

Cost-Efficient Prompt Engineering for Unsupervised Entity Resolution

Navapat Nananukul¹, Khanin Sisaengsuwanchai¹,
Mayank Kejriwal¹

¹ University of Southern California, Information Sciences Institute, Los Angeles, CA, United States of America.

Contributing authors: nananuku@isi.edu; sisaengs@isi.edu;
kejriwal@isi.edu;

Abstract

Entity Resolution (ER) is the problem of semi-automatically determining when two entities refer to the same *underlying* entity, with applications ranging from healthcare to e-commerce. Traditional ER solutions required considerable manual expertise, including domain-specific feature engineering, as well as identification and curation of training data. Recently released large language models (LLMs) provide an opportunity to make ER more seamless and domain-independent. However, it is also well known that LLMs can pose risks, and that the quality of their outputs can depend on how prompts are engineered. Unfortunately, a systematic experimental study on the effects of different prompting methods for addressing unsupervised ER, using LLMs like ChatGPT, has been lacking thus far. This paper aims to address this gap by conducting such a study. We consider some relatively simple and cost-efficient ER prompt engineering methods and apply them to ER on two real-world datasets widely used in the community. We use an extensive set of experimental results to show that an LLM like GPT3.5 is viable for high-performing unsupervised ER, and interestingly, that more complicated and detailed (and hence, expensive) prompting methods do not necessarily outperform simpler approaches. We provide brief discussions on qualitative and error analysis, including a study of the inter-consistency of different prompting methods to determine whether they yield stable outputs. Finally, we consider some limitations of LLMs when applied to ER.

Keywords: large language models, prompt engineering, unsupervised entity resolution, inter-consistency of prompting

1 Introduction

Entity Resolution (ER) is (at least) a 50 year-old problem that has been studied in many real-world domains [22, 33, 11, 24, 12]. ER can be defined as the algorithmic problem of determining when two or more entities refer to the same underlying entity. Ironically, ER itself has been studied under many names, including entity matching, instance matching, deduplication, and record linkage, to just name a few [18]. In real-world domains like healthcare and e-commerce [22], ER can be a complex problem (sometimes, even for human beings). Figure 1 provides a representative example in e-commerce.

Many promising solutions to ER have been proposed over the decades, and significant progress has been achieved. Nevertheless, we are far from solving the problem at human performance levels, and errors can be costly. While initially (and in some contexts still), rule-based and manually engineered solutions were prevalent [47, 67], machine learning methods became increasingly popular as ER solutions, as with many other problems amenable to learning from data, starting from the early-mid 2000s [63, 32]. Over the last decade, deep learning solutions have also been applied to ER [16, 15, 43, 31, 35], and even more recently, transformer-based models such as BERT [38]. While performance has steadily improved, general performance on difficult datasets remains subpar.

At the same time, because of the domain-specific nature of ER (e.g., ER in e-commerce [25] can be significantly different than in a healthcare setting [28], where precision requirements might be stronger), manual engineering continues to play an important role, requiring considerable effort in setting up and executing an ER architecture. One issue is that a typical ER workflow comprises two steps: *blocking* and *similarity* [37, 3]. Blocking is necessary to mitigate the quadratic complexity of ER. This problem arises because, given n entities or ‘records’, and a matching function f that indicates whether a pair of records matches or not, a naive ER solution would need $O(n^2)$ comparisons, which is not realistic [49, 34]. A blocking function clusters *approximately similar* records together in *sub-quadratic* runtime, at least in practice. Only records within these clusters, called blocks, are then compared with one another using a similarity function that is typically trained using machine or deep learning.

In recent years, there has been enormous progress on learning-based and even unsupervised blocking [46], but it is less clear how to apply LLMs to blocking. In this study, we focus solely on similarity, as our intent is to study whether a large language model (LLM) can be used as a good similarity method. However, keeping the expensive nature of ER in mind, one of our goals is to consider a range of reasonable prompting methods (as similarity functions), some of which are more expensive but may also provide higher performance. For instance, we consider whether adding a persona to the prompt, or providing more detailed instructions (or more finely structured inputs), all of which lead to more tokens in the input and hence, higher *per-prompt* cost, end up yielding proportionately better performance.

Somewhat more broadly, we also seek to explore whether an established, widely-in-use LLM like ChatGPT offers a promising unsupervised technique of telling when two records are the same or not, at least in a significant application area such as e-commerce. However, clear evidence of this has been lacking in the literature due to the

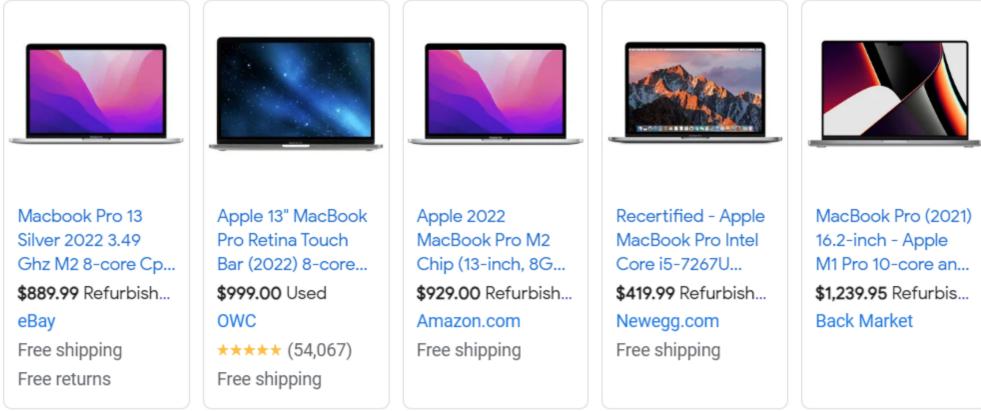


Fig. 1 An illustrative example of products from Google Product Search offering the same product from different sources

recency of the LLMs. At the same time, because of the growing literature on prompt engineering [74, 71, 61, 68, 72], there is considerable evidence showing that LLMs can be sensitive to the manner in which they are prompted.

Motivated by both of these observations, we propose an empirical case study that aims to provide systematic guidance on how prompt engineering affects the performance of an LLM like GPT-3.5 on ‘unsupervised’ ER (where no explicit training or fine-tuning is conducted). Specific experimental questions that we consider include: what is the general performance of GPT-3.5 on ER? Is it the case that more expensive and detailed prompt engineering methods necessarily outperform simpler, less expensive and more obvious methods? We investigate these questions using six prompt engineering methods on two real-world e-commerce benchmark datasets that have been extensively used in the literature. At least one of these (based on matching products across Google and Amazon) has been found to be challenging even for state-of-the-art ER systems [52, 51, 1].

The rest of this article is structured as follows. First, we begin with a discussion of related work in Section 2, followed by some background and preliminaries on ER in Section 3. Next, we describe six different prompting ‘patterns’ (or methodologies) for conducting ER in Section 4. These prompting patterns form the basis of the case study. We describe the actual experimental study and results in Section 5, before concluding the article in Section 7.

2 Related Work

Entity Resolution. Entity Resolution (ER) is a fundamental task in data management and quality, aimed at identifying and linking records that refer to the same real-world entity across different data sources, or even within a single source. Typically, ER methods for structured data [18] include at least a *blocking* step and a *matching* step. This paper focuses on the latter, where the matching step partitions a candidate

set of pairs (obtained through blocking) into matching pairs (duplicates) and non-matching pairs (non-duplicates). Traditional methods for performing ER tend to be grouped into two main categories: *machine learning-based* and *rule-based*.

Previous research has highlighted machine learning-based methods for ER, utilizing classifiers trained on labeled datasets to identify duplicates. Commonly used algorithms in the past have included *Support Vector Machines (SVM)*, *Naive Bayes*, and *Decision Trees* [10, 6, 14, 56]. These learning-based methods first extract features from each record pair in the candidate set using the full set of attributes in the dataset’s schema and uses the extracted features to train a binary classifier. One important step required for the learning-based methods is selection of appropriate classification features for the model (feature engineering). Feature engineering is inherently labor-intensive and expertise-driven, often becoming a bottleneck in this ER process. More recently, deep learning ER methods have proposed to eliminate this step through automatic (neural network-based) representation learning [16, 27].

Rule-based methods [19, 58, 60] compute the similarity between corresponding attribute values using similarity metrics [13]. The final ER decisions are based on pre-defined rules and thresholds. Since rule-based methods rely on the establishment of explicit rules and thresholds that determine how entities are matched, feature engineering is not required; however, the formulation of rules can still be a cumbersome process and may not be data-driven or optimal. Hence, the efficacy of rule-based methods significantly depends on the expertise of domain specialists who devise and fine-tune the specific rules and thresholds. The involvement of these experts is critical to ensuring that the rules accurately reflect the nuances and complexities of the specific domain of the datasets.

Large Language Models (LLMs). Recent years have seen significant progress on generative AI models, of which the large language models like ChatGPT mark an important milestone. Multiple LLMs have been developed and released recently. Notable examples include OpenAI’s ChatGPT [45], Meta’s LLaMA [65], and Google’s Gemini [64]. LLMs have rapidly become popular for a range of problems involving both text (and more recently, multi-modal inputs) and have proven adept at (i) processing natural language context, (ii) generating human-like text, and (iii) using the trained knowledge-intensive corpus (on which they have been pretrained) to engage in ‘in-context’ learning and respond to a variety of prompts. It is this last ability of LLMs that we study in this paper specifically in the context of nearly-unsupervised ER. Applications of LLMs span across multiple domains related to natural language processing, e.g., search [2], customer support [62], and translation [30], and continue to grow. Even in more domain-specific or specialized scenarios, such as healthcare [68, 41], academic writing [26], and visual generation [59], LLMs have tended to perform competitively, and have augmented human abilities in impressive ways.

The use of LLMs for ER is a new area of study. An obvious approach is to use LLMs to act as a similarity measure since they have pre-trained knowledge about real-world entities. Following this approach, LLMs provide a possibility to mitigate some of the inherent challenges of traditional methods, such as feature engineering and rule-construction by domain experts. While LLMs do require ‘prompt engineering’, they are usually robust to prompts in many task-areas. Nonetheless, the manner in which

the LLM is prompted can make a difference, as we also explore in this paper. In related work on using LLMs for ER and other such problems, recent LLM-based research in knowledge graph construction has explored how prompt engineering techniques, such as few-shot prompting [39], can enhance information extraction using LLMs [57, 53, 29]. Another paper proposed a knowledge graph construction framework by performing entity extraction and triple generation [5]. To our knowledge, there is a gap in research regarding the use of LLMs as a similarity function for entity resolution, as well as prompt engineering LLMs for ER. Recognizing this gap, we provide a detailed case study in this paper using GPT-3.5 for the matching step. We also report insights into how different prompting patterns affect GPT-3.5’s ER performance relative to the cost, and supplement our quantitative results with qualitative analyses using responses obtained from GPT-3.5.

3 Background: Entity Resolution

This section provides some technical background on ER to lay the groundwork for the remainder of the article. As intuited earlier, the goal of ER is to identify which records (also called “entity profiles” or “mentions” [48, 4]) from one or more data sources refer to the same real-world entity. The challenge arises from the fact that the same real-world entity can be represented in different ways, making it difficult for machines to follow a rigid set of rules for matching entities. In contrast, the problem is easy for most humans in common domains.

To study the problem algorithmically, we begin by defining an *entity profile* e_{id} as a pair (id, A_{id}) , where id is a unique identifier for the profile, and A_{id} is a *dictionary* of attributes describing each profile i.e., A_{id} may be expressed as a set of key-value pairs: $(attribute_name, attribute_value)$. We denote a set E of entity profiles as an *entity collection*. We provide examples of entity profile representations of the first two laptops in Figure 1.

Given an entity collection E , the task of ER is to partition the entity profiles in E into clusters, such that each cluster represents a unique entity [23, 9]. This general problem affords many variants, the most common one of which is to find *pairs* of entity profiles (rather than clusters), such that each pair represents a ‘match’ or a ‘duplicate’. One reason for interpreting ER as a pairwise problem is that, given matches, often with accompanying similarity scores, graph clustering algorithms can be used to combine matches into clusters. In many applications, matching pairs are enough and more general clusters are not needed. Hence, most ER benchmarks are set up to evaluate the pairwise version of the problem. We assume the same here as well.

Formally, we define the task of ER as discovering all duplicates (e_i, e_j) , with $e_i, e_j \in E$. Such (predicted) duplicates are then evaluated with respect to a *ground-truth* of (typically, manually labeled) duplicates, similar to other problems in machine learning. In its supervised variant, some fraction of these manually labeled pairs are provided to the algorithm as input, which then has to learn from them and predict the remaining duplicates. Any pair not labeled as a duplicate is automatically considered as a non-duplicate. One other point to note is that, although we assumed a single entity collection E , in practice, it is not uncommon to assume *two* entity collections E_1

and E_2 that are individually de-duplicated, but that have matching entities between them. We see this situation reflected in Figure 1: the laptop from eBay would need to be matched to the laptop from OWC, the laptop from Amazon.com, and so on. If two entity collections (e.g., eBay and Amazon) are assumed, as in this article, then the problem of ER can be defined analogously as discovering all duplicates (e_i, e_j) between E_1 and E_2 ($e_i \in E_1, e_j \in E_2$).

The performance of an ER system is measured both by its effectiveness and efficiency. *Effectiveness* refers to the number of actual duplicates that the ER algorithm is able to find (among all the duplicates in the ground-truth), while minimizing the number of false positives (predicted duplicates that are actually non-duplicates). It is measured using metrics like precision, recall, and F-measure, and that we briefly describe in Section 5. *Efficiency* refers to the system’s computational cost, which is quadratic in the worst case ($O(|E_1||E_2|)$), even if the matching or ‘similarity function’ is known beforehand. If the function ran in constant time, a brute-force approach would still have to perform *all* pairwise comparisons between the collections, which quickly runs into the millions even for a few thousand entity profiles in each collection.

To mitigate this brute-force quadratic complexity, the ER community has developed a family of approaches called *blocking* that clusters approximately similar entities into blocks, typically using indexing-like algorithms that run in near-linear time. Even simple blocking can help eliminate a large fraction of unnecessary comparisons. For example, given entity profiles representing people, some of which are duplicates, a blocking algorithm might use the birthyear and the first two digits of the person’s address-zipcode to only compare pairs of people with both of these ‘blocking keys’ in common. Many sophisticated blocking algorithms now exist; we refer the interested reader to [50, 37] for comprehensive surveys and approaches.

The final ER result is determined by using the similarity function on pairs of entity profiles that share blocks (indexed by blocking keys, such as in the example above). Over the decades, many different types of similarity functions have been proposed, ranging from rule-based approaches to string similarity functions (e.g., edit distance) and more recently, machine learning and deep learning (as discussed earlier in the related work) [42, 44, 73]. However, a systematic evaluation of LLMs as similarity functions, especially when prompted in different but reasonable ways, has been lacking. Next, we describe six such prompting methodologies that can be used as LLM-based in-context similarity functions for low-supervision ER.

4 Prompt Engineering Methods for ER

This section discusses the the prompt engineering methods we used to study GPT-3.5’s ER performance. We begin by describing the underlying ER workflow (when an LLM like GPT-3.5 is used as the similarity function), followed by six specific prompting ‘patterns’ underlying our experiments.

Figure 2 illustrates an ER workflow where an LLM serves as the similarity function. While the LLM is technically a ‘black box’ in that it is infeasible to re-train it from scratch, or (without training data) fine-tune it, it can be controlled through prompting.

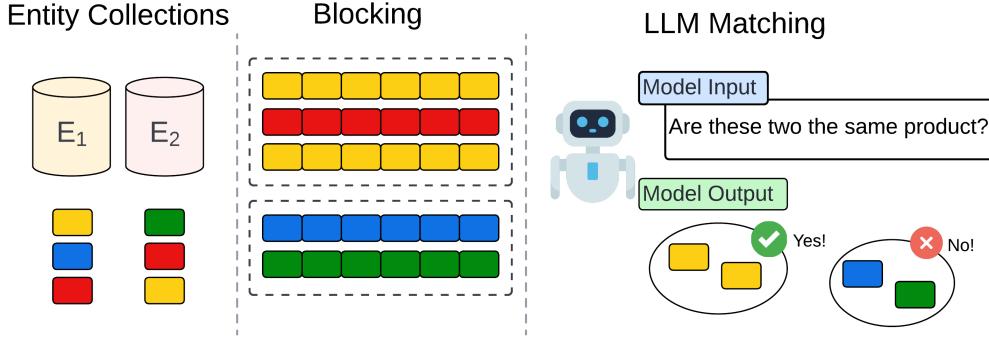


Fig. 2 An illustrative example of a typical ER workflow, but with an LLM used as the similarity (or ‘matching’) function. Each entity collection represents a structured ER dataset, with individual entities represented using colored boxes. As explained in Section 3, blocking is first applied to cluster approximately similar entities into blocks, in order to mitigate the quadratic complexity of comparing all pairs of entities. Only entities sharing a block are paired and presented to the LLM for making a final decision on whether they match (yes) or not (no).

One example is *few-shot prompting*, a technique that integrates a small set of question-and-answer examples, referred to as *demonstrations*, into the prompts [39]. Few-shot prompting enables in-context learning by guiding LLMs to respond based on the patterns and logic observed in the provided demonstrations. In more advanced techniques, such as *Chain-of-Thought (CoT) prompting* [70, 69], intermediate reasoning steps are added to question-and-answer demonstrations. This technique directs LLMs to mimic these reasoning steps when generating responses. Prompt engineering is the process of designing, refining, and implementing prompts that guide the LLMs to output better results [17]. The original approach in the prompt engineering process involves crafting the main components of the prompt, such as prompt instruction, context/persona, and output format. Despite the simplicity of its implementation, effective prompt engineering is a non-trivial process, especially if we take both costs and benefits into account. For instance, it is not always clear that longer prompts yield better performance than short prompts, even though the former is more expensive (involving more tokens) than the latter. Prompt engineering has been actively researched in domains ranging from graph analytics [21] and healthcare [41] to business process management [7], but to our knowledge, remains to be systematically studied in ER.

We construct a prompt template to guide GPT-3.5 in performing ER. In order for GPT-3.5 to produce ER results, the prompt template must include three main components: a *candidate pair* (that we are seeking to determine is a match or non-match), *ER instructions*, and a *result format*. Modification to these components results in different *prompt patterns* that influence the LLM’s ER results. To study the influence of prompting, we construct several such prompt patterns by altering the three main components in the prompt template. Besides altering these main components, prompt patterns can also be generated by appending *optional* components that help influence ER results. We experimented with two such optional components in this study: *persona* and *few-shot examples*. The *persona* component makes the LLM’s role more specific by defining a character profile that outlines how the LLM should behave. Users

can set up the persona as a preset by adding a text description when using GPT-3.5. Figure 3 illustrates a text explanation of the ER expert persona we used in this experiment. In contrast, as previously discussed, *few-shot examples* guide the LLM to generate responses by learning in-context from the demonstrations embedded within these examples. In ER, we can use sampled pairs with labels from the ground-truth as demonstrations to generate a few-shot prompt pattern.

In the following subsections, we illustrate these six prompt patterns and describe the rationale behind their creation. Throughout this section, we use a candidate pair in Table 1 from the Amazon-Google dataset to illustrate the prompt patterns.

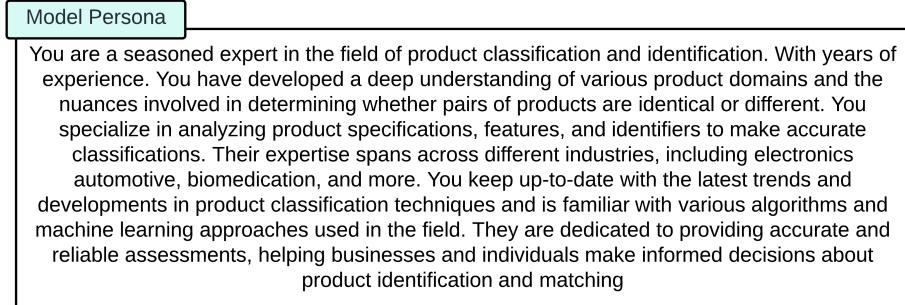


Fig. 3 A text description of the *persona* describing the conceptual role of GPT-3.5 when performing ER. Since this study uses a dataset in the product domain, we set up the role of GPT-3.5 as an expert on product classification, with an emphasis on using related knowledge from multiple product domains to resolve the pair.

Datasets	Product Title	Manufacturer	Product Descriptions
Amazon	Apple final cut studio 2 (mac)	Apple	final cut studio 2 delivers an integrated post-production solution that lets you move effortlessly ...
Google	Apple final cut studio 2 production suite for mac av production software	N/A	final cut studio 2 production software suite for mac - final cut pro 6 motion 3 soundtrack pro 2 color compressor 3 dvd studio pro 4 ...

Table 1 Example of a candidate pair from the Amazon-Google dataset featuring the product ‘Apple final cut studio 2’, with key attributes displayed: *product title*, *manufacturer*, and *product description*.

4.1 Single-Attribute Prompt Pattern

The *single-attribute* (henceforth, *single-attr*) is the simplest and cheapest prompting pattern out of the six patterns. This pattern assumes that there is a single attribute containing the appropriate information to enable the model to make a good matching decision, and that we (as human prompters) are able to identify this attribute. While the assumption seems restrictive, it is a good baseline, as in many cases, domain experts have an intuition for which attributes contain high ‘information density’.

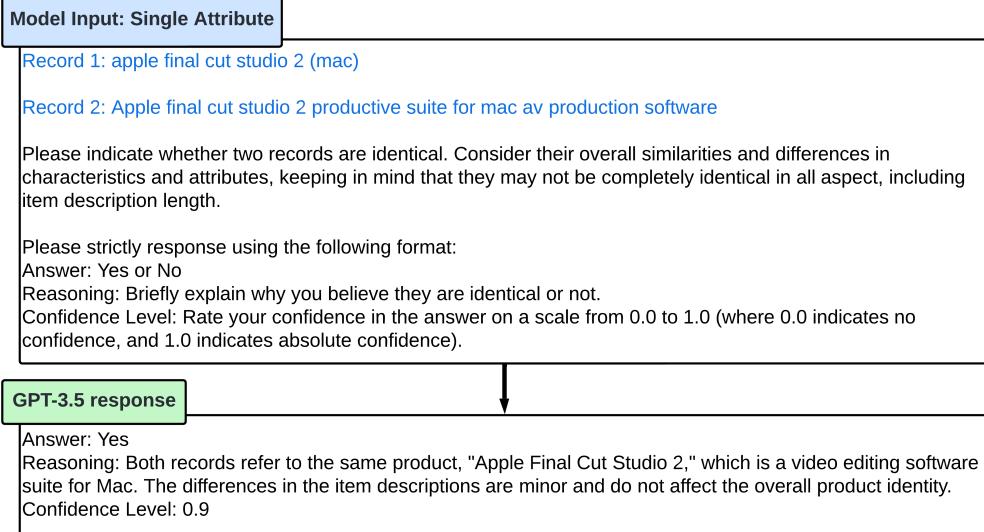


Fig. 4 An illustrative example of the *single-attr* prompt pattern. The example includes two records for comparison: both are versions of 'Apple Final Cut Studio 2'. The records information only contains product titles (highlighted in blue). The task instruction asks GPT-3.5 to determine if the records refer to identical products, considering their similarities and differences. The GPT-3.5 response shows that the records are identical and explains the reasoning behind the decision, along with an associated confidence level.

In the context of the e-commerce domain, to construct the pattern, we selected the *product title* as the primary attribute for this pattern because of its concise and descriptive nature, which can be useful for the LLM in identifying products within e-commerce benchmark datasets, such as WDC and Amazon-Google Products (subsequently detailed). The advantage of this pattern is its cost efficiency due to significant token reduction by using only one attribute. However, it has the obvious drawback of making mistakes when the information that can differentiate two records happens to be in *other* attributes. The prompt example is shown in Figure 4, where GPT-3.5 concludes the result based on information from the product title (highlighted in blue).

4.2 Multi-Attribute Prompt Patterns

Unlike *single-attr*, which uses one attribute to represent candidate pairs, the subsequent prompt patterns utilize multiple attributes from an ER dataset without any preprocessing. These patterns can be categorized into two methods: (1) *multiple attributes without using the LLM persona* (henceforth, *no-persona*) and (2) *multiple attributes with persona* (henceforth, *multi-attr*). The distinction between these two patterns is the option to use or omit the ER expert persona. Specifically, the *no-persona* pattern refers to omitting the user-defined textual information that characterizes the LLMs' given 'role', as explained earlier in this section. The *no-persona* is the only pattern that omits the persona, while every other method in this study includes a persona in the model. In contrast, *multi-attr* integrates the persona in

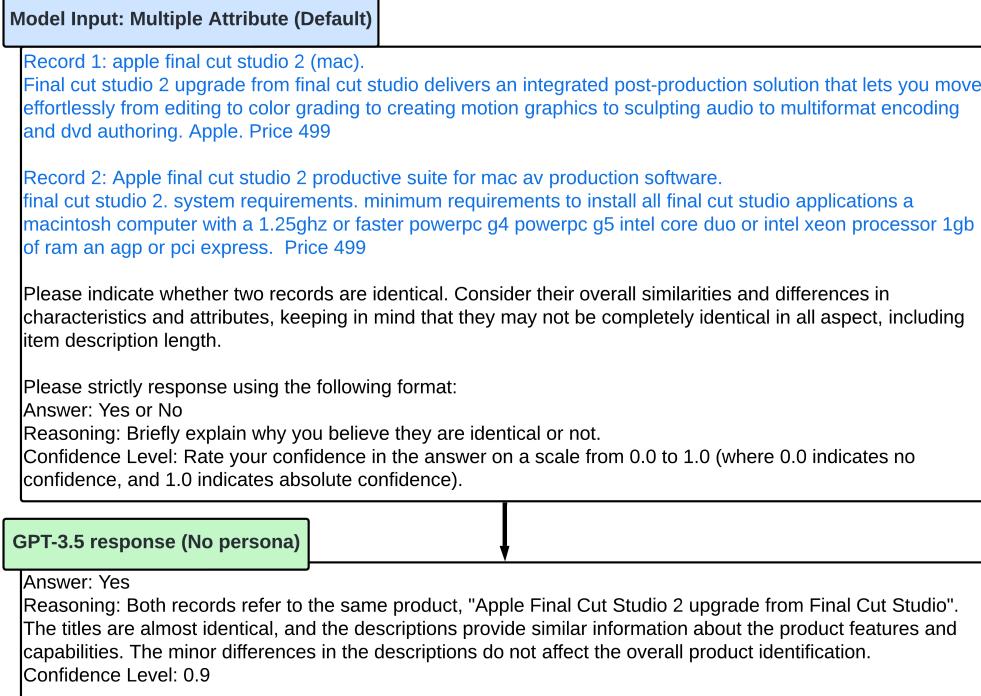


Fig. 5 An illustrative example of the *multi-attr* and *no-persona* prompt patterns, comparing records with multiple attributes concatenated (highlighted in blue). As with the *single-attr* pattern, GPT-3.5 is tasked with evaluating whether the detailed product records, now including additional attributes like *manufacturer*, *description*, and *price*, refer to the same underlying product. The GPT-3.5 *response format* remains the same as the *single-attr* pattern.

Figure 3 with multiple attributes, as demonstrated in Figure 5, where the candidate pair is presented as a concatenation of all attributes in the dataset.

4.3 Multi-Attribute with JSON Prompt Pattern

One problem with concatenation is that it might make the content ambiguous; hence, we also consider a machine-readable structured version called *multiple attributes with JSON format* (henceforth, *multi-json*) whereby the attributes are represented in a JSON-like format, with the schema information. The format retains the same content as the *multi-attr* method. We hypothesized that a machine-readable format would enhance LLMs' understanding of candidate pairs' attributes, thereby enhancing the LLMs' performance. Figure 6 illustrates an example of JSON-formatted pairs, highlighted in blue, where attribute names serve as keys and are paired with their respective values.

Model Input: Multiple Attribute (JSON)

Record 1: {"**product_title**": apple final cut studio 2 (mac), "**product_descriptionManufacturerPrice**

Record 2: {"**product_title**": Apple final cut studio 2 productive suite for mac av production software, "**product_descriptionbrandPrice**

Please indicate whether two records are identical. Consider their overall similarities and differences in characteristics and attributes, keeping in mind that they may not be completely identical in all aspect, including item description length.

Please strictly response using the following format:

Answer: Yes or No

Reasoning: Briefly explain why you believe they are identical or not.

Confidence Level: Rate your confidence in the answer on a scale from 0.0 to 1.0 (where 0.0 indicates no confidence, and 1.0 indicates absolute confidence).

GPT-3.5 response

Answer: Yes

Reasoning: Both records refer to the same product, "Apple Final Cut Studio 2 upgrade from Final Cut Studio". The titles are almost identical, and the descriptions provide similar information about the product features and capabilities. The minor differences in the descriptions do not affect the overall product identification.

Confidence Level: 0.9

Fig. 6 An illustrative example of *multi-json* prompt pattern, showing a product pair in the JavaScript Object Notation (JSON) format. The attributes are organized into a structured format, which includes the *product title*, *description*, *manufacturer*, and *price* for each record (JSON keys are highlighted in blue to emphasize the structure). The remaining instructions are the same as those for the *multi-attr* prompt pattern.

4.4 Multi-Attribute with Similarity Prompt Pattern

As discussed thus far, we defined a prompt instruction as directing an LLM to decide whether records are duplicates or non-duplicates. In this prompt pattern (*multiple attributes with similarity*; henceforth, *multi-sim*), we modified the prompt instruction for GPT-3.5 to generate a *similarity score* between two products, instead of giving a definitive ER decision (match or non-match). We did not specify a similarity metric, such as edit distance, in the prompt. Instead, we rely on GPT-3.5's judgment to generate the similarity score on its own. Figure 7 displays an example of this pattern, highlighting the modified instruction and similarity score in blue. After obtaining similarity scores for each candidate pair, we choose a threshold θ that yields the best F-1 score for the entire dataset to compute the final ER results.

4.5 Few-Shot Examples Prompt Pattern

Finally, we consider a *few-shot* prompting method that is nearly (but not fully) unsupervised. This prompt pattern presents the model with three randomly selected pair examples from the training partition of the corresponding benchmark dataset, which

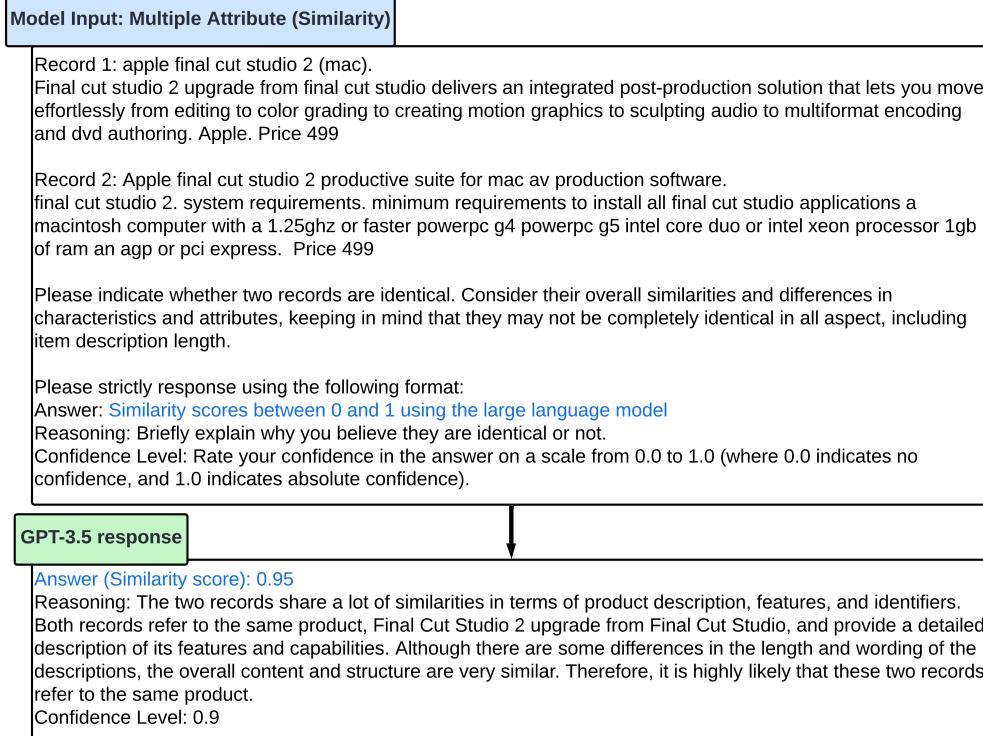


Fig. 7 An illustrative example of the *multi-sim* prompt pattern, where the key difference is the instruction for GPT-3.5 to provide a similarity score ranging from 0 to 1 (highlighted in blue). The remaining structure aligns with that of the *multi-attr* pattern.

is otherwise unused for the other methods. Figure 8 illustrates few-shot examples, highlighted in blue, where we concatenated two duplicate demonstrations and one non-duplicate demonstration into the original *multi-attr* prompt pattern. As stated earlier, few-shot prompting enables in-context learning to guide LLMs’ responses, with the intuition that the model can make a more informed decision if it gets concrete examples of what constitutes duplicate and non-duplicate pairs.

5 Experimental Study

E-commerce is a challenging domain because entities within it are both attribute-rich and can have small differences (such as a 13-inch versus a 15-inch laptop) that make two products different, even though other attributes may be the same. This allows us to check both the robustness and generalization of LLMs like GPT-3.5, which are sometimes known to hallucinate [40, 8], or otherwise focus too much on language compared to the knowledge embedded in the language. Our experimental design is inspired by previous research [25, 66], where the ER problem (called *product ER*) takes entity collections of real-world product-profiles as input, and needs to find pairs of profiles referring to the same underlying product. In the datasets described

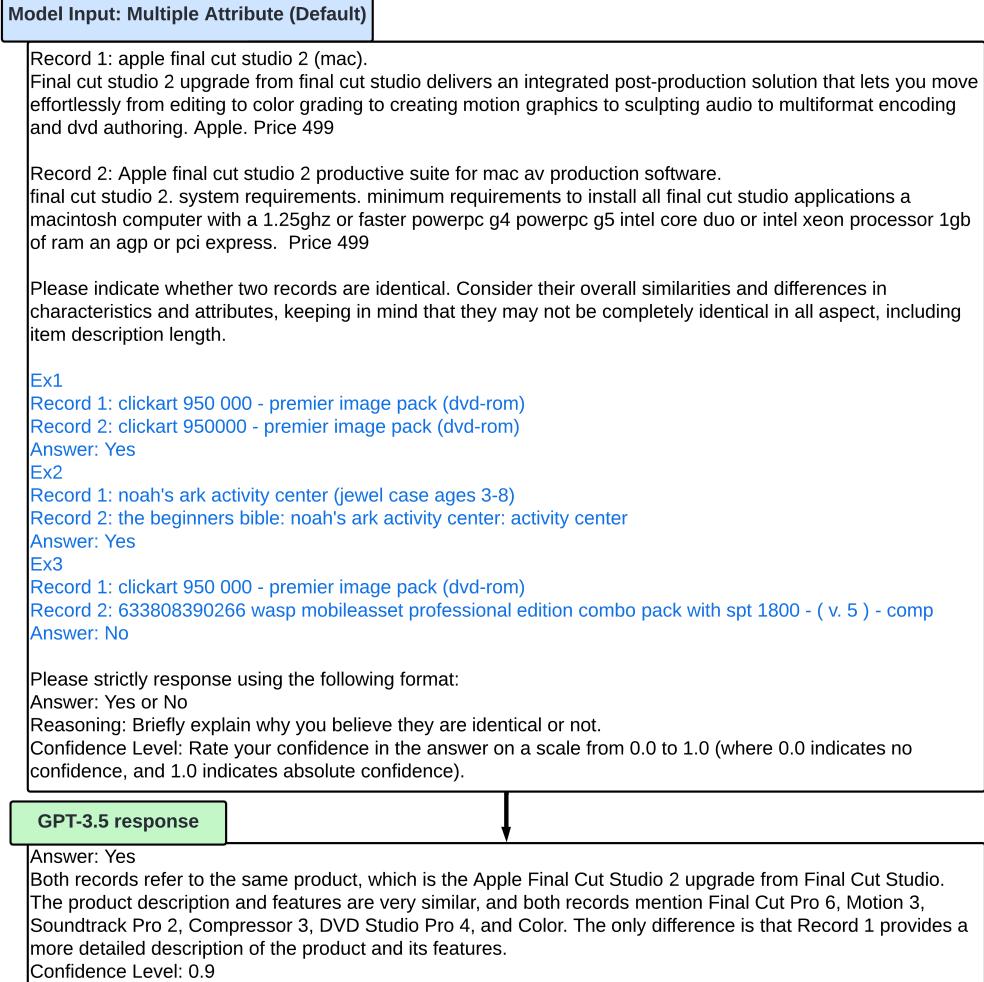


Fig. 8 An illustrative example of the *few-shot* prompt pattern, incorporating three examples (two duplicates and one non-duplicate) highlighted in blue. These examples are added to the *multi-attr* prompt pattern, with the rest of the prompt structure remaining unchanged.

below, thousands of merchants offer a variety of products using different names and descriptions. Products from multiple platforms can have heterogeneous descriptions, including missing values.

Earlier, we illustrated an example in Figure 1 for the Google Product keyword search *Macbook pro 2022*. The search results showed differences in product names and descriptions across different merchants. Not only do different merchants use heterogeneous names and descriptions for describing the same product but the order of details and the way that the text is written is also different. This real-world example underscores the difficulty of product ER. We also considered product ER to be a good application area because studies by other authors [55, 20] have demonstrated LLMs'

general capability to reason over e-commerce products in other applications. However, in product ER, LLMs’ abilities have not been measured or commented upon in depth.

We use two publicly available and real-world ER benchmarks in the e-commerce domain for this study:

1. **Web Data Commons (WDC)** [54]: The full version of this dataset contains 26 million products and descriptions from different e-commerce websites scraped from the Web as part of the Web Data Commons (Common Crawl). The products are categorized based on computers, cameras, shoes, and watches. We selected products of the *computer* type as the test dataset. This smaller curated dataset consists of seven attributes and comprises 1,100 pairs, of which 300 are duplicates and the rest are non-duplicates.
2. **Amazon-Google Products (AG)** [36]: This benchmark contains products based on software and computer hardware from Amazon and Google. The ER linkage must happen between the platforms. The dataset is larger and more challenging, consisting of 11,460 pairs, of which 1,166 are positives. There are only three attributes, one of which is a detailed ‘description’ attribute that is text-heavy.

For quantifying ER performance, we use standard metrics such as precision, recall, and F-Measure (FM), also called the F1-Score. The precision is the ratio of true positives returned by the ER model to all positives predicted by the model, while the recall is the ratio of true positives returned by the ER model to all positives in the ground-truth. The F-Measure is the harmonic mean of the two quantities, and represents a single-point estimate of a method’s ER performance. For the sake of completeness, we also provide formulae for these below:

$$Precision = \frac{|TP|}{|TP| + |FP|} \quad (1)$$

$$Recall = \frac{|TP|}{|TP| + |FN|} \quad (2)$$

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

Here, TP, FP, and FN stand for the sets of true positives, false positives, and false negatives returned by the model, respectively. Cost estimates for a complete experiment (i.e., per-benchmark and per-prompting method) are directly obtained from OpenAI’s invoices and are also reported.

5.1 Results and Discussion

Table 2 shows the result of each prompt pattern across both benchmarks. Overall, the methods generally perform well, with most methods achieving more than 80% F-Measure even on the (relatively difficult) Amazon-Google benchmark. As expected, there is a significant *relative* difference in costs (in percentage terms) due to the considerably lower number of per-prompt tokens used by *no-persona* and *single-attr* compared to (for example) *multi-attr* and *few-shot*. Below, we discuss some specific findings derived from the experimental results.

Prompt Pattern	WDC				Amazon-Google			
	Precision	Recall	FM	Cost	Precision	Recall	FM	Cost
multi-attr	0.92	0.90	0.91	\$0.93	0.97	0.79	0.87	\$3.04
single-attr	0.91	0.94	0.93	\$0.59	0.96	0.70	0.81	\$2.19
multi-json	0.96	0.70	0.81	\$0.99	0.98	0.53	0.69	\$3.23
few-shot	0.94	0.87	0.90	\$1.36	0.97	0.79	0.87	\$3.75
multi-sim	0.78	0.66	0.71	\$0.95	0.93	0.97	0.95	\$3.11
no-persona	0.97	0.85	0.91	\$0.68	0.97	0.56	0.71	\$2.01

Table 2 Evaluation results and cost summary for the six different prompt engineering methods (Section 2) for both benchmarks (WDC and Amazon-Google).

LLMs’ performance declined when using JSON-formatted prompts: Table 2 shows that the *multi-json* pattern under-performed compared to both *multi-attr* and *multi-sim*, despite containing identical pair information. For example, altering the format from *multi-attr* to *multi-json* led to a drop in FM from 0.91 to 0.81 in the WDC dataset and from 0.87 to 0.69 in the Amazon-Google dataset. While the JSON format provides structured data, it does not inherently enhance LLMs’ performance in ER tasks. The decrease in FM across datasets indicates that simpler, less structured formats like *multi-attr* and *multi-sim* are more effective for LLMs in ER.

LLM-generated similarity scores create uncertainty: The performance of *multi-sim* shows that the similarity score generated from GPT-3.5 can create uncertainty in its results, even when selecting the optimal threshold value for maximizing FM. On the WDC dataset, the performance dropped from 0.91 to 0.71. However, the FM increased from the original method of 0.87 to 0.95 for the Amazon-Google Dataset. Therefore, this method shows variability but may be valuable in some contexts. Its cost was similar to *multi-attr* (albeit the slight cost difference could become more magnified if the experiments are conducted on a large scale).

The *single-attr* prompt pattern is simple and viable prompt for performing cost-efficient ER if its underlying assumptions hold: One of the goals behind this study was to determine the most cost-efficient method for ER, while controlling for relative gain in performance. We consider *single-attr* and *no-persona* as potential candidates, which are cheaper than other prompt patterns. We consider *multi-attr* as a comparison-baseline due to its comprehensive attribute coverage, which tends to incur higher running costs. This setup allows us to directly assess the impact of attributes in the datasets and persona integration on ER efficiency, as *multi-attr* can be simplified to *single-attr* by removing all but the *product title* attribute, and to *no-persona* by omitting the *persona* component. First, we compare the ER performance of *multi-attr* with *no-persona*, and found that removing persona negatively affects performance on a more difficult dataset such as Amazon-Google, where the FM decreased from 0.87 to 0.71. Additionally, the recall decreased to the lowest among all patterns, from 0.79 to 0.56. On the other hand, using *single-attr* did not affect the performance as much. For the Amazon-Google dataset, the FM decreased from 0.87 to 0.81, while on the WDC dataset, *single-attr* outperformed *multi-attr*. This shows that *single-attr*, despite its seeming simplicity, is a viable option for cost-efficient ER since

	multi-attr	single-attr	multi-json	few-shot	multi-sim
no-persona (A-G)	0.000	0.000	0.019	0.000	0.000
no-persona (WDC)	0.000	0.000	0.068	0.000	0.000

Table 3 A partial p-value matrix generated using paired Student’s t-tests across the ER results between *no-persona* versus all other five prompt patterns.

	multi-attr	single-attr	multi-json	few-shot	multi-sim	no-persona
multi-attr	0.00/0.00	0.19/0.01	0.30/0.00	0.16/0.00	0.00/0.01	0.25/0.01
single-attr	0.10/0.01	0.00/0.00	0.27/0.00	0.15/0.00	0.01/0.01	0.26/0.00
multi-json	0.04/0.00	0.10/0.00	0.00/0.00	0.11/0.00	0.00/0.00	0.15/0.00
few-shot	0.17/0.00	0.24/0.00	0.37/0.00	0.00/0.00	0.01/0.00	0.33/0.00
multi-sim	0.20/0.01	0.29/0.01	0.46/0.00	0.21/0.00	0.00/0.00	0.00/0.01
no-persona	0.02/0.01	0.12/0.00	0.18/0.00	0.10/0.00	0.00/0.01	0.00/0.00

Table 4 Duplicate pairs / non-duplicate pairs in the ground-truth (Amazon-Google dataset) that the method in the row labeled correctly, but the method in the column labeled incorrectly, expressed as a fraction of the total number of duplicate and non-duplicate pairs in the ground-truth, respectively.

	multi-attr	single-attr	multi-json	few-shot	multi-sim	no-persona
multi-attr	0.00/0.00	0.02/0.02	0.22/0.01	0.07/0.02	0.04/0.03	0.07/0.01
single-attr	0.05/0.02	0.00/0.00	0.26/0.01	0.10/0.02	0.06/0.03	0.10/0.01
multi-json	0.01/0.01	0.02/0.01	0.00/0.00	0.04/0.01	0.02/0.01	0.03/0.01
few-shot	0.02/0.02	0.03/0.02	0.21/0.01	0.00/0.00	0.04/0.02	0.07/0.01
multi-sim	0.05/0.03	0.05/0.03	0.26/0.01	0.10/0.02	0.00/0.00	0.11/0.01
no-persona	0.01/0.01	0.01/0.01	0.18/0.01	0.05/0.01	0.03/0.01	0.00/0.00

Table 5 Duplicate pairs / non-duplicate pairs in the ground-truth (WDC dataset) that the method in the row labeled correctly, but the method in the column labeled incorrectly, expressed as a fraction of the total number of duplicate and non-duplicate pairs in the ground-truth, respectively.

no substantial difference is found (compared to *multi-attr*) across the performance metrics; nevertheless, the cost when using a single attribute is considerably reduced (by about 37%). We note, however, an important limitation here: *single-attr* ultimately only works under the assumption that there is a single attribute that contains the information density for helping the LLM make the distinction between a match and a non-match; furthermore, it also assumes that this attribute is fixed across the dataset and that the domain expert has knowledge of it. In some situations, this assumption may prove restrictive and the advantages of *multi-attr* may emerge more clearly.

Differences in ER results using *no-persona* as a baseline: Because many LLM prompting engineering papers rely on the persona as a means of improving the performance of the LLM, we also analyzed the *no-persona* method by reporting the results of each of the five methods against the *no-persona* method. Table 3 summarizes the p-value results for both datasets; for Amazon-Google, we found that the difference

between *no-persona* and four out of the five prompt patterns (except *multi-json*) was significant at the 99.9% confidence level or above. The *multi-json* method was slightly less significant ($p = 0.019$). For WDC, the difference between *no-persona* and four other prompt methods (except *few-shot*), was significant at the 99.9% confidence level or above. The *few-shot* method was moderately significant against no-persona, with a p-value of 0.068. Even when the overall performance across prompting patterns is consistent, results from the paired t-test suggest that prompting patterns do affect the ER prediction results.

Given the statistical differences noted earlier, we report the analog of a ‘confusion matrix’ in Table 4 for the Amazon-Google dataset and Table 5 for the WDC dataset to better quantify disagreement between these methods. In both datasets, we found that the methods disagreed significantly less on the *non-duplicates*, but that disagreement can be non-trivial and non-zero for duplicates. For instance, on duplicate pairs in the ground-truth that *multi-sim* classified correctly, the *multi-json* method (which used roughly the same information as *multi-sim* in the prompt, but with more structure), classified 46% of those pairs incorrectly (as non-duplicates) in Amazon-Google dataset and 26% in WDC dataset. This suggests that consistency, while high overall, is not always guaranteed between different prompting methods. In the extreme case, we found that there were pairs where all methods agreed (and were incorrect), and cases where the methods were equally divided (obviously, with only some correct). Because each method also prompted the LLM for a reason for its output in its instructions, it is feasible to do a qualitative analysis in the near future of both (inter-prompt) disagreement, and of the LLM’s rationale when it was consistently wrong, using the data released with this study.

This confusion matrix supports the claim made earlier that *single-attr* is a better option to perform cost-efficient ER compared to *no-persona*. In the Amazon-Google dataset, as suggested by Table 4, *single-attr* (on average) provides more correct ER results on pairs that other patterns get wrong, compared to *no-persona*. Additionally, when comparing the row values of *multi-attr* for the *single-attr* column (row 1, column 2) and *no-persona* (row 1, column 6), the values are 0.19 and 0.25, respectively. This shows that out of all candidate pairs that *multi-attr* provided correct decisions on, *single-attr* made fewer mistakes when compared to *no-persona*. We observe similar trends in Table 5 for the WDC dataset, suggesting the findings are stable and not just a measure of random fluctuation.

We end with a comment on why the costs listed in Table 2 are not as trivial as they might seem. In practice, the records in the dataset would not be already paired (as was assumed for purposes of prompting the LLM) but would instead be represented simply as a set of n records. For example, considering the Amazon-Google dataset, the Amazon partition has 1,363 records and the Google partition has 3,226 records. Without the blocking step (see Section 1), the total number of pairs to be evaluated would be more than 3 million (compared to fewer than 12,000 that we evaluated herein). As noted earlier, because the scope of this paper is limited to similarity, not blocking, we assumed perfect blocking and used a set of duplicate and non-duplicate pairs present in the ground-truth that the different prompting methods were evaluated on, similar to an ordinary binary classification problem. Even if blocking yielded 95%

reduction in the number of pairs, we would still need to prompt the LLM with almost 150,000 pairs, leading to 10-15x more expense than shown in Table 2, even for this relatively small dataset.

6 Qualitative Analyses

In the experimental results described earlier, we gained insights into the overall ER performance and cost of each prompt pattern through quantitative and statistical analyses. In this section, we conduct selective *qualitative* analysis of specific candidate pairs to understand the differences and similarities between the prompt patterns, as well as draw some broader conclusions about GPT-3.5 by focusing on both success and failure cases.

We begin by discussing some of GPT-3.5’s success cases on the ER benchmarks, by considering the candidate pairs where GPT-3.5 successfully outputs true positives (TP) and true negatives (TN) for *all* six prompt patterns. Figure 9 shows a candidate pair consisting of (1) AMD Ryzen Threadripper 1950X Box@es and (2) AMD Ryzen Threadripper Sixteen Core 1950X 4.00GHz (Socket TR4) Processor, where both entries refer to the same real-world product but with slight differences in naming. Arguably, this is an ‘easy’ case, and we find (expectedly) that the *single-attr* response demonstrates GPT-3.5’s ability to handle such pairs, showing that slight variations in phrasing and formatting do not negatively affect the ER result. Similar responses are observed using the *multi-attr* and *multi-sim* patterns, in which the pairs include additional details from the description, manufacturer, and price attributes, yet GPT-3.5 still accurately decides the correct ER result. All other prompt methods showed the same outcome and explanation, so we omitted the responses due to space limitations. This result shows that GPT-3.5 does not get confused easily on pairs that are relatively obvious (even when presented with extraneous information), which marks an advance from methods like string matching, which would presumably get more confused due to inclusion of irrelevant attributes and information, at least for the purposes of determining whether the pairs are matching or not.

Similar observations hold for ‘easy’ non-duplicate pairs. Figure 10 shows such a pair consisting of (1) Gigabyte Intel Z370M D3H Coffee Lake Micro ATX Mother-board@en Motherboard and (2) Gigabyte B250M D3H Kaby Lake Micro ATX Mother-board@en Motherboard LN77113 where a clear distinction is in their product number (Z370M vs B250M). The *single-attr* prompt pattern is again found to be able to differentiate between these products based on their titles, pinpointing the chipset variants Z370M and B250M and elaborating that *Coffee Lake* and *Kaby Lake* refer to distinct processor designs. Furthermore, responses using the *multi-attr* and *multi-sim* prompt patterns demonstrate GPT-3.5’s capability to incorporate record descriptions into the decision-making and further contextualize the explanation by comparing additional features like memory types, connectivity options, and USB ports between the two motherboards.

Next, we discuss cases and examples of candidate pairs that GPT-3.5 found more challenging. As shown in the example in Figure 11, GPT-3.5 may provide inaccurate responses when processing pairs that include complex technical terminology or jargon. The figure shows a duplicate pair of (1) Corsair Vengeance LPX 16GB 2400MHz

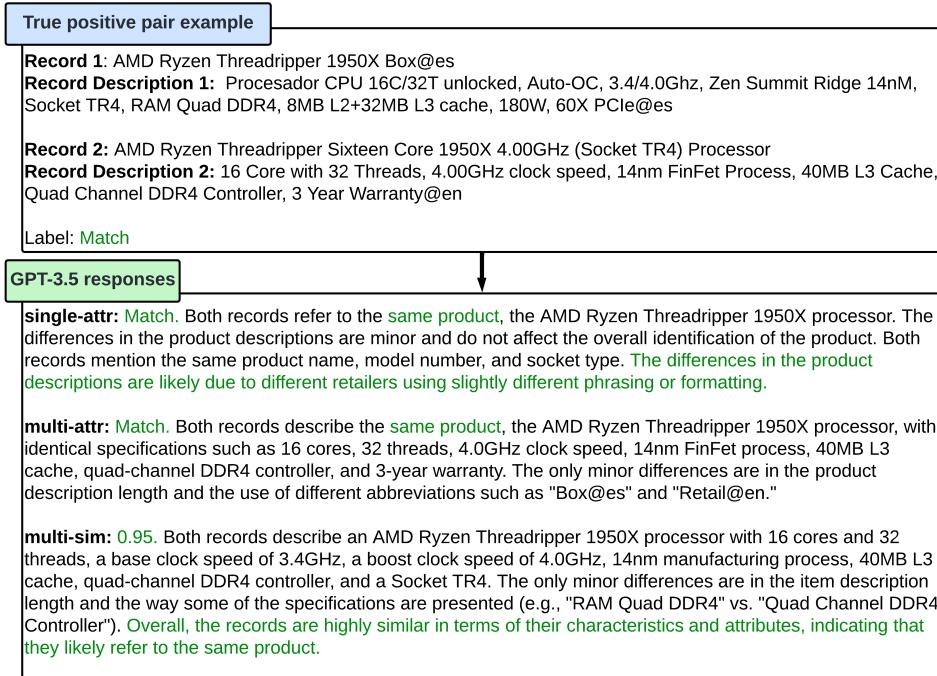


Fig. 9 An illustration of a relatively ‘easy’ duplicate pair, where the two records differ in naming and description order. GPT-3.5 accurately concludes this pair as a duplicate using all prompt patterns, with the correct ER results and explanations highlighted in green.

Quad Channel Kit and (2) Corsair Vengeance LPX 16GB 2400MHz Dual Channel Kit, with details such as full names and descriptions provided in the figure. These two products have differences in their channel kit description (dual vs quad channel), which explains the number of RAM channels tested to operate simultaneously in the product. However, this information does not differentiate the product identity. The response from *single-attr* concludes the correct result and explains that the small difference in dual channel vs quad channel does not affect the identity of the product. However, all other prompt patterns with multiple attributes made incorrect decisions. This is due to two factors: firstly, the description of record 2 contains a tested speed of 2800 MHz that does not exist in the record 1 description. This confused GPT-3.5 with the actual speed that both products have at 2400MHz. Secondly, the difference in CAS timing format between the two records causes GPT-3.5 to misinterpret CL16 as different from 16-16-16-19 CAS timing. These observations highlight how discrepancies in technical specifications and formatting can lead GPT-3.5 to misinterpret critical details, resulting in incorrect ER decisions.

Another challenging candidate pair is shown in Figure 12, representing a non-duplicate pair, where the only difference between the two records is the model number (SDCFXPS-128GB-X46 vs SDCFXPS-128GB). This pair presents a candidate pair that is also challenging for humans, who typically do not memorize details such as model numbers. GPT-3.5 also struggled to provide the correct result, where all prompt

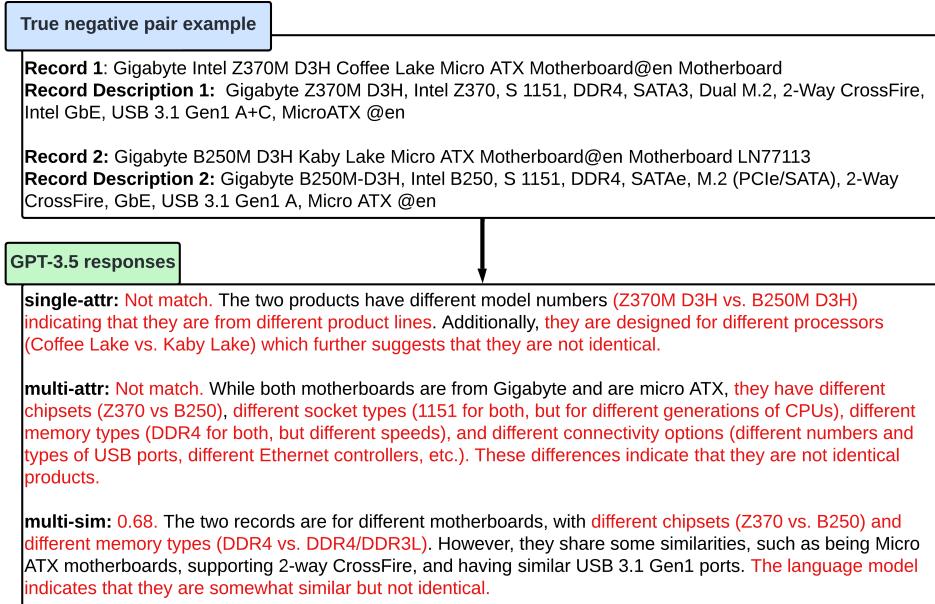


Fig. 10 An illustration of a relatively ‘easy’ non-duplicate pair, where the two records clearly differ in their model numbers. GPT-3.5 accurately concludes this pair as a non-duplicate using all prompt patterns, with the correct ER results and explanation indicated with red highlights.

patterns provided incorrect ER results. The explanations from all patterns hallucinated and mentioned that the model numbers of the two products are the same. This shows that GPT-3.5 still lacks the knowledge to differentiate products at the granular level of model numbers.

In considering both qualitative and quantitative results, therefore, we find an increase in candidate pair attributes doesn’t necessarily translate to better performance for GPT-3.5. The *multi-json* prompt pattern even shows a decrease in performance, likely due to the complexity and structure of JSON affecting GPT-3.5’s processing capabilities. The LLM’s ability to generate similarity scores (e.g., when using *multi-sim*) demonstrates that it can achieve competitive ER performance, even in the absence of predefined similarity metrics. However, it introduced a level of uncertainty, suggesting that GPT-3.5-generated approximations can be ambiguous and not always reliable. The *single-attr* prompt pattern, with its simplicity and relative cost-efficiency, emerged as a viable and cost-efficient option for ER under the assumption that we know a single attribute that contains the most information density for enabling the LLM to make ER decisions.

Qualitatively, GPT-3.5 demonstrated consistent performance in processing pairs that are either clearly identical or distinct, indicating robustness in handling straightforward ER tasks. Although we provided only two detailed examples of such pairs, such results are found to be common across the full dataset. We refer readers interested in the comprehensive results to the supplementary materials included with this paper. In contrast, the model struggled with intricate details such as product model

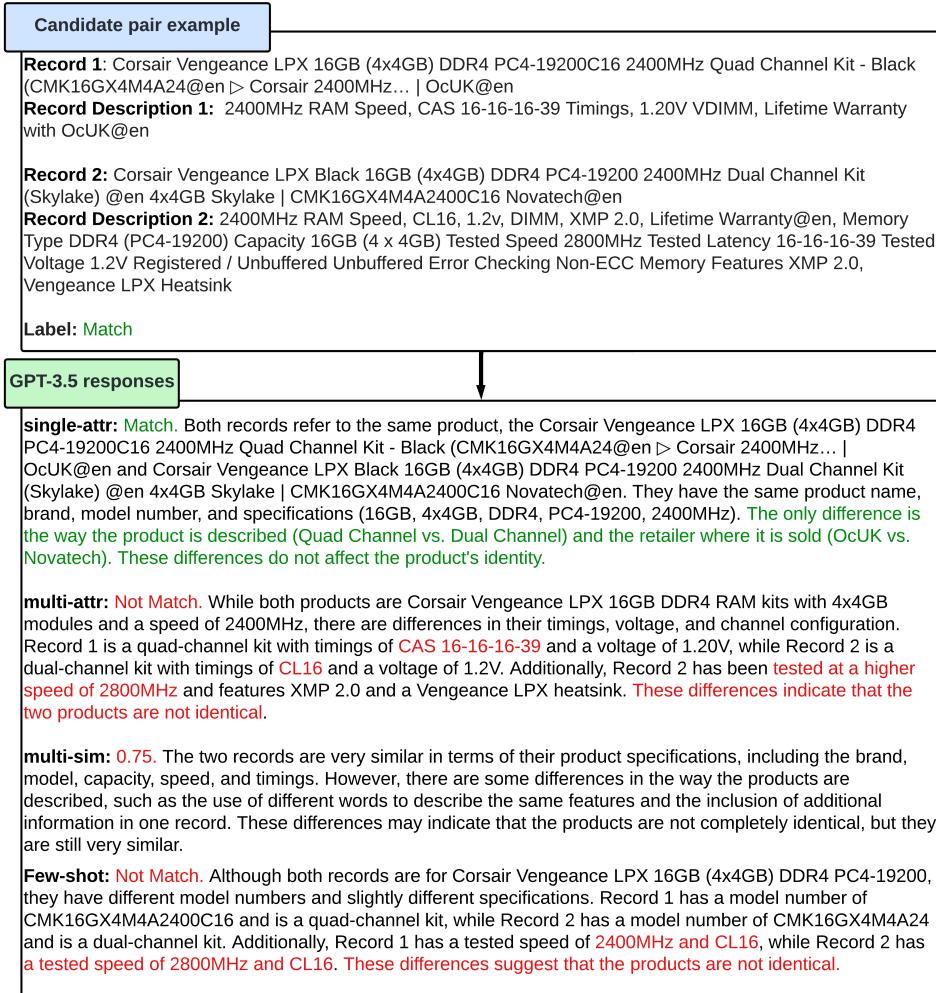


Fig. 11 An illustrative example of a duplicate pair with disagreement on ER results from different prompt patterns. This candidate pair is more ‘difficult’, containing complex technical terminology in the title and description. Of the six patterns, only one correctly identifies the pair as duplicates, showing the challenges GPT-3.5 faces when processing domain-specific lingo. We highlight the wrong explanations from GPT-3.5 in red.

numbers, testing methods, and specifications. This suggests a gap in the LLM’s deep knowledge or understanding when processing pairs that are highly correlated but have slight differences in their detail. Furthermore, excessive information can sometimes lead to confusion, indicating that an overload of data can negatively direct GPT-3.5 to make incorrect ER decisions. These insights collectively highlight the nuanced balance between the quantity of information provided and the quality of ER outcomes, and may point toward more optimized prompting strategies for LLMs in future ER applications.

Candidate pair example

Record 1: Sandisk Extreme Pro Compact Flash CF Memory Card 160MB/s UDMA 7 - SKU
SDCFXPS-128G-X46 - 128GB@en

Record Description 1: video computer photo cinema quality Full HD capture We only supply original Sandisk products. Supplied brand new in manufacturers sealed retail packaging. Card's innovative memory controller offers SanDisk's best performanceUp to 160MB/s write speed for faster burst-mode shootingUDMA interface high data transfer between the card and cameraVPG flawless vape audio products etc. Please browse our store discover more...@en ideal professional broadcasting. Manufacturers Limited Lifetime Warranty.We sell a large very competitive range of also cases inkjet batteries

Record 2: SanDisk **SDCFXPS-128G** DFC 128GB 50p CF 1066x 160MB/s CompactFlash Card Sandisk Extreme Pro

Record Description 2: **SDCFXPS-128G** DFC 128GB 50p CF 1066x 160MB/s CompactFlash Card Sandisk Extreme Pro SanDisk

Label: Not Match

GPT-3.5 responses

single-attr: **Match.** Both records refer to the same product, the Sandisk Extreme Pro Compact Flash CF Memory Card 160MB/s UDMA 7 - 128GB. The differences in the product description are minor and do not affect the overall identification of the product.

multi-attr: **Match.** Both records describe the same product, the SanDisk Extreme Pro Compact Flash CF Memory Card 160MB/s UDMA 7 - 128GB. **They have the same model number (SDCFXPS-128G-X46), write speed (up to 160MB/s), and manufacturer warranty (limited lifetime).** The only difference is in the length and wording of the product description.

multi-sim: **0.95.** Both records describe a Sandisk Extreme Pro Compact Flash CF Memory Card with 160MB/s write speed and 128GB capacity. They also mention UDMA 7 interface, VPG, and high-quality performance for professional use. The **SKU and MPN numbers are the same**, and both mention a manufacturer's limited lifetime warranty. The only minor differences are in the wording and additional details provided in each record, such as the mention of "flawless vape audio products" in Record 1 and the specific model number in Record 2. However, these differences do not significantly affect the overall product identification.

Few-shot: **Match.** Both records refer to the same product, the Sandisk Extreme Pro Compact Flash CF Memory Card 160MB/s UDMA 7 - 128GB. The product name, **model number, and specifications are identical in both records.**

Fig. 12 An illustrative example of a non-duplicate pair with disagreement on ER results from different prompt patterns. This pair contains a minor difference in model numbers, causing GPT-3.5 to make mistakes. The responses across all prompt patterns hallucinated and provided incorrect explanations (highlighted in red), pointing out (incorrectly) that the two products have the same model numbers.

7 Conclusion

This experimental study sought to evaluate whether a recent LLM like GPT-3.5 could offer a promising, cost-efficient and unsupervised alternative to conventional ER similarity functions, which generally require training data (and computation time) and domain-specific feature engineering. In addition to comparing the costs and performance of these methods, we also quantified the extent to which these models disagreed. Our results showed that, while the prompting method does matter, the results are generally stable and consistent (but that disagreement can be high between some prompting methods, particularly on duplicates). Furthermore, there can be significant variance in cost, which becomes important at the scales for which industrial ER is typically designed. While integrating blocking into an LLM-based ER problem remains

an open problem at present, we hope that this study can provide guidance on efficiently using LLMs for unsupervised ER similarity. To support further analysis and replication, we have collected and will release all primary data underlying our analyses as supplementary materials.

Data Availability

All raw data and analyses cited and used in the main text may be accessed here: https://drive.google.com/drive/folders/18taqVQ8oJeNunMb6nz_qZNZ1EnYy7JCF?fbclid=IwAR0jLxyoZQgUbI7wtcL-mbIM6YWdH0L8QYeg99ZKl2xP3FiA4pmrt4Op99Q.

References

- [1] Mehdi Akbarian Rastaghi, Ehsan Kamaloo, and Davood Rafiei. Probing the robustness of pre-trained language models for entity matching. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM ’22, page 3786–3790, New York, NY, USA, 2022. Association for Computing Machinery.
- [2] Beatriz Botero Arcila. Is it a platform? is it a search engine? it’s chatgpt! the european liability regime for large language models. *J. Free Speech L.*, 3:455, 2023.
- [3] Janani Balaji, Faizan Javed, Mayank Kejriwal, Chris Min, Sam Sander, and Ozgur Ozturk. An ensemble blocking scheme for entity resolution of large and sparse datasets. *arXiv preprint arXiv:1609.06265*, 2016.
- [4] Indrajit Bhattacharya and Lise Getoor. Entity resolution in graphs. *Mining graph data*, 311, 2006.
- [5] Zhen Bi, Jing Chen, Yinuo Jiang, Feiyu Xiong, Wei Guo, Huajun Chen, and Ningyu Zhang. Codekgc: Code language model for generative knowledge graph construction. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 23(3):1–16, 2024.
- [6] Mikhail Bilenko and Raymond J Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 39–48, 2003.
- [7] Kiran Busch, Alexander Rochlitzer, Diana Sola, and Henrik Leopold. Just tell me: Prompt engineering in business process management. In *International Conference on Business Process Modeling, Development and Support*, pages 3–11. Springer, 2023.
- [8] Yuyan Chen, Qiang Fu, Yichen Yuan, Zhihao Wen, Ge Fan, Dayiheng Liu, Dongmei Zhang, Zhixu Li, and Yanghua Xiao. Hallucination detection: Robustly discerning reliable answers in large language models. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 245–255, 2023.
- [9] P Christen. Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection. 2012.

- [10] Peter Christen. Febrl: a freely available record linkage system with a graphical user interfac. In *the second Australasian workshop on Health data and knowledge management*, volume 80, pages 14–25, 2008.
- [11] Vassilis Christophides, Vasilis Efthymiou, Themis Palpanas, George Papadakis, and Kostas Stefanidis. An overview of end-to-end entity resolution for big data. *ACM Computing Surveys (CSUR)*, 53(6):1–42, 2020.
- [12] Vassilis Christophides, Vasilis Efthymiou, and Kostas Stefanidis. *Entity resolution in the web of data*, volume 5. Springer, 2015.
- [13] William W Cohen, Pradeep Ravikumar, Stephen E Fienberg, et al. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, volume 3, pages 73–78, 2003.
- [14] William W Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 475–480, 2002.
- [15] Vincenzo Di Cicco, Donatella Firmani, Nick Koudas, Paolo Merialdo, and Divesh Srivastava. Interpreting deep learning models for entity resolution: an experience report using lime. In *Proceedings of the Second International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, pages 1–4, 2019.
- [16] Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. Deeper-deep entity resolution. *arXiv preprint arXiv:1710.00597*, 2017.
- [17] Sabit Ekin. Prompt engineering for chatgpt: a quick guide to techniques, tips, and best practices. *Authorea Preprints*, 2023.
- [18] Ahmed K Elmagarmid, Panagiotis G Ipeirotis, and Vassilios S Verykios. Duplicate record detection: A survey. *IEEE Transactions on knowledge and data engineering*, 19(1):1–16, 2006.
- [19] Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. Reasoning about record matching rules. *Proceedings of the VLDB Endowment (PVLDB)*, 2(1):407–418, 2009.
- [20] Sachin Farfade, Sachin Vernekar, Vineet Chaoji, and Rajdeep Mukherjee. Scaling use-case based shopping using llms. 2024.
- [21] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*, 2023.
- [22] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.
- [23] Lise Getoor and Ashwin Machanavajjhala. Entity resolution: theory, practice & open challenges. *Proceedings of the VLDB Endowment*, 5(12):2018–2019, 2012.
- [24] Lise Getoor and Ashwin Machanavajjhala. Entity resolution for big data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1527–1527, 2013.
- [25] Mozhdeh Gheini and Mayank Kejriwal. Unsupervised product entity resolution using graph representation learning. In *eCOM@ SIGIR*, 2019.

- [26] Louie Giray. Prompt engineering with chatgpt: a guide for academic writers. *Annals of biomedical engineering*, 51(12):2629–2633, 2023.
- [27] Ram Deepak Gottapu, Cihan Dagli, and Bharami Ali. Entity resolution using convolutional neural network. *Procedia Computer Science*, 95:153–158, 2016.
- [28] Tanya Gupta and Varad Deshpande. Entity resolution for maintaining electronic medical record using oyster. In *EAI International Conference on Big Data Innovation for Sustainable Cognitive Computing: BDCC 2018*, pages 41–50. Springer, 2020.
- [29] Yan Hu, Qingyu Chen, Jingcheng Du, Xueqing Peng, Vipina Kuttichi Keloth, Xu Zuo, Yujia Zhou, Zehan Li, Xiaoqian Jiang, Zhiyong Lu, et al. Improving large language models for clinical named entity recognition via prompt engineering. *Journal of the American Medical Informatics Association*, page ocad259, 2024.
- [30] Marzena Karpinska and Mohit Iyyer. Large language models effectively leverage document-level context for literary translation, but critical errors persist. *arXiv preprint arXiv:2304.03245*, 2023.
- [31] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. Low-resource deep entity resolution with transfer and active learning. *arXiv preprint arXiv:1906.08042*, 2019.
- [32] Mayank Kejriwal. *Populating a linked data entity name system: A big data solution to unsupervised instance matching*, volume 27. IOS Press, 2016.
- [33] Mayank Kejriwal. Named entity resolution in personal knowledge graphs. *arXiv preprint arXiv:2307.12173*, 2023.
- [34] Mayank Kejriwal and Daniel P Miranker. An unsupervised algorithm for learning blocking schemes. In *2013 IEEE 13th International Conference on Data Mining*, pages 340–349. IEEE, 2013.
- [35] Nihel Kooli, Robin Allesiardo, and Erwan Pigneul. Deep learning based approach for entity resolution in databases. In *Asian conference on intelligent information and database systems*, pages 3–12. Springer, 2018.
- [36] Hanna Köpcke, Andreas Thor, and Erhard Rahm. Evaluation of entity resolution approaches on real-world match problems, 09 2010.
- [37] Bo-Han Li, Yi Liu, An-Man Zhang, Wen-Huan Wang, and Shuo Wan. A survey on blocking technology of entity resolution. *Journal of Computer Science and Technology*, 35:769–793, 2020.
- [38] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. Deep entity matching with pre-trained language models. *arXiv preprint arXiv:2004.00584*, 2020.
- [39] Robert L Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv preprint arXiv:2106.13353*, 2021.
- [40] Timothy R. McIntosh, Tong Liu, Teo Susnjak, Paul Watters, Alex Ng, and Malka N. Halgamuge. A culturally sensitive test to evaluate nuanced gpt hallucination. *IEEE Transactions on Artificial Intelligence*, pages 1–13, 2023.
- [41] Bertalan Meskó. Prompt engineering as an important emerging skill for medical professionals: tutorial. *Journal of Medical Internet Research*, 25:e50638, 2023.
- [42] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park,

- Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 international conference on management of data*, pages 19–34, 2018.
- [43] Youcef Nafa, Qun Chen, Zhaoqiang Chen, Xingyu Lu, Haiyang He, Tianyi Duan, and Zhanhuai Li. Active deep learning on entity resolution by risk sampling. *Knowledge-Based Systems*, 236:107729, 2022.
 - [44] Hao Nie, Xianpei Han, Ben He, Le Sun, Bo Chen, Wei Zhang, Suhui Wu, and Hao Kong. Deep sequence-to-sequence entity matching for heterogeneous entity resolution. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 629–638, 2019.
 - [45] OpenAI. Gpt-4 technical report, 2024.
 - [46] Kevin O'Hare, Anna Jurek-Loughrey, and Cassio de Campos. A review of unsupervised and semi-supervised blocking methods for record linkage. *Linking and Mining Heterogeneous and Multi-view Data*, pages 79–105, 2019.
 - [47] George Papadakis, Ekaterini Ioannou, Emanoil Thanos, and Themis Palpanas. *The four generations of entity resolution*. Springer, 2021.
 - [48] George Papadakis, George Mandilaras, Luca Gagliardelli, Giovanni Simonini, Emmanoil Thanos, George Giannakopoulos, Sonia Bergamaschi, Themis Palpanas, and Manolis Koubarakis. Three-dimensional entity resolution with jedai. *Information Systems*, 93:101565, 2020.
 - [49] George Papadakis, Dimitrios Skoutas, Emmanoil Thanos, and Themis Palpanas. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(2):1–42, 2020.
 - [50] George Papadakis, Dimitrios Skoutas, Emmanoil Thanos, and Themis Palpanas. Blocking and filtering techniques for entity resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(2):1–42, 2020.
 - [51] Ralph Peeters and Christian Bizer. Supervised contrastive learning for product matching. In *Companion Proceedings of the Web Conference 2022, WWW '22*, page 248–251, New York, NY, USA, 2022. Association for Computing Machinery.
 - [52] Ralph Peeters and Christian Bizer. Entity matching using large language models, 2023.
 - [53] Maciej P Polak and Dane Morgan. Extracting accurate materials data from research papers with conversational language models and prompt engineering—example of chatgpt. *arXiv preprint arXiv:2303.05352*, 2023.
 - [54] Anna Primpeli, Ralph Peeters, and Christian Bizer. The wdc training dataset and gold standard for large-scale product matching, 2019.
 - [55] Konstantinos I Roumeliotis, Nikolaos D Tselikas, and Dimitrios K Nasiopoulos. Llms in e-commerce: a comparative analysis of gpt and llama models in product review evaluation. *Natural Language Processing Journal*, 6:100056, 2024.
 - [56] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–278, 2002.
 - [57] Sigurd Schacht, Sudarshan Kamath Barkur, and Carsten Lanquillon. Promptie-information extraction with prompt-engineering and large language models. In *International Conference on Human-Computer Interaction*, pages 507–514.

Springer, 2023.

- [58] Warren Shen, Xin Li, and A Doan. Constraint-based entity matching. In *AAAI*, pages 862–867, 2005.
- [59] Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. What does clip know about a red circle? visual prompt engineering for vlms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11987–11997, 2023.
- [60] Parag Singla and Pedro Domingos. Entity resolution with markov logic. In *Sixth International Conference on Data Mining (ICDM’06)*, pages 572–582. IEEE, 2006.
- [61] Taylor Sorensen, Joshua Robinson, Christopher Michael Rytting, Alexander Glenn Shaw, Kyle Jeffrey Rogers, Alexia Pauline Delorey, Mahmoud Khalil, Nancy Fulda, and David Wingate. An information-theoretic approach to prompt engineering without ground truth labels. *arXiv preprint arXiv:2203.11364*, 2022.
- [62] Sofia Eleni Spatharioti, David M Rothschild, Daniel G Goldstein, and Jake M Hofman. Comparing traditional and llm-based search for consumer choice: A randomized experiment. *arXiv preprint arXiv:2307.03744*, 2023.
- [63] John R Talburt. *Entity resolution and information quality*. Elsevier, 2011.
- [64] Gemini Team. Gemini: A family of highly capable multimodal models, 2023.
- [65] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [66] Damir Vandic, Flavius Frasincar, Uzay Kaymak, and Mark Riezebos. Scalable entity resolution for web product descriptions. *Information Fusion*, 53:103–111, 2020.
- [67] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk-a link discovery framework for the web of data. *Ldow*, 538:53, 2009.
- [68] Jiaqi Wang, Enze Shi, Sigang Yu, Zihao Wu, Chong Ma, Haixing Dai, Qiushi Yang, Yanqing Kang, Jinru Wu, Huawei Hu, et al. Prompt engineering for healthcare: Methodologies and applications. *arXiv preprint arXiv:2304.14670*, 2023.
- [69] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- [70] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [71] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C Schmidt. A prompt pattern catalog to enhance prompt engineering with chatgpt. *arXiv preprint arXiv:2302.11382*, 2023.
- [72] Xi Ye and Greg Durrett. The unreliability of explanations in few-shot prompting for textual reasoning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave,

- K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 30378–30392. Curran Associates, Inc., 2022.
- [73] Chen Zhao and Yeye He. Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *The World Wide Web Conference*, pages 2413–2424, 2019.
- [74] Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*, 2022.