability to generalize to other in-domain datasets as well as across topical domains. Our experiments show that fine-tuning significantly improves the performance of the smaller models while the results for the larger models are mixed. Fine-tuning also improves the generalization to in-domain datasets while hurting cross-domain transfer. We show that adding structured explanations to the training set has a positive impact on the performance of three out of four LLMs, while the proposed example selection and generation methods only improve the performance of Llama 3.1 8B while decreasing the performance of GPT-40 Mini.

#### **Artifact Availability:**

The source code, data, and/or other artifacts have been made available at https://github.com/wbsg-uni-mannheim/TailorMatch.

#### 1 INTRODUCTION

Entity matching [1, 2, 5] is the task of identifying entity descriptions in different data sources that refer to the same real-world entity. Entity matching is a central step in data integration pipelines [3]. Many state-of-the-art entity matching methods build on pre-trained language models (PLMs) [20], such as BERT or RoBERTa [9, 16, 19, 24]. Recent work on using generative large language models (LLMs) [26], such as GPT, Llama, Gemini, or Mixtral, for entity matching has shown that LLMs have higher zero-shot performance compared to PLMs and generalize better to unseen entities [14, 17, 18]. Most research on using LLMs for entity matching focuses on prompt engineering and in-context learning [6, 14, 17]. Initial papers have

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097. doi:XX.XX/XXX.XX

started to investigate the potential of improving beyond in-context learning by fine-tuning LLMs for entity matching [18, 25]. This paper builds on this work and presents a deeper investigation of the utility of fine-tuning LLMs for entity matching. Besides the effect of fine-tuning on the matching performance, we investigate how fine-tuning influences the model's ability to generalize to different in-domain datasets as well as across domains. Our investigation focuses on two dimensions: First, the representation of training examples, where we investigate the effect of augmenting each training example with textual as well as structured explanations about why a pair of entity descriptions matches or not. The second dimension is the selection and generation of examples for the training set that is used for fine-tuning. We experiment with using LLMs to filter misleading examples from the training set as well as to generate additional training examples. We further investigate the error-based selection of training examples. Figure 1 gives an overview of the fine-tuning and inference setup. The effects of fine-tuning are compared between open-source (Llama-3.1) and proprietary (GPT-40) models. While we test models of varying sizes, the focus of the investigation lies on smaller models, as these models are less resource-intensive to deploy.

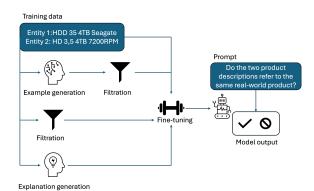


Figure 1: Fine-tuning and inference setup including explanation generation, example generation, and filtration.

Contributions: This paper makes the following contributions:

- (1) Standard Fine-Tuning: To establish baselines, we fine-tune open source and proprietary LLMs of different sizes using standard fine-tuning techniques. The baselines show that fine-tuning significantly improves the performance of the smaller models while the results for the larger models are mixed.
- (2) **In-domain and Cross-domain Generalization:** We assess the effectiveness of fine-tuned models in both in-domain

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit https://creativecommons.org/licenses/by-nc-nd/4.0/ to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

- and cross-domain settings. The experiments show that finetuned models often generalize to in-domain datasets, while their performance in cross-domain settings falls short of the zero-shot baseline in the majority of cases.
- (3) Example Representation: We investigate the impact of augmenting the training set with different types of LLMgenerated explanations. Adding structured explanations improves the performance of three out of four fine-tuned models, while also improving in-domain generalization.
- (4) **Example Selection and Generation:** We introduce different approaches for using LLMs for filtering the training set and for the generation of new training examples. Both techniques improve the performance after fine-tuning, with example generation particularly enhancing in-domain generalization.
- (5) Prompt Sensitivity: We examine how fine-tuning affects the model's sensitivity to variations of the prompt. Our experiments show that fine-tuning generally reduces prompt sensitivity, with the introduction of structured explanations further stabilizing model performance across different prompts.

#### 2 EXPERIMENTAL SETUP

This section provides details on the models, benchmark datasets, fine-tuning setup, and the performance evaluation that we use for the experiments. All artefacts are provided in the project repository, meaning that all experiments can be replicated.

**Large Language Models:** We compare the following hosted proprietary models and locally-run open-source models:

- (1) **gpt-4o-2024-08-06:** Released in May 2024, this model is OpenAI's flagship LLM and currently used by ChatGPT.
- (2) gpt-4o-mini-2024-07-18: This hosted LLM, released in July 2024, has lower usage fees compared to GPT-4o. Although the exact model size is not disclosed, the naming convention indicates that it is smaller than GPT-4o.
- (3) Meta-Llama-3.1-8B-Instruct<sup>1</sup>: Released in July 2024, this open-source model from Meta features the same 128k token context length as the GPT models.
- (4) **Meta-Llama-3.1-70B-Instruct**<sup>2</sup>: Also developed by Meta, this model is medium-sized within the Llama series. Further details are available in the accompanying paper by Meta [4].

We use the OpenAI batch API<sup>3</sup> and Hugging Face's Transformers<sup>4</sup> library to prompt both hosted and local models. In both cases, the generation temperature is set at zero to minimize randomness. The open-source models run on local machines with varying hardware configurations, including an AMD EPYC 7413 processor, 1024GB RAM, and up to eight Nvidia RTX6000 GPUs.

**Datasets:** We evaluate in-domain and cross-domain generalization using benchmark datasets from two topical domains [8, 19]: products and scholarly works. The datasets are selected based on three criteria: First, we chose datasets with at least 150 matches

in the test set (see Table 1) to ensure the stability of the performance measurement. Second, we prioritize datasets that contain a significant number of challenging corner case record pairs. These corner cases are either matching or non-matching pairs that closely resemble the opposite class due to very similar or dissimilar surface forms [19]. Finally, to evaluate cross-domain generalization, we chose datasets from two topical domains. We use the following datasets:

- WDC Products: This dataset includes product offers from various categories (e.g., electronics, clothing) originating from numerous online shops. We use the most challenging version, with 80% corner cases (hard positives and negatives).
- **Abt-Buy**: This benchmark provides examples within similar categories to those found in WDC Products.
- Walmart-Amazon: The Walmart-Amazon benchmark includes products from both Walmart and Amazon, with categories comparable to those in WDC Products and Abt-Buy.
- Amazon-Google: Unlike the other benchmarks, the Amazon-Google dataset focuses on software products, such as different versions of the Windows operating system and various image/video editing applications. This dataset introduces a different product type for evaluating model performance.
- DBLP-Scholar: This benchmark requires matching bibliographic entries between DBLP and Google Scholar, representing a domain distinct from the product datasets.
- DBLP-ACM: Similar to DBLP-Scholar, the DBLP-ACM benchmark involves matching bibliographic entries.

We use the title attribute of the product datasets for fine-tuning, while we concatenate the author, title, venue, and year attributes of the bibliographic datasets using a semicolon character as a delimiter.

Table 1: Dataset statistics for the training, validation, and test sets.

| Dataset               | Training Set |        | Valida | tion Set | Test Set |       |
|-----------------------|--------------|--------|--------|----------|----------|-------|
|                       | # Pos        | # Neg  | # Pos  | # Neg    | # Pos    | # Neg |
| WDC Products (small)  | 500          | 2,000  | 500    | 2,000    | 500      | 4,000 |
| WDC Products (medium) | 1,500        | 4,500  | 500    | 3,000    | 500      | 4,000 |
| WDC Products (large)  | 8,471        | 11,364 | 500    | 4,000    | 500      | 4,000 |
| Abt-Buy (A-B)         | 822          | 6,837  | 206    | 1,710    | 206      | 1,710 |
| Amazon-Google (A-G)   | 933          | 8,234  | 234    | 2,059    | 234      | 2,059 |
| Walmart-Amazon (W-A)  | 769          | 7,424  | 193    | 1,856    | 193      | 1,856 |
| DBLP-Scholar (D-S)    | 4,277        | 18,688 | 1,070  | 4,672    | 1,070    | 4,672 |
| DBLP-ACM (D-A)        | 1,776        | 8,114  | 444    | 2,029    | 444      | 2,029 |

**Fine-tuning Setup:** In order to keep computational costs manageable, we use the default hyperparameter setting recommended by the model providers and do not perform a hyperparameter search for each experiment. The fine-tuning setup differs for hosted and open-source models. We use the following default hyperparameter settings for the hosted OpenAI models: learning rate multiplier 1.8, batch size 16, and a constant random seed for reproducibility. We use the following fine-tuning setup for the open-source models: We employed Low-Rank Adaptation (LoRA) with an alpha of 16 to allow moderate adjustments to model weights, a dropout rate of 0.1 to prevent overfitting, and a rank (r) of 64 to balance performance and computational efficiency. The learning rate was configured at

<sup>&</sup>lt;sup>1</sup>https://huggingface.co/meta-llama/Meta-Llama-3.1-8B-Instruct

<sup>&</sup>lt;sup>2</sup>meta-llama/Meta-Llama-3.1-70B-Instruct

<sup>&</sup>lt;sup>3</sup>https://platform.openai.com/docs/guides/batch

<sup>&</sup>lt;sup>4</sup>https://huggingface.co/docs/transformers/en/index

2e-4. To ensure reproducibility, a consistent random seed was used across all libraries, and CUDA was set to deterministic mode. Both open-source and hosted models are trained for 10 epochs, with checkpoints generated after each epoch. Open-source models are validated at each checkpoint using custom evaluation callbacks. In contrast, OpenAI models provided only two intermediate checkpoints plus the final version, limiting the validation process.

**Evaluation:** We use the standard evaluation metrics precision, recall, and F1-score, following related work [1]. All subsequent tables show the F1 performance of a model in the given setting. The precision and recall results of all experiments are provided in the repository accompanying this paper. Model responses in natural language are evaluated using Narayan et al.'s [14] method of parsing responses to contain "yes" or "no".

#### 3 STANDARD FINE-TUNING

In order to establish baselines, this section reports the results of a series of fine-tuning experiments using a simple, straight-forward representation of the training examples. Figure 2 shows two training examples from this fine-tuning set. Each example is represented by a prompt inquiring whether the entity descriptions refer to the same entity, followed by the two descriptions and the completion "Yes" or "No", depending on whether the descriptions are a match or a non-match.

| Prompt   | Completion |
|--|------------|
| "Do the two entity descriptions refer to the same real-world product?" Entity 1: Jabra EVOLVE 80 MS Stereo (7899-823-109) Entity 2: Jabra Evolve 80 UC stereo Skype for Business | "Yes."     |
| "Do the two entity descriptions refer to the same real-world product?" Entity 1: CLARKS Sram, PG-730, 7sp cassette, 12-32T Entity 2: Sram PG 1130 11sp cassette 11-36T           | "No."      |

Figure 2: Representation of two training examples in the standard fine-tuning setup.

Table 2 reports the results of the standard fine-tuning experiments. Each row reports the results for a specific LLM (column Model) which was fine-tuned using a specific training set (column Training set). The following columns contain the results of evaluating this fine-tuned model on different test sets. The "zero-shot" rows report results for the specific models without any fine-tuning.

# 3.1 Effectiveness

This section discusses the results of testing the LLMs with examples from the same dataset (a non-transfer setting). The fine-tuned models show a performance increase on most datasets compared to the baseline models.

**Smaller Models:** Llama-8B shows an average F1 score gain of 17.31 points over the zero-shot baseline (Table 2, upper section). Five out of six fine-tuned Llama 8B models significantly improve on their zero-shot baselines, with Abt-Buy achieving the highest gain of 30.77 points (Table 2, A-B column). The Amazon-Google model is an outlier, only exceeding the baseline by 0.84 points, underperforming compared to other product domain models (Table 2, A-G column). GPT-40-mini shows an average F1 score increase of 11.72 points, a significant gain given the models high zero-shot performance (Table 2, , rows gpt-40-m). Unlike Llama-8B, GPT-40-mini

demonstrates significant performance gains on the Amazon-Google dataset, achieving the highest absolute performance improvement among all 40-mini models as presented in the A-G column of Table 2

Large Models: Given resource constraints, we fine-tune the larger models only on the WDC small training set. For the Llama 70B model fine-tuning results in a decrease in performance, with a reduction of 2.53 points in F1 score on the WDC test set, as shown in the WDC column of Table 2. In contrast, GPT-40, which starts with a similar zero-shot score as GPT-40-mini, continues the trend of fine-tuning improving performance. It achieves a 5.43-point increase to a F1 score of 87.07 on the WDC dataset, the highest result so far.

#### 3.2 Generalization

We evaluate two types of generalization: in-domain, where performance is assessed across datasets within the same topical domain, and cross-domain, where a model is trained using a training set from one domain and tested with a test set from a different topical domain (e.g., from products to scholar).

**Transfer Gain:** We quantify the capability of a fine-tuned model to generalize to target datasets by relating the performance gains of the fine-tuned model on the target datasets to the performance gain that is achieved by fine-tuning specialized models with training examples from each target dataset. The resulting transfer gain is calculated by dividing the average performance gain on target datasets by the average gain achieved by models which were fine-tuned specifically on those datasets.

$$\label{eq:Transfer Gain = } \frac{\text{Avg performance gain on target datasets}}{\text{Avg performance gain of specialized models}}$$

For example, the generalization gain of the WDC model is 72% (Table 2, row Llama 8B/WDC, column Transfer Gain) and is calculated by dividing the average gain of the WDC model of 10.52 points on the product datasets by the average gain of specialized models for these datasets, which is 18.41 points F1.

**In-domain Generalization:** In the product domain, five out of six Llama 8B models improve performance on both their own and other datasets. Even the underperforming Amazon-Google model shows a gain on the Abt-Buy dataset. However, it is the only model to experience a decline in performance in an in-domain generalization setting, as indicated by the WDC column in Table 2. On average, Llama 8B models that were trained on a source dataset achieve 59% of the performance in an in-domain transfer scenario compared to models that were fine-tuned directly for the target datasets. GPT-40-mini also exhibits some ability to transfer between different product datasets, but to a lesser extent, achieving only a 15% performance increase relative to dataset-specific models. This lower average is primarily influenced by the poor performance of the Amazon-Google and Walmart-Amazon models on the WDC dataset, as shown in Table 2. Considering that the larger models are only fine-tuned using the WDC dataset, in-domain generalization can only be assessed within the product domain. The fine-tuned Llama 70B model demonstrates the ability to generalize to the Amazon-Google and Walmart-Amazon datasets, as evidenced by its performance exceeding the baseline by 3.92 and 4.94 F1 points, respectively. However, it underperforms on the Abt-Buy dataset,

| Table 2: F1 scores after standard fine-tuning. The rows determine the model/training set combination, while the columns                |
|--|
| contain results for different test sets. The best result of each model per test set is set bold, the second best result is underlined. |

| Model     | Training set |                | Product        | Domain         |                | Transfer | Scholar        | Scholar Domain |       |
|-----------|--------------|----------------|----------------|----------------|----------------|----------|----------------|----------------|-------|
| 1110401   | Training sec | A-B            | A-G            | W-A            | WDC            | Gain     | D-A            | D-S            | Gain  |
| Llama 8B  | Zero-shot    | 56.57 (0.00)   | 49.16 (0.00)   | 42.04 (0.00)   | 53.36 (0.00)   | -        | 85.52 (0.00)   | 67.69 (0.00)   | -     |
| Llama 8B  | A-B          | 87.34 (+30.77) | 59.16 (+10.00) | 60.39 (+18.35) | 66.07 (+12.71) | 102%     | 79.60 (-5.92)  | 42.89 (-24.80) | -83%  |
| Llama 8B  | A-G          | 67.48 (+10.91) | 50.00 (+0.84)  | 44.73 (+2.69)  | 39.53 (-13.83) | -1%      | 76.28 (-9.24)  | 60.89 (-6.80)  | -43%  |
| Llama 8B  | W-A          | 86.24 (+29.67) | 60.41 (+11.25) | 65.65 (+23.61) | 57.80 (+4.44)  | 96%      | 71.71 (-13.81) | 51.19 (-16.50) | -82%  |
| Llama 8B  | WDC          | 81.78 (+25.21) | 52.29 (+3.13)  | 53.74 (+11.70) | 69.19 (+15.83) | 72%      | 74.52 (-11.00) | 67.40 (-0.30)  | -30%  |
| Llama 8B  | D-A          | 58.02 (+1.45)  | 49.66 (+0.50)  | 40.82 (-1.22)  | 39.63 (-14.72) | -20%     | 97.42 (+11.90) | 79.56 (+11.87) | 47%   |
| Llama 8B  | D-S          | 65.71 (+9.15)  | 46.22 (-2.95)  | 42.35 (+0.31)  | 52.00 (-1.35)  | 7%       | 96.70 (+11.18) | 92.95 (+25.26) | 94%   |
| gpt-4o-m  | Zero-shot    | 87.68 (0.00)   | 59.20 (0.00)   | 65.06 (0.00)   | 81.61 (0.00)   | -        | 94.16 (0.00)   | 87.96 (0.00)   | -     |
| gpt-4o-m  | A-B          | 94.09 (+6.41)  | 67.18 (+7.98)  | 68.81 (+3.75)  | 82.69 (+1.08)  | 35%      | 96.94 (+2.79)  | 88.85 (+0.89)  | 28%   |
| gpt-4o-m  | A-G          | 83.51 (-4.17)  | 80.25 (+21.05) | 68.97 (+3.91)  | 73.99 (-7.62)  | -36%     | 96.28 (+2.12)  | 85.60 (-2.37)  | -2%   |
| gpt-4o-m  | W-A          | 92.08 (+4.40)  | 67.50 (+8.30)  | 78.85 (+13.79) | 78.52 (-3.09)  | 33%      | 95.58 (+1.43)  | 86.97 (-0.99)  | 3%    |
| gpt-4o-m  | WDC          | 91.44 (+3.76)  | 64.11 (+4.91)  | 68.92 (+3.86)  | 84.38 (+2.77)  | 9%       | 85.35 (-8.81)  | 76.33 (-11.63) | -155% |
| gpt-4o-m  | D-A          | 88.94 (1.26)   | 67.32 (+8.12)  | 67.51 (2.45)   | 81.34 (0.27)   | 27%      | 99.10 (+4.94)  | 89.93 (+1.97)  | 24%   |
| gpt-4o-m  | D-S          | 89.76 (+2.08)  | 65.71 (+6.50)  | 68.46 (+3.39)  | 70.87 (-10.73) | 3%       | 95.36 (+1.20)  | 96.22 (+8.26)  | 24%   |
| Llama 70B | Zero-shot    | 79.12 (0.00)   | 51.44 (0.00)   | 55.62 (0.00)   | 75.19 (0.00)   | -        | 80.50 (0.00)   | 69.47 (0.00)   | -     |
| Llama 70B | WDC          | 77.94 (-1.18)  | 55.36 (+3.92)  | 60.56 (+4.94)  | 72.66 (-2.53)  | -        | 69.90 (-10.60) | 63.85 (-5.62)  | -     |
| gpt-4o    | Zero-shot    | 92.20 (0.00)   | 63.45 (0.00)   | 70.67 (0.00)   | 81.64 (0.00)   | -        | 87.18 (0.00)   | 74.59 (0.00)   | -     |
| gpt-4o    | WDC          | 91.99 (-0.21)  | 65.12 (+1.67)  | 68.55 (-2.12)  | 87.07 (+5.43)  | -        | 89.27 (+2.09)  | 80.74 (+6.15)  | -     |

with a decrease of 1.18 F1 points. Thus, fine-tuning improves performance on two out of three in-domain datasets.

In contrast, GPT-40, while showcasing higher performance gains on the WDC dataset, exhibits poorer in-domain generalization. The fine-tuned version only manages to outperform the baseline on one out of three datasets.

The Llama 8B models achieve 62% of the dedicated performance when generalizing within the scholar domain, while GPT-4o-mini attains 66%. These higher generalization scores are likely due to the higher similarity among the scholar datasets, as both benchmarks include records from DBLP.

Cross-Domain Generalization: In a product-to-scholar cross-domain generalization scenario, Llama 8B models experience an average decrease of 11.05 F1 points, while GPT-4o-mini models show a decrease of 2.07 F1 points compared to their zero-shot performance. Specifically, all eight Llama 8B models and four out of eight GPT-4o-mini models underperform relative to their zero-shot baselines, as seen in the DBLP-ACM and DBLP-Scholar columns of Table 2.

This trend continues with the scholar versions of the models. Llama 8B models underperform their zero-shot counterparts in the product domain by 1.11 F1 points, whereas GPT-4o-mini models gain an average of 1.60 F1 points over zero-shot. Larger models exhibit similar patterns: Llama 70B fails to surpass zero-shot performance in cross-domain scenarios, while GPT-4o gains an average of 4.12 F1 points, as reflected in the scholar domain columns of Table 2.

In conclusion, standard fine-tuning significantly enhances indomain generalization for the smaller models, while the results for the larger models as well as for cross-domain generalization are mixed.

# 3.3 Prompt Sensitivity

Variations of the wording of a prompt can have considerable performance implications on the model [10, 15, 17, 27]. It is recommended to use the same prompt for fine-tuning that will later be used for querying the model <sup>5</sup>. In this section, we investigate how the performance of the fine-tuned models changes when a different prompt is used for querying the model.

**Prompts:** We assess the performance of the following three alternative prompts for querying models that were fine-tuned using the prompt shown in Figure 2:

- **simple-free:** "Do the two product descriptions match?"
- **complex-force:** "Do the two product descriptions refer to the same real-world product? Answer with 'Yes' if they do and 'No' if they do not."
- **simple-force:** "Do the two product descriptions match? Answer with 'Yes' if they do and 'No' if they do not."

We measure the prompt sensitivity as the standard deviation of the performance across the prompt that is used for fine-tuning and the three alternative prompts. The detailed prompt sensitivity results can be found in the accompanying repository. Post-fine-tuning Llama 8B's standard deviation drops from 15.76 to 1.87 in a non-transfer setting. The sensitivity increases slightly for indomain transfer scenarios, with product models at 2.27 F1 points and scholar models at 2.60 F1 points. Fine-tuned Llama models show an average prompt sensitivity of 3.54 F1 points, down from 15.76 pre-fine-tuning. Furthermore, the prompt used for fine-tuning does not necessarily perform best with the alternative prompts resulting in the highest F1 score in 29 out of 42 scenarios for Llama.

<sup>5</sup> https://platform.openai.com/docs/guides/fine-tuning

GPT-4o-mini shows a similar trend, with a pre-fine-tuning prompt sensitivity of 2.72 F1 points. Post-fine-tuning, prompt sensitivity is reduced to 0.26 F1 points in non-transfer scenarios, 0.85 F1 points in the product domain, and 0.18 F1 points in the scholar domain. Across all datasets, fine-tuned GPT-4o-mini models exhibites a prompt sensitivity of 1.31 F1 points. With the prompt used for fine-tuning reporting the highest result in 50% of cases.

#### 4 DIMENSION 1: EXAMPLE REPRESENTATION

This section investigates how different training example representations affect the performance and generalization capabilities of fine-tuned models. We compare three representations: 1) simple pairs as used in Section 3, 2) pairs augmented by the LLM with textual explanations why the entities match or do not match, and 3) pairs augmented with structured explanations. The complete prompts for instructing the model to generate explanations as well as the augmented training sets are found in the project repository.

Textual Explanations: We evaluate two types of textual explanations generated using GPT-40-mini. The first approach presents the model with the prompt shown in Figure 2, followed by the label. Afterwards, the model is asked to explain the label. The model is given no guidance on the format of the explanation. This results in detailed explanations, having an average length of 293 tokens, which describe why the entity descriptions match or do not match. As a second approach, we re-run the technique proposed by Wadhwa et al. [21], which generates more concise explanations by including short explanations for other entity pairs as demonstrations in the prompt. The resulting explanations have an average length of 90 tokens. Figure 3 shows a training example containing an explanation in the AI part which was generated using this approach. The prompts used for generating these explanations, as well as the longer, more open-ended explanations, are available in the accompanying repository.

| Actor | Prompt contents   |
|-------|---|
| User  | "Do the two entity descriptions refer to the same real-world product?" Entity 1: Jabra EVOLVE 80 MS Stereo (7899-823-109) Entity 2: Jabra Evolve 80 UC stereo Skype for Business  |
| AI    | "Yes. Both entities refer to the Jabra Evolve 80 headset, which is designed for professional use, specifically for communication and audio purposes. Despite the slight differences in naming conventions (MS vs. UC), they both describe the same product line and model, indicating that they are compatible with similar functionalities. Therefore, they are considered a match." |

Figure 3: Training example containing a textual explanation. The explanation is generated using Wadhwa et al.'s prompt.

**Structured Explanations:** Our third approach is based on the assumption that LLMs can learn more from structured explanations compared to textual ones. Thus, building on the work of Peeters et al. [18], we instruct the model to generate explanations in a structured format that explicitly mention the attributes that were used for the matching decision, their importance for the decision, as well as the similarity of the attribute values. Figure 4 shows an example of a structured explanation that was generated by GPT-40-mini. The complete prompt for the generation of the explanation is found in the project repository and is discussed in [18]. The example

shows that the LLM incorporates factors beyond string matching for calculating similarities, such as recognizing the relationship between Microsoft and Skype for Business.

| Actor | Prompt contents  |
|-------|--|
| User  | "Do the two entity descriptions refer to the same real-world product?" Entity 1: Jabra EVOLVE 80 MS Stereo (7899-823-109) Entity 2: Jabra Evolve 80 UC stereo Skype for Business   |
| Al    | "Yes. attribute=brand   importance=0.05   values=Jabra\#\#\#Jabra   similarity=1.00 attribute=model   importance=0.95   values=EVOLVE 80 MS Stereo\#\#\#Evolve 80 UC   similarity=0.90 attribute=type   importance=0.90   values=Stereo \#\#\\$stereo   similarity=1.00 attribute=compatibility   importance=0.85   values=MS (Microsoft)\#\#\#Skype for Business   similarity=0.80 attribute=SkU   importance=0.00   values=7899-823-109 \#\#\#missing   similarity=0.00" |

Figure 4: Training example including a structured explanation as used during fine-tuning.

#### 4.1 Effectiveness

We compare the effectiveness of different explanation approaches in a non-transfer scenario using the WDC Products dataset. The result of these experiments are found in the No Transfer column of Table 3.

Smaller Models: Llama 8B outperforms standard fine-tuning across all evaluated explanation approaches, as shown in Table 3, column No Transfer. Long textual explanations provide the smallest increase of 1.48 F1 points. The approach by Wadhwa et al. results in a 4.01-point improvement. Structured explanations further improve F1 scores by 0.93 over Wadhwa's approach. GPT-40-mini shows a different trend, where only structured explanations outperform the standard fine-tuning, increasing by 0.97 F1 points (Table 3, WDC Products column). The other configurations underperform, with three out of five approaches performing worse than zero-shot (Table 3, Non-transfer column).

Larger Model: Structured explanations are exclusively tested on larger models due to their consistent performance improvement in smaller models. While Llama 70B does not benefit from standard fine-tuning, structured explanations improve its performance by 1.50 F1 points over zero-shot (Table 3, WDC column). In contrast, GPT-40 does not follow this trend. While structured explanations lead to a 1.60-point increase over zero-shot, standard fine-tuning results in a 5.50-point improvement, as shown in the GPT-40 rows in the WDC column of Table 3. Therefore, GPT-40 does not significantly benefit from structured explanations in a non-transfer setting.

# 4.2 Generalization

We fine-tune the models using the WDC dataset and afterwards apply them to the test sets of the other datasets. We measure the generalization performance within the product domain and across domains using the scholarly datasets.

**In-Domain Generalization:** Standard fine-tuning results in Llama 8B achieving 72% of the dedicated models' performance on product datasets (Table 3, In-Domain Transfer Gain column). The

| Model     | Train set     | No Transfer    | In             | -Domain Transf | er             | Transfer | Cross-Doma     | ain Transfer   | Transfer |
|-----------|---------------|----------------|----------------|----------------|----------------|----------|----------------|----------------|----------|
| Model     | Train set     | WDC            | A-B            | A-G            | W-A            | Gain     | D-A            | D-S            | Gain     |
| Llama 8B  | Zero-shot     | 53.36 (-15.83) | 56.57 (-25.21) | 49.16 (-3.13)  | 42.04 (-11.70) | -        | 85.52 (+11.00) | 67.69 (+0.30)  | -        |
| Llama 8B  | WDC           | 69.19 (0.00)   | 81.78 (0.00)   | 52.29 (0.00)   | 53.74 (0.00)   | 72%      | 74.52 (0.00)   | 67.40 (0.00)   | -30%     |
| Llama 8B  | long textual  | 70.67 (+1.48)  | 83.33 (+1.55)  | 45.95 (-6.34)  | 46.53 (-7.21)  | 51%      | 51.11 (-23.41) | 47.92 (-19.48) | -146%    |
| Llama 8B  | Wadhwa et al. | 73.20 (+4.01)  | 79.00 (-2.78)  | 50.30 (-1.99)  | 48.90 (-4.84)  | 55%      | 69.14 (+5.39)  | 63.35 (-4.05)  | -56%     |
| Llama 8B  | no imp.&sim.  | 73.58 (+4.39)  | 85.25 (+3.47)  | 52.56 (+0.27)  | 55.76 (+2.02)  | 83%      | 55.55 (-18.98) | 51.14 (-16.25) | -125%    |
| Llama 8B  | no importance | 73.82 (+4.63)  | 84.82 (+3.04)  | 54.26 (+1.97)  | 60.00 (+6.26)  | 93%      | 86.06 (+11.63) | 69.19 (+1.79)  | 5%       |
| Llama 8B  | structured    | 74.13 (+4.94)  | 86.89 (+5.11)  | 51.84 (-0.45)  | 59.32 (+5.58)  | 91%      | 79.88 (+5.36)  | 63.67 (-3.72)  | -26%     |
| gpt-4o-m  | Zero-shot     | 81.61 (-1.80)  | 87.68 (-2.77)  | 59.20 (-3.09)  | 65.06 (-2.39)  | -        | 94.16 (+8.81)  | 87.96 (+11.63) | -        |
| gpt-4o-m  | WDC           | 83.41 (0.00)   | 90.45 (0.00)   | 62.29 (0.00)   | 67.45 (0.00)   | 13%      | 85.35 (0.00)   | 76.33 (0.00)   | -55%     |
| gpt-4o-m  | long textual  | 81.30 (-2.11)  | 88.94 (-1.51)  | 61.37 (-0.92)  | 64.23 (-3.22)  | 5%       | 89.75 (+4.40)  | 88.10 (+11.77) | -11%     |
| gpt-4o-m  | Wadhwa et al. | 80.81 (-2.60)  | 84.12 (-6.33)  | 59.03 (-3.26)  | 64.19 (-3.26)  | -14%     | 93.18 (+7.83)  | 87.77 (11.44)  | -3%      |
| gpt-4o-m  | no imp.&sim.  | 81.04 (-2.37)  | 90.95 (+0.50)  | 61.30 (-0.99)  | 66.40 (-1.05)  | 7%       | 92.80 (+7.45)  | 85.73 (+9.40)  | -10%     |
| gpt-4o-m  | no importance | 83.17 (-0.24)  | 90.26 (-0.19)  | 60.71 (-1.58)  | 65.09 (-2.36)  | 4%       | 90.51 (+5.16)  | 84.82 (+8.49)  | -18%     |
| gpt-4o-m  | structured    | 84.38 (+0.97)  | 91.44 (+0.99)  | 64.11 (+1.82)  | 68.92 (+1.47)  | 23%      | 88.87 (+3.52)  | 79.45 (+3.12)  | -37%     |
| Llama 70B | Zero-shot     | 75.20 (+2.50)  | 79.10 (+1.20)  | 51.40 (-4.00)  | 55.60 (-5.00)  | -        | 80.50 (+10.60) | 69.50 (+5.60)  | -        |
| Llama 70B | WDC           | 72.70 (0.00)   | 77.90 (0.00)   | 55.40 (0.00)   | 60.60 (0.00)   | -        | 69.90 (0.00)   | 63.90 (0.00)   | -        |
| Llama 70B | structured    | 76.70 (+4.00)  | 84.80 (+6.90)  | 52.80 (-2.60)  | 65.80 (+5.20)  | -        | 70.10 (+0.20)  | 62.10 (-1.80)  | -        |
| gpt-4o    | Zero-shot     | 81.60 (-5.50)  | 92.20 (+0.20)  | 63.45 (-1.65)  | 70.67 (+2.17)  | -        | 87.18 (-2.09)  | 74.59 (-6.15)  | -        |
| gpt-4o    | WDC           | 87.10 (0.00)   | 92.00 (0.00)   | 65.10 (0.00)   | 68.50 (0.00)   | -        | 89.27 (0.00)   | 80.74 (0.00)   | -        |
| gpt-4o    | structured    | 83.20 (-3.90)  | 90.60 (-1.40)  | 62.80 (-2.30)  | 66.50 (-2.00)  | -        | 84.69 (-4.58)  | 74.90 (-5.84)  | -        |

Table 3: Fine-tuning results for different types of explanations (F1 scores, deltas to standard fine-tuning on WDC).

Wadhwa et al. approach and longer textual explanations underperform, achieving 55% and 51%, respectively. However, structured explanations surpass the fine-tuned baseline, with the *structured* and *no importance* approaches achieving 93% and 91% of source model performance (Table 3, In-Domain Transfer Gain column).

GPT-40-mini continues the trend where four out of five explanation approaches underperform the fine-tuned baseline. However, structured explanations outperform the baseline, achieving 23% of the source models' performance compared to 13% by the fine-tuned baseline (Table 3, In-Domain Transfer Gain column).

Llama 70B benefits from structured explanations, outperforming both the fine-tuned baseline and zero-shot in in-domain generalization. It shows the highest generalization performance on two out of three product datasets (Table 3, In-Domain Transfer Gain column).

As shown in the last row of Table 3, structured explanations do not improve the generalization performance of GPT-40, as this approach underperforms both the fine-tuned baseline and zero-shot.

**Cross-Domain Generalization:** Llama 8B struggles to generalize effectively in cross-domain settings, with all approaches underperforming compared to zero-shot performance. In contrast, GPT-4o-mini shows improvements across all explanation approaches, however non-rival zero-shot performance.

In conclusion, structured explanations significantly enhance indomain generalization, while cross-domain generalization remains limited.

# 5 DIMENSION 2: EXAMPLE SELECTION AND GENERATION

This section explores how the selection and generation of training examples for fine-tuning impact in-domain performance and model generalization. We examine three strategies: filtering the existing training sets, generating new artificial examples based on existing data, and adding additional training examples that are similar to examples on which the model failed. We evaluate the techniques without augmenting the training examples with explanations. We provide the complete prompts for the example selection and generation as well as filtered and augmented training sets in the project repository.

## 5.1 Training Set Filtration

The filtration approach is based on the assumption that training data quality outweighs quantity and it is thus beneficial to remove potentially misleading examples from the training set.

Error-based Filtering: GPT-40-mini is prompted to match pairs in the small WDC training split using the *complex-force* prompt (see Section 3.3). Training examples for which the model's labels differ from the ground truth labels are discarded. This approach may discard some correctly labelled examples. However, it is hypothesized that overall dataset quality improves. The resulting training set WDC-filtered has a size of 2006 examples, compared to 2500 examples of the original WDC-small training set (see Table 4).

**Relevancy-based Filtering:** The second approach involves filtering for relevance, operating on the assumption that fewer, more

Table 4: Impact of the filtration and generation methods on the size the training set.

| Dataset          | # Pos | # Neg | # Total |
|------------------|-------|-------|---------|
| WDC-small        | 500   | 2000  | 2500    |
| WDC-filtered     | 445   | 1561  | 2006    |
| WDC-filtered-rel | 442   | 166   | 608     |
| Syn              | 4932  | 15208 | 20140   |
| Syn-filtered     | 3264  | 10560 | 13824   |
| Syn-filtered-rel | 2182  | 6718  | 8900    |
|                  |       |       |         |

meaningful examples lead to better outcomes. For example, comparing a hard drive and a TV might pass initial filtration but offer limited value due to their low complexity. By removing such examples, computational demands are certainly reduced, while training results may potentially be enhanced. This approach prompts GPT-40 to select only "interesting" training examples, a term we intentionally leave undefined, allowing the model to interpret it. The model appears to define "interesting" as pairs that share many attributes and are thus highly similar (corner cases). Pairs deemed irrelevant by the model are discarded. The resulting WDC-filtered-rel training set has a total size of 608 examples (see Table 4).

**Effectiveness:** As shown in Table 5 column No Transfer, the Llama 8B model trained on the *WDC-s-filtered* and *WDC-s-filtered-rel* datasets outperforms the fine-tuned baseline by 4.73 and 3.18 F1 points, respectively. Both filtered approaches surpass the Llama 8B model trained on the WDC large training set, containing nearly 20,000 examples (see Table 1). In contrast, GPT-4o-mini does not exhibit similar improvements, as both filtered datasets perform below the zero-shot model.

Generalization: While filtration improves Llama 8B's performance, it does not significantly impact in-domain generalization. The fine-tuned baseline captures 72% of dedicated performance, while filtered approaches capture 75% and 76% with relevancy filtration (see Table 5, In-Domain Transfer Gain column). Since GPT-40-mini does not improve its performance with filtration in a no-transfer scenario, the same trend is observed in in-domain generalization, where this approach underperforms the fine-tuned baseline across all product datasets.

For cross-domain generalization, both Llama 8B and GPT-4omini continue to underperform relative to the zero-shot baselines.

## 5.2 Example Generation

While the earlier approaches focused on discarding irrelevant examples, this section investigates using GPT-40 to generate additional training examples for fine-tuning. We test three methods for generating examples. All methods iterate over the WDC small training set and use the examples from this set as seeds derive additional training examples. The complete prompts used by the methods are provided in the project repository.

**Brief Generation Prompt:** This method provides the model with a short task description, instructing it to generate three non-matches and one match. The prompt also includes a seed pair from the training set.

**Detailed Generation Prompt:** This prompt contains a comprehensive task description, including background information on entity matching and specific requirements for the generated examples. It instructs the model to produce examples from the same product category and to include similar matching challenges as the challenges found in the provided seed example. Additionally, the prompt explains key concepts like corner cases to deepen the model's understanding of the task.

**Demonstration-based Creation:** Building on the second method, this approach includes six entity pairs as demonstrations into the prompt. These pairs are selected from the WDC small dataset based on their similarity to the seed example in the OpenAI embedding space.

Inspection of the Generated Examples: We manually inspect a subset of the generated pairs. This inspection reveals that the first method encounters two main issues: generating matching examples with differing strings and maintaining correctness. The model often produces matching examples that are easy non-matches, such as laptops from different product lines being incorrectly treated as matches. The detailed generation prompt results in examples with more variation, though correctness remains mixed. The third approach increases variance further. However, many of these examples remain inaccurate upon human evaluation.

Combining Generation and Filtration: Given the mixed quality of the generated examples, we combine example generation with the filtering approach introduced above, further refined by relevancy filtering (syn.-filter-rel.). The resulting dataset sizes are presented in Table 4. Since the generation methods are intended to extend existing datasets, both training splits are combined with the unfiltered version of the WDC small dataset, and the counts in Table 4 reflect this combination.

**Effectiveness:** Table 5 column No Transfer shows that the Llama 8B model gains 3.35 F1 points without relevancy filtration and 4.85 F1 points with relevancy filtration. In contrast, GPT-4o-mini shows a 6.42 F1 point decrease compared to the fine-tuned baseline, underperforming zero-shot. Due to the high cost of fine-tuning the GPT series and the underperformance of this approach, we chose not to pursue fine-tuning with relevancy filtration.

**Generalization:** In a non-transfer scenario, the generated examples perform similarly to the filtration approach. However, the additional examples enable the models to outperform other approaches in a generalization context. The generated examples with relevancy filtration capture 97% of the dedicated model performance in an in-domain generalization scenario, compared to 72% for the fine-tuned baseline. For GPT-40-mini, both trailed approaches fail to outperform the fine-tuned baseline in in-domain generalization performance. Cross-domain transfer continues the trend of Llama underperforming in this regard, while GPT-40-mini shows a 23% improvement with generated examples.

#### 5.3 Error-based Example Selection

The idea of the error-based example selection (err.-sel.) approach is to guide the model to prevent errors by fine-tuning it with examples that are similar to entity pairs that are matched incorrectly by the model. For this, the Llama 8B model is initially trained on the standard 2,500 examples from the WDC small training dataset. After

| Model    | Train set        | No Transfer    | In-Domain Transfer |                |                | Transfer | Cross-Domain Transfer |                | Transfer |
|----------|------------------|----------------|--------------------|----------------|----------------|----------|-----------------------|----------------|----------|
| Woder    | Ham set          | WDC            | A-B                | A-G            | W-A            | Gain     | D-A                   | D-S            | Gain     |
| Llama 8B | Zero-shot        | 53.36 (-15.83) | 56.57 (-25.21)     | 49.16 (-3.13)  | 42.04 (-11.70) | -        | 85.52 (+11.00)        | 67.69 (+0.30)  | -        |
| Llama 8B | WDC-small        | 69.19 (0.00)   | 81.78 (0.00)       | 52.29 (0.00)   | 53.74 (0.00)   | 72%      | 74.52 (0.00)          | 67.40 (0.00)   | -30%     |
| Llama 8B | WDC-medium       | 67.45 (-1.74)  | 78.80 (-2.98)      | 52.93 (+0.64)  | 54.89 (+1.15)  | 70%      | 75.06 (+0.54)         | 65.22 (-2.18)  | -35%     |
| Llama 8B | WDC-large        | 72.13 (+2.94)  | 70.06 (-11.72)     | 44.89 (-7.40)  | 48.50 (-5.24)  | 28%      | 78.47 (+3.94)         | 56.95 (-10.44) | -48%     |
| Llama 8B | WDC-s-filter     | 73.92 (+4.73)  | 85.12 (+3.34)      | 49.47 (-2.82)  | 54.51 (+0.77)  | 75%      | 80.89 (+6.37)         | 74.29 (+6.89)  | 5%       |
| Llama 8B | WDC-s-filter-rel | 72.37 (+3.18)  | 79.43 (-2.35)      | 54.73 (+2.44)  | 55.68 (+1.94)  | 76%      | 76.49 (+1.97)         | 66.11 (-1.29)  | -29%     |
| Llama 8B | Syn-filter       | 72.54 (+3.35)  | 80.98 (-0.80)      | 51.25 (-1.04)  | 56.65 (+2.91)  | 74%      | 68.37 (-6.15)         | 57.23 (-10.17) | -74%     |
| Llama 8B | Syn-filter-rel   | 74.04 (+4.85)  | 86.00 (+4.22)      | 54.73 (+2.44)  | 59.48 (+5.74)  | 97%      | 75.06 (+0.53)         | 67.20 (-0.20)  | -29%     |
| Llama 8B | WDC-s-err-sel    | 74.37 (+5.18)  | 85.19 (+3.41)      | 52.88 (+0.59)  | 55.80 (+2.06)  | 83%      | 61.99 (-12.53)        | 55.32 (-12.08) | -97%     |
| gpt-4o-m | Zero-shot        | 77.44 (-5.87)  | 85.47 (-4.78)      | 57.20 (-5.14)  | 64.03 (+1.61)  | -        | 94.16 (+8.81)         | 87.96 (+11.63) | -        |
| gpt-4o-m | WDC-small        | 83.31 (0.00)   | 90.25 (0.00)       | 62.34 (0.00)   | 62.42 (0.00)   | 9%       | 75.65 (0.00)          | 76.33 (0.00)   | -55%     |
| gpt-4o-m | WDC-s-filter     | 77.06 (-6.25)  | 81.38 (-8.87)      | 44.67 (-17.67) | 49.84 (-12.58) | -61%     | 92.89 (+7.54)         | 78.34 (+2.01)  | -29%     |
| gpt-4o-m | Syn-filter       | 76.89 (-6.42)  | 84.84 (-5.41)      | 60.29 (-2.05)  | 61.67 (-0.75)  | -2%      | 94.84 (+9.49)         | 79.32 (+2.99)  | -21%     |

Table 5: Fine-tuning with example selection and generation. Deltas against fine-tuning on WDC-s

this initial training phase, the model is validated to identify the remaining errors. Additional training examples are then selected from the large WDC products dataset, as shown in Table 1, simulating additional labelling capacity. These examples are chosen based on their embedding similarity to the identified errors.

To maintain consistent training set sizes, each iteration starts with 2,500 examples from the WDC small dataset, adding 2,500 additional examples based on errors. The model is retrained on the expanded dataset for 5 epochs, and this process is repeated five times. Among all optimization rounds, the model with the highest F1 score on the validation set is selected as the best. Due to OpenAI's fine-tuning limitations, this approach is only tested with the Llama series.

**Effectiveness:** The "WDC-s-err-sel" approach yields the highest F1 score for Llama 8B among all tested methods, with a 5.18-point increase to 74.37, a 1.99-point improvement over the large WDC dataset. This result demonstrates that adding only highly relevant examples outperforms merely increasing training data quantity.

**Generalization:** This approach outperforms the generation method in a non-transfer scenario but underperforms in in-domain generalization, achieving 83% of dedicated performance compared to 72% for the fine-tuned baseline.

In cross-domain generalization, this approach shows a slight improvement over standard fine-tuning, with an average increase of 2.77 F1 points. However, it still lags behind the baseline model, which achieves, on average, 17.95 more F1 points when applied to the scholar domain.

# 6 RELATED WORK

Entity matching [2, 5] has been researched for over 50 years [7]. While early techniques rely on hand-crafted matching rules, supervised machine learning techniques dominate the field today [1, 3, 9, 16, 22–24]. Narayan et al. [14] are the first to apply LLMs for entity matching using GPT-3. [6, 13, 17] extended Narayan's work by testing a wider range of zero-shot and few-shot prompts on more recent models. While these approaches have shown strong

performance, the prompt sensitivity of the LLMs in in-context learning settings has results in a costly and time-consuming search for the best prompt for each model/dataset combination [11, 12, 18]. Zhang et al. [25] fine-tune LLMs using multi-task learning for handling various structured data-related tasks. Their set tasks include entity matching. More recently, Peeters et al. [18] experimented with fine-tuning the GPT3.5-Turbo-0613 model, achieving an F1 score of 88.34 on WDC Products. However, their results cannot be directly compared to the results in this paper as they are generated using only a subset of the WDC test set. Wadhwa et al. [21] are the first to experiment with adding textual explanations to the training set used for fine-tuning LLMs. We extend this idea by augmenting the training set with different types of structured explanations. Wadhwa et al. note improvements, especially in out-of-domain scenarios. In contrast, our work shows that explanations primarily improve results in non-transfer and in-domain generalization settings.

# 7 CONCLUSION

This study demonstrates that standard fine-tuning provides significant performance improvements for smaller LLMs, as well as for certain larger models, such as GPT-40. Small and large models benefit from augmenting the training set with structured explanations, which improve performance as well as generalization. The proposed example selection and generation methods only improve the performance of Llama 3.1 8B while decreasing the performance of GPT-40 Mini. As future work, we plan to refine the example selection and generation methods and develop strategies to improve cross-domain generalization.

#### REFERENCES

- Nils Barlaug and Jon Atle Gulla. 2021. Neural Networks for Entity Matching: A Survey. ACM Transactions on Knowledge Discovery from Data 15, 3 (2021), 52:1–52:37
- [2] Peter Christen. 2012. Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Springer-Verlag, Berlin Heidelberg.
- [3] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, and et al. 2020. An Overview of End-to-End Entity Resolution for Big Data. Comput. Surveys 53, 6 (2020), 127:1–127:42.

- [4] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, and et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI]
- [5] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate Record Detection: A Survey. IEEE Transactions on Knowledge and Data Engineering 19, 1 (2007), 1–16.
- [6] Meihao Fan, Xiaoyue Han, Ju Fan, and et al. 2023. Cost-Effective In-Context Learning for Entity Resolution: A Design Space Exploration. arXiv:2312.03987 [cs]
- [7] Ivan P. Fellegi and Alan B. Sunter. 1969. A Theory for Record Linkage. J. Amer. Statist. Assoc. 64, 328 (1969), 1183–1210.
- [8] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of Entity Resolution Approaches on Real-World Match Problems. Proceedings of the VLDB Endowment 3, 1-2 (Sept. 2010), 484–493.
- [9] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, and et al. 2020. Deep Entity Matching with Pre-Trained Language Models. Proceedings of the VLDB Endowment 14, 1 (2020), 50–60.
- [10] Pengfei Liu, Weizhe Yuan, Jinlan Fu, and et al. 2023. Pre-Train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. Comput. Surveys 55, 9, Article 195 (2023).
- [11] Yao Lu, Max Bartolo, Alastair Moore, and et al. 2021. Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity. arXiv preprint arXiv:2104.08786 (2021). arXiv:2104.08786
- [12] Katerina Margatina, Timo Schick, Nikolaos Aletras, and et al. 2023. Active Learning Principles for In-Context Learning with Large Language Models. arXiv preprint arXiv:2305.14264 (2023). arXiv:2305.14264
- [13] Navapat Nananukul, Khanin Sisaengsuwanchai, and Mayank Kejriwal. 2024. Cost-Efficient Prompt Engineering for Unsupervised Entity Resolution. arXiv:2310.06174 [cs.AI]
- [14] Avanika Narayan, Ines Chami, Laurel Orr, and et al. 2022. Can Foundation Models Wrangle Your Data? Proc. VLDB Endow. 16, 4 (Dec. 2022), 738–746. https://doi.org/10.14778/3574245.3574258
- [15] Avanika Narayan, Ines Chami, Laurel Orr, and et al. 2022. Can Foundation Models Wrangle Your Data? Proceedings of the VLDB Endowment 16, 4 (2022), 738-746.

- [16] Ralph Peeters and Christian Bizer. 2021. Dual-Objective Fine-Tuning of BERT for Entity Matching. Proceedings of the VLDB Endowment 14, 10 (2021), 1913–1921.
- [17] Ralph Peeters and Christian Bizer. 2023. Using ChatGPT for Entity Matching. In New Trends in Database and Information Systems (Communications in Computer and Information Science). Springer Nature Switzerland, Cham, 221–230.
- [18] Ralph Peeters and Christian Bizer. 2024. Entity Matching Using Large Language Models. arXiv:2310.11244 [cs.CL]
- [19] Ralph Peeters, Reng Chiz Der, and Christian Bizer. 2024. WDC Products: A Multi-Dimensional Entity Matching Benchmark. In Proceedings of the 27th International Conference on Extending Database Technology, Paestum, Italy, March 25 - March 28, 22–33.
- [20] Ashish Vaswani, Noam Shazeer, Niki Parmar, and et al. 2017. Attention Is All You Need. In Proceedings of the 31st International Conference on Neural Information Processing Systems. 6000–6010.
- [21] Somin Wadhwa, Adit Krishnan, Runhui Wang, and et al. 2024. Learning from Natural Language Explanations for Generalizable Entity Matching. arXiv:2406.09330 [cs.CL]
- [22] Jin Wang, Yuliang Li, and Wataru Hirota. 2021. Machamp: A Generalized Entity Matching Benchmark. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 4633–4642.
- [23] Dezhong Yao, Yuhong Gu, Gao Cong, and et al. 2022. Entity Resolution with Hierarchical Graph Attention Networks. In Proceedings of the 2022 International Conference on Management of Data. 429–442.
- [24] Alexandros Zeakis, George Papadakis, Dimitrios Skoutas, and et al. 2023. Pre-Trained Embeddings for Entity Resolution: An Experimental Analysis. Proceedings of the VLDB Endowment 16, 9 (2023), 2225–2238.
- [25] Haochen Zhang, Yuyang Dong, Chuan Xiao, and et al. 2023. Jellyfish: A Large Language Model for Data Preprocessing. arXiv:2312.01678 [cs]
- [26] Wayne Xin Zhao, Kun Zhou, Junyi Li, and et al. 2023. A Survey of Large Language Models. arXiv preprint arXiv:2303.18223 (2023). arXiv:2303.18223
- [27] Zihao Zhao, Eric Wallace, Shi Feng, and et al. 2021. Calibrate Before Use: Improving Few-Shot Performance of Language Models. In Proceedings of the 38th International Conference on Machine Learning. 12697–12706.