Large Language Models for Data Augmentation in Product Matching

Kateřina Macková and Martin Pilát
Charles University, Faculty of Mathematics and Physics
Malostranské náměstí 25, 118 00 Prague, Czech Republic
Katerina.Mackova@mff.cuni.cz, Martin.Pilat@mff.cuni.cz

Abstract

Large language models (LLMs) are increasingly used for natural language processing tasks in data-scarce domains. In this paper, we study their application to product matching, the task of deciding whether two product descriptions from two different e-shops describe the same product. We use three recent LLMs (Llama 3.2, Gemma 3, and DeepSeek-R1) to enrich product records in product-matching datasets with descriptions generated from the limited available information, and we show that the generated descriptions improve the performance of both LLM-based and traditional machine-learning matchers.

1 Introduction

Large language models (LLMs) have become ubiquitous in natural language processing research and practice. Models such as GPT-4 (OpenAI et al., 2024), Llama 3 (Grattafiori et al., 2024), Gemma 3 (Gemma Team et al., 2025), and DeepSeek-R1 (DeepSeek-AI et al., 2025) have opened new possibilities for applications in domains with limited annotated data, including text summarization (Zhang et al., 2025), information retrieval (Li et al., 2025b), machine translation (Kocmi et al., 2024), and question answering (Yue, 2025). At the same time, these models are known to suffer from well-documented limitations such as hallucinations (Huang et al., 2025), limited controllability (Li et al., 2025a), and lower helpfulness for non-English languages (Kocmi et al., 2025).

Product matching is a special case of the entity resolution (Karapiperis et al., 2026) problem, where the task is deciding whether two product listings from different e-shops refer to the same underlying product. It is also sometimes referred to as product mapping. The input is a pair of product records, typically consisting of the product name and a few structured attributes (such as manufacturer, price, or category), sometimes accompanied by short free-text descriptions; the output is a binary match/non-match decision for the pair. Product matching is an essential step in applications such as price-comparison portals, market analysis, or merging and deduplicating product catalogs from multiple sources. The task is difficult because it requires correctly interpreting the meaning of all texts characterizing the products and extracting the most important information from them. While it is easy to decide that two completely different products are non-matching, the greatest practical challenge is to distinguish similarly looking but different products, for example two laptop models with nearly identical names but different processors.

In many real-world settings, the available product information is quite limited; some datasets contain only product names. Using machine learning in these cases is challenging. In this paper, we investigate whether LLMs can be used to generate more detailed product descriptions from the limited available information and whether these generated descriptions improve the performance of product-matching models.

We evaluate three recent open-weight LLMs (Llama 3.2, Gemma 3, and DeepSeek-R1) on six product-matching datasets: ProMapEN, ProMapCZ, and two versions each of Amazon-Google and Amazon-Walmart. We use these LLMs to enrich the product records in these datasets with generated descriptions. Additionally, we compare the results to baselines obtained with earlier LLMs – ChatGPT 3.5 (Liu et al., 2023), Bard (Manyika, 2023), Llama 2 (Touvron et al., 2023b), and Bing Chat (Mehdi, 2023) (now Microsoft Copilot, then based on the GPT-4 model).

The main contributions of the paper are:

1. Evaluation of seven different LLMs with different prompts in the product-matching task.
2. Using LLMs to generate product descriptions

Proceedings of the 19th International Natural Language Generation Conference: Long Papers, pages 753–769
October 17-21, 2026 ©2026 Association for Computational Linguistics

in order to increase the performance of both LLM-based and traditional ML-based matchers.

Additionally, as we compare two different generations of LLMs, we also discuss the improvements made both to the models and to the tools used to work with them.

2 Data and Models

First, we describe the datasets and LLMs used in this work.

2.1 Datasets

ProMap (Macková and Pilát, 2024b) is a collection of product-matching datasets in Czech and English. We use the extended versions (Macková and Pilát, 2024a), ProMapCzExt and ProMapEnExt, which contain roughly 1,500 product pairs each divided into matches, close non-matches, and medium non-matches. Matches are identical products; close and medium non-matches are similar products with the same brand, similar name, and similar price, but medium non-matches allow larger attribute differences. The extended versions of the ProMap datasets add about 500 distant non-matches – pairs of products from different categories. We use only the test set, which contains 400 pairs. Each product record includes its name, short and long descriptions, specifications, and images. We omit the "Ext" suffix and refer to these datasets as ProMapCZ and ProMapEN.

Amazon-Walmart (Das et al., 2016) and Amazon-Google (Köpcke et al., 2010) are more classical datasets for product matching. They contain only matching product pairs, so non-matching pairs have to be generated. We use two versions of each dataset. The first version was introduced together with the ProMap datasets; its non-matches were created by choosing random pairs of non-matching products. We refer to these as Amazon-Google-ProMap (AmGoPm) and Amazon-Walmart-ProMap (AmWaPm). The second version comes from the DeepMatcher paper (Mudgal et al., 2018); its non-matches are pairs of products selected to be more similar to each other, which makes this version harder than the ProMap one. We refer to these as Amazon-Google-DeepMatcher (AmGoDm) and Amazon-Walmart-DeepMatcher (AmWaDm).

Apart from the different strategies used to generate non-matches, the ProMap and DeepMatcher versions also differ in the textual information they provide. The ProMap versions contain a short product description in addition to the product name, manufacturer, price, category, and model number. The DeepMatcher versions contain only the product name, manufacturer, price, category, and model number; no short description is available. This difference is important for our experiments, because for the DeepMatcher versions the generated description is the only additional textual signal besides the product name.

2.2 Models

As baselines we use the older LLMs ChatGPT3.5 (Liu et al., 2023), Bard (Manyika, 2023), Llama2 (Touvron et al., 2023a), and Bing Chat. They lack two features that are important for our main experiments: structured JSON output and a large context window. Further details are given in Appendix A.

For the main experiments we use three modern open-weight LLMs that can be run offline through the Ollama tool[1]: Llama 3.2 (3B, Meta AI, September 2024), Gemma 3 (12B, Google DeepMind, March 2025) (Gemma Team et al., 2025), and DeepSeek-R1 (32B, DeepSeek-AI, January 2025) (DeepSeek-AI et al., 2025). Ollama supports JSON-schema-constrained output, which makes parsing and reproduction easier, and the models accept up to 128k tokens, enabling few-shot-learning prompts. Appendix C gives the exact model variants and runtime setup.

3 Related Work

Language models have been used for the problem of product matching and the more general problem of entity matching recently. Specifically, Peeters and Bizer (2024) evaluate a large range of LLMs in the entity matching problem and conclude that LLM-based matchers should be preferred due to their good zero-shot performance in cases where it is expensive to maintain a good training set. Locally hosted LLMs can also be a privacy-preserving alternative to API-based models. Nananukul et al. (2024) evaluate different prompting techniques, their costs and their effect on product matching performance and show that LLMs can provide a cost-efficient unsupervised alternative to traditional approaches. Steiner et al. (2025) fine tune the LLMs for the product matching task and show

[1] https://ollama.com/

that it leads to significant improvement for smaller LLMs. Zhang et al. (2024) on the other hand create AnyMatch, a small language model specifically focused on entity matching and show that it can provide competitive performance with much lower computation overhead. Narayan et al. (2022) use LLMs for error detection, entity matching, and data imputation. Finally, Alves et al. (2024) experiment with cross-lingual learning where they pre-train the models on English data that are later evaluated on Portuguese data.

While LLMs have been used for many tasks related to product matching and entity matching in general, we are not aware of any works that use them specifically to generate additional information about the products or entities to improve the performance of the matching models.

4 Generating product descriptions

Many of the product matching datasets contain only basic information about the products, such as only the product name, manufacturer and price as in the DeepMatcher versions of the Amazon-Google and Amazon-Walmart datasets (Amazon-Walmart also contains product category and model number). Such limited information makes product matching harder, because the model has little evidence beyond lexical overlap between titles. Different e-shops often use different terminology for the same product – for example, one title may call a device a "laptop" while another calls it a "notebook", or one lists a "LED TV" while another uses "LED television". A matcher that relies on surface similarity may therefore fail to recognize matching products even when the underlying item is identical.

In this paper, we augment the datasets by generating richer product descriptions using LLMs. The key motivation is that LLMs can leverage their parametric knowledge to infer what a product is and what it does from only a short title or a few attributes, similarly to how people interpret e-shop listings. The generated description can also "normalize" heterogeneous terminology into a consistent vocabulary, making it easier for downstream matchers to identify matching products. This generated description contains more details for machine learning models to decide whether two product descriptions refer to the same product or not. Figure 1 illustrates the whole process.

We also generate descriptions for the ProMap datasets, which already contain detailed descriptions. In this case, we test whether the LLM-generated descriptions provide an additional normalization benefit beyond the original texts, particularly for keyword-based matchers such as the ProMap-NN baseline.

In order to generate the descriptions, we use a simple prompt that asks the model to create a detailed description based on a JSON dictionary containing the available product information. We request the output in JSON format, containing the product name, brand, price, and the generated description (the exact wording of the prompt is in Appendix C). We evaluate the quality of a generated description by its effect on downstream product-matching performance: a description is considered better if it leads to higher F1 scores in the matching task (the metrics are defined in Section 5.1). We do not use lexical overlap metrics such as BLEU against existing descriptions because the available product information – which often already includes the original description – is part of the prompt. A model that simply repeats the input text would score highly on BLEU without necessarily producing a more useful description for matching.

5 Experiments

In this section, we first define the evaluation task and metrics, then briefly describe baseline experiments with older LLMs, and finally proceed to describe the experiments with generating the descriptions.

5.1 Evaluation Task and Metrics

All the experiments reported in this paper evaluate the same task: the binary classification of product pairs, i.e., determining whether two product listings from different shops or catalogues refer to the same underlying product. For the datasets that distinguish several categories of non-matching pairs (such as the close, medium, and distant non-matches in the ProMap datasets), all non-matching categories are merged into a single negative class. We calculate precision, recall, and F1 for this binary classification task, treating same-product pairs (matches) as the positive class.

5.2 Baselines

In this section, we describe the initial baseline experiments we ran in product matching with several widely used LLMs. These pilot experiments were performed in July and August 2023, before the

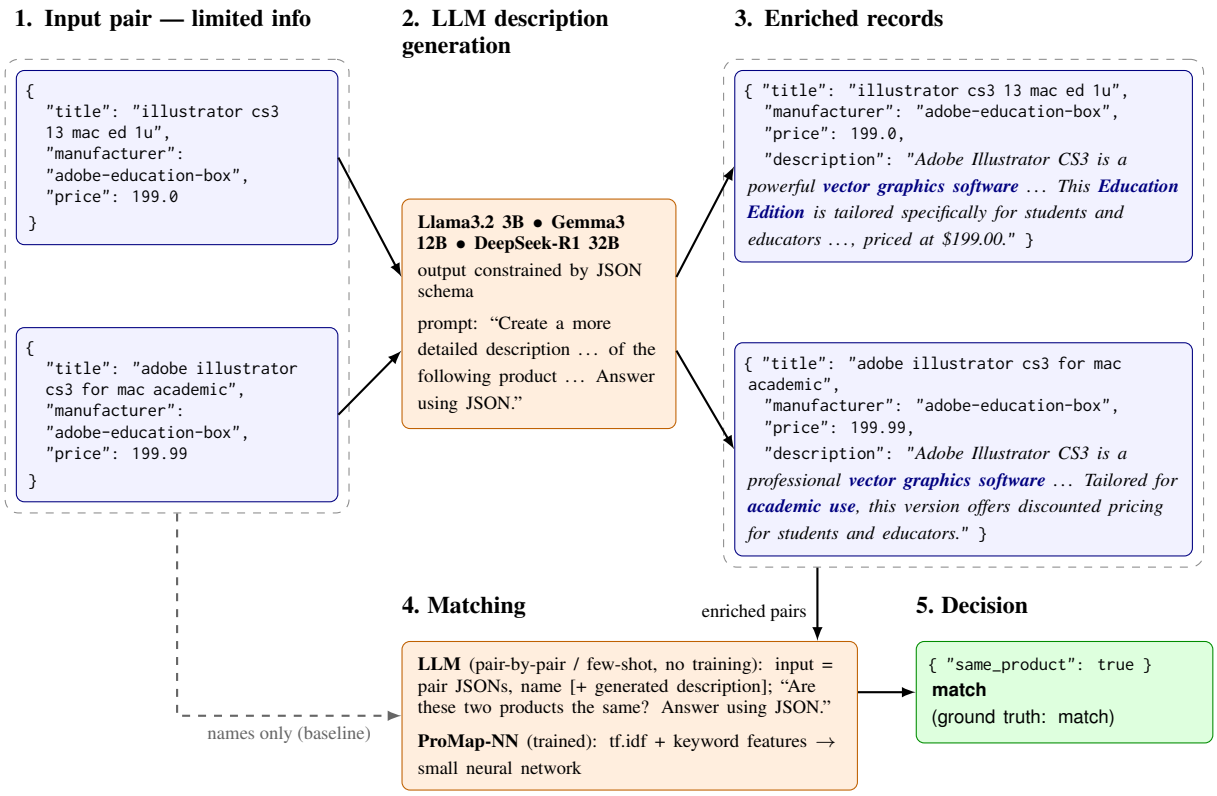


Figure 1: Illustration of the description-enrichment pipeline on one example matching product pair. First (steps 1–3), each product record with only limited information (name, manufacturer, price) is passed to an LLM which generates a detailed product description; the LLM output is constrained by a JSON schema and the generated descriptions (shown shortened) from the two e-shops use a shared terminology (highlighted in bold), which makes the matching decision easier. Then (steps 4–5), a matcher – an LLM prompted with the pair-by-pair or few-shot prompt, or the trained feature-based ProMap-NN model – decides whether the product pair matches. The dashed arrow shows the baseline variant, in which only the original product names are used and no descriptions are generated.

recent generation of open-weight models became available. They provide a realistic point of comparison: in production settings, it is not practical to rebuild the entire pipeline every time a new model is released, so understanding how the older generation performs remains relevant. More details about these experiments, including the exact wording of the prompts and some additional results where the models are combined in a voting ensemble, are presented in the Appendix (Section B).

More specifically, we evaluated two datasets from the extended ProMap Datasets collection – ProMapCz and ProMapEn – on four different LLMs: ChatGPT3.5, Bard, LLaMA2, and Bing. We also evaluated the ProMap version of the Amazon-Walmart and Amazon-Google datasets on ChatGPT and Bing. As most of these models are published without the possibility of being fine-tuned for particular tasks, we directly asked them whether the product pairs match without any training. We used only the test part of the datasets. We ran two types of experiments. In the first one, we used only the names of the products; in the other one, we used names with short descriptions. This was required by the limitations of the models regarding the input size. We evaluated each dataset three times with each model to reach more objective results and to test the consistency of the models. We report the average of the values in the table below. After each evaluation, we created a new conversation with the model to ensure that the results from the previous run do not influence the results of the new one. We tested two types of prompts – Product Matching Championship and Pair-by-Pair approach. In the Product-Matching-Championship prompt, the prompt contains a list of pairs of products and the LLM answers for each of them in batches; in the Pair-by-Pair prompt, the prompt contains only one product pair at a time.

Finally, we also compare the results to a more traditional ML model based on feature extraction and neural networks. We chose the model introduced in

	ProMapCZ			ProMapEN		
model	F1	prec.	recall	F1	prec.	recall
ProMap-NN-N	0.74	0.70	0.79	0.69	0.63	0.77
ProMap-NN-D	0.77	0.78	0.76	0.70	0.61	**0.82**
Bard-D/P	0.74	0.78	0.70	**0.71**	0.68	0.73
Bard-N/P	0.74	0.61	**0.94**	0.69	0.68	0.70
Bing-N/C	0.68	0.79	0.59	0.65	0.71	0.59
Bing-N/P	**0.79**	0.77	0.81	0.61	0.75	0.51
ChatGPT-D/C	0.45	0.54	0.40	0.53	0.49	0.59
ChatGPT-N/C	0.72	0.71	0.74	0.64	0.68	0.60
ChatGPT-N/P	0.70	**0.80**	0.62	0.49	**0.86**	0.35
LLaMA-N/P	0.39	0.34	0.46	0.48	0.44	0.53
	AmWaPm			AmGoPm		
model	F1	prec.	recall	F1	prec.	recall
ProMap-N	**0.92**	0.90	**0.94**	**0.99**	**1.00**	**0.98**
ChatGPT-N/C	0.67	0.95	0.52	0.79	0.99	0.66
ChatGPT-N/P	0.67	**0.98**	0.51	0.66	0.98	0.50
Bing-N/C	0.78	0.97	0.66	0.90	0.98	0.84
Bing-N/P	0.56	0.95	0.39	0.70	0.99	0.54

Table 1: Baseline results for the product-matching task. The table compares the feature-based ProMap-NN model with older LLMs (Bard, Bing Chat, ChatGPT 3.5, and Llama 2) on the ProMapCZ, ProMapEN, AmWaPm, and AmGoPm test sets. Suffix "-N" indicates experiments using only product names, while "-D" uses names and short descriptions; suffixes "/C" and "/P" indicate the Product-Matching-Championship and Pair-by-Pair prompts, respectively.

the papers on ProMap datasets (Macková and Pilát, 2024b,a). The model is based on standard NLP techniques such as tokenization of the text features, lemmatization, and then cosine distance between the tf.idf vectors. Other features based on keyword detection were also added, and their similarity is computed. A three-layer neural network model is trained on these vectors to predict a similarity score. A threshold is then set for the final classification to matching and non-matching pairs. We call this model ProMap-NN in the rest of this paper.

We present the results of these experiments in Table 1 showing the performance of all the models on all the datasets in all the runs. We use the suffix "-N" in the model name to indicate the experiments with names only, and the suffix "-D" to indicate the experiments with names and descriptions. The type of prompt is indicated by the letter after the "/": "/C" for the Product Matching Championship and "/P" for the Pair-by-Pair prompt. Note that some of the models do not work with the Product-Matching-Championship prompt. These are omitted from the results. See the Appendix for more details.

The best results for the Czech ProMapCz dataset were obtained with Bing (F1 score of 0.79 is slightly better than that of the ProMap-NN model). However, the best results for the ProMapEN dataset were obtained by Bard (slightly beating the ProMap-NN models). For the Amazon-Walmart and Amazon-Google datasets, the ProMap-NN model beats the LLMs. In the experiments, we can also see that the Pair-by-Pair prompt often provides better results that the Product-Matching-Championship one. So, in the main experiments in this paper, we focus only on the Pair-by-Pair prompts.

From a more practical point of view, most of these models lack simple interfaces. The experiments with the baselines had to be done manually and the model outputs had to be manually interpreted (these models do not support structured output). This was also the main motivation for using the Product-Matching-Championship prompt that allowed for processing multiple product pairs in batches, thus simplifying the manual process. The modern models have simple APIs that can be easily used and, more importantly, support consistent output in a structured JSON format, which makes working with them much easier.

5.3 Generated Descriptions

Now, we present the main results of the paper – the effect of generating the product descriptions using LLMs. We evaluate the three above-mentioned modern LLMs (Llama3.2, Gemma 3, and DeepSeek-r1) and the ProMap-NN model on six datasets – two versions of Amazon-Google and Amazon-Walmart datasets, and the ProMapEN and ProMapCZ datasets. Because the Llama3.2 model does not support Czech, we do not use it to generate the descriptions for ProMapCZ. We evaluated its outputs in preliminary experiments, and the generated descriptions were very poor – the text is not proper Czech, and it sometimes contains invalid UTF-8 sequences, that complicate the processing.

For the NN-based ProMap model, we used the same pre-processing as in the ProMap papers and trained the model on the respective training sets. We did two versions of the experiments – in one of them, we included only the product name and generated description, in the other one, we included all the available information. We test the version with all information only with the ProMap-NN model, as, especially in the few-shot learning prompt discussed below, the input would be too long for the LLMs.

For the LLM models, we generated prompts based on the test sets of the datasets (no training or fine-tuning is done for the LLMs). We use the prompt-by-prompt technique described in the previous section with two different prompts. One is a slightly modified version of the Pair-by-Pair prompt from the baseline experiments – we only added the instruction to answer using JSON. The product JSON dictionaries contain only the product name, or the product name with the generated description. Additionally, we also experiment with a few-shot-learning prompt that is extended by one positive and one negative example taken randomly from the corresponding training set. The exact wording of both prompts is given in Appendix C.

All the source codes used to generate the prompts and the resulting datasets with the generated descriptions are available as supplementary material[2]. Details about the hardware and runtime are provided in Appendix C.

The results of these experiments are presented in Tables 2, 3, and 4. The dataset name in all the tables consists of two parts – the first one is the name of the original dataset, the second one is the name of the model used to generate the description. DeepSeek-r1 is abbreviated to DS-r1.

Tables 2 and 3 show the results of the tested models on data containing either only product names from the indicated dataset or product names together with a generated description. The former table shows the results of the LLMs with the simple pair-by-pair prompt and of the ProMap-NN model, the latter one shows the results of the LLMs with the few-shot-learning prompt. We can see that in almost all cases enriching the records with the generated description improves the performance. The only exception is the Llama3.2 model that has better performance without the generated description. This is probably caused by the fact that the model is the smallest one of the tested ones with only 3B parameters, making it too simple to fully understand the longer prompts created when the generated description is included. For the ProMap-NN model, the generated descriptions improve the results in all cases except for the ProMapEN dataset. In most cases, the descriptions generated by Gemma3 seem to provide the best performance, although the differences are not very large. For the ProMapEN dataset, the ProMap-NN model is overfitting – the scores on training data (not shown here) with the generated descriptions are better than without them, but the performance on the test data is worse.

For the DeepSeek-r1 and Gemma, the results with the generated descriptions are also almost always better than without them (except for DeepSeek-r1 on AmGoPm and on ProMapEN). The improvement brought by the generated description is much larger than in the case of the ProMap-NN model. Interestingly, it seems that the generated descriptions improve mainly the recall for Gemma3 and mainly the precision for DeepSeek-r1. When comparing the individual LLMs to the ProMap-NN model, we can see that without the generated descriptions, their performance is generally poor, but with the generated descriptions, they are very close to the ProMap-NN model and in some cases even significantly better (e.g. Gemma3 on AmWaDm has F1=0.80, while the ProMap-NN model has only F1=0.50).

Table 3 shows the results of the LLM models on the same datasets, but using the few-shot-learning prompt. We can see that, again, in most cases for DeepSeek-r1 and Gemma3, the results are better with the generated descriptions than without them. For Llama3.2, we again see that the shorter prompts resulting from not using the generated descriptions

[2]Figshare: https://doi.org/10.6084/m9.figshare.29575358

model	DS-r1_32B			Gemma3_12B			Llama3.2_3B			ProMap-NN		
dataset	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.
AmGoDm	0.39	0.31	**0.51**	0.40	0.43	0.38	**0.12**	0.47	**0.07**	0.53	0.44	**0.67**
+ DS-r1_32B	**0.46**	0.41	**0.51**	0.52	0.41	0.71	0.09	**0.61**	0.05	0.53	0.47	0.62
+ Gemma3_12B	0.42	**0.42**	0.43	**0.56**	**0.44**	**0.76**	0.02	0.43	0.01	**0.54**	**0.49**	0.59
+ Llama3.2_3B	0.41	0.39	0.43	0.52	0.43	0.65	0.02	0.27	0.01	**0.54**	0.48	0.62
AmGoPm	**0.60**	0.86	**0.46**	0.62	**1.00**	0.45	**0.12**	**1.00**	**0.06**	**0.98**	0.99	0.97
+ DS-r1_32B	0.55	**1.00**	0.38	**0.84**	**1.00**	**0.73**	0.11	**1.00**	**0.06**	**0.98**	0.98	**0.99**
+ Gemma3_12B	0.59	0.97	0.42	**0.84**	**1.00**	0.72	0.04	**1.00**	0.02	0.97	0.98	0.96
+ Llama3.2_3B	0.54	0.99	0.37	0.82	**1.00**	0.69	0.09	**1.00**	0.05	**0.98**	**1.00**	0.96
AmWaDm	0.39	0.29	0.57	0.53	0.67	0.44	**0.24**	0.74	**0.15**	0.45	0.46	0.44
+ DS-r1_32B	0.48	0.43	0.55	0.68	0.61	0.78	0.17	0.82	0.09	0.49	**0.53**	0.45
+ Gemma3_12B	**0.52**	**0.45**	**0.60**	**0.80**	**0.78**	**0.83**	0.17	0.78	0.09	**0.50**	0.49	**0.51**
+ Llama3.2_3B	0.41	0.38	0.45	0.63	0.59	0.68	0.11	**1.00**	0.06	0.47	0.43	**0.51**
AmWaPm	0.62	0.84	0.49	0.57	**1.00**	0.40	**0.21**	**1.00**	**0.12**	0.90	**0.91**	0.89
+ DS-r1_32B	0.63	0.96	0.47	**0.89**	0.98	**0.81**	0.08	**1.00**	0.04	**0.91**	**0.91**	0.92
+ Gemma3_12B	**0.72**	0.96	**0.57**	0.88	0.99	0.79	0.09	**1.00**	0.05	**0.91**	0.88	**0.94**
+ Llama3.2_3B	0.67	**0.97**	0.52	0.88	0.99	0.79	0.09	**1.00**	0.05	0.90	0.88	0.92
ProMapCZ	0.58	0.59	**0.56**	0.66	0.91	0.52	**0.26**	**0.88**	**0.15**	**0.74**	**0.70**	**0.79**
+ DS-r1_32B	0.53	0.61	0.47	0.65	0.84	0.53	0.21	0.86	0.12	0.71	0.66	0.78
+ Gemma3_12B	**0.59**	**0.64**	0.55	**0.69**	**0.92**	**0.55**	0.13	0.78	0.07	**0.74**	**0.70**	0.78
ProMapEN	**0.64**	0.66	**0.62**	0.74	**0.87**	0.64	**0.22**	0.76	**0.13**	**0.69**	**0.63**	0.77
+ DS-r1_32B	0.59	0.68	0.51	0.74	0.73	0.74	0.18	0.77	0.10	0.64	0.50	**0.89**
+ Gemma3_12B	0.61	0.64	0.57	**0.79**	0.84	**0.75**	0.16	**0.82**	0.09	0.63	0.59	0.68
+ Llama3.2_3B	0.63	**0.74**	0.55	0.74	0.75	0.73	0.11	0.75	0.06	0.65	0.57	0.75

Table 2: Results with the pair-by-pair prompt. Only the names of the products, or names and generated descriptions are used. The model used for generating the descriptions is shown after the "+" in the dataset name. Bold values are the best results for given dataset/model combination, indicating the best generated descriptions.

lead to better performance. We can, however, also see that the few-shot-learning prompt gives much better results than the pair-by-pair one for all the models. The difference is most pronounced for Llama3.2 (although its absolute results are still quite poor compared to the other LLMs). With the generated description and the few-shot-learning prompt, DeepSeek-r1 and Gemma3 beat the simple ProMap-NN model in most cases except for the relatively simple AmGoPm and AmWaPm datasets.

When we compare the quality of the generated descriptions by their downstream effect on product-matching performance, Gemma3 is consistently the best or very close to the best across all experiments, regardless of which model is used for the final classification. Gemma3 also has the best results of the LLMs in the classification task, especially with the pair-by-pair prompt. For the few-shot-learning one, it is tied with DeepSeek-r1. This is quite surprising given Gemma3 is a much smaller model.

Figure 2 shows the critical difference plot (Demšar, 2006) with statistical evaluation of the different model and prompt combinations and of the different models used as product description generators. In the latter case, we removed the results of the Llama3.2 model (when used for classification) as it cannot successfully deal with the longer prompts when using the generated descriptions. We first ran the Friedman test to show that the differences are statistically significant ($p \leq 0.0002$ in both cases), and then we used the Conover test with Bonferroni correction as the post-hoc test. The lines in the plot connect models that are not statistically significantly different at significance level $\alpha = 0.05$. Tables with p-values for all pairwise comparisons are in the Appendix. The results are consistent with what we discussed above – the Gemma3 and DeepSeek-r1 models give the best results for the product matching task (with

model	DS-r1_32B			Gemma3_12B			Llama3.2_3B		
	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.
AmGoDm	0.38	0.24	**0.93**	**0.56**	**0.44**	0.79	**0.29**	0.18	**0.69**
+ DS-r1_32B	0.57	0.43	0.84	0.53	0.39	**0.86**	0.17	0.36	0.11
+ Gemma3_12B	**0.58**	**0.46**	0.76	0.54	0.40	0.85	0.09	**0.39**	0.05
+ Llama3.2_3B	0.53	0.40	0.76	0.52	0.39	0.75	0.16	0.29	0.11
AmGoPm	**0.92**	0.89	**0.94**	**0.93**	**1.00**	0.87	**0.80**	0.95	**0.69**
+ DS-r1_32B	0.90	**1.00**	0.82	**0.93**	**1.00**	0.87	0.17	**1.00**	0.10
+ Gemma3_12B	0.88	**1.00**	0.78	**0.93**	**1.00**	**0.88**	0.25	**1.00**	0.15
+ Llama3.2_3B	0.89	**1.00**	0.79	**0.93**	**1.00**	**0.88**	0.21	**1.00**	0.12
AmWaDm	0.38	0.24	**0.87**	0.54	0.45	0.68	**0.27**	0.18	**0.52**
+ DS-r1_32B	0.60	0.47	0.83	0.60	0.46	0.86	0.11	**0.79**	0.06
+ Gemma3_12B	**0.75**	**0.70**	0.80	**0.70**	**0.58**	**0.88**	0.05	0.50	0.03
+ Llama3.2_3B	0.59	0.47	0.78	0.58	0.45	0.78	0.09	0.60	0.05
AmWaPm	0.89	0.86	**0.92**	0.83	0.98	0.73	**0.68**	0.85	**0.57**
+ DS-r1_32B	**0.91**	**0.98**	0.84	**0.95**	0.98	**0.91**	0.07	**1.00**	0.04
+ Gemma3_12B	0.90	**0.98**	0.82	**0.95**	**0.99**	**0.91**	0.10	0.86	0.05
+ Llama3.2_3B	**0.91**	**0.98**	0.85	0.94	0.98	0.90	0.09	**1.00**	0.05
ProMapCZ	0.67	0.56	**0.85**	0.72	**0.78**	0.66	**0.47**	0.44	**0.51**
+ DS-r1_32B	0.74	0.73	0.76	0.75	0.68	0.85	0.21	0.63	0.12
+ Gemma3_12B	**0.77**	**0.77**	0.77	**0.77**	0.68	**0.90**	0.17	**1.00**	0.09
ProMapEN	0.76	0.67	**0.87**	0.77	**0.75**	0.78	**0.34**	0.76	**0.22**
+ DS-r1_32B	0.77	0.71	0.83	0.75	0.65	0.88	0.11	0.86	0.06
+ Gemma3_12B	**0.78**	**0.81**	0.75	**0.79**	0.70	**0.91**	0.07	0.67	0.04
+ Llama3.2_3B	0.76	0.75	0.77	0.75	0.64	**0.91**	0.06	**1.00**	0.03

Table 3: Results with the few-shot-learning prompt. Only the names of the products, or names and generated descriptions are used. The model used for generating the descriptions is shown after the "+" in the dataset name. Bold values are the best results for given dataset/model combination, indicating the best generated descriptions.

Gemma3 being significantly better than the baseline ProMap-NN model). For generating descriptions, the Gemma3 model is significantly better than all the other ones (including not using any generated descriptions at all).

Additionally, Table 4 shows the results of the ProMap-NN model on the AmWaPm, AmGoPm, and the ProMap datasets when all the available data are used. We did not include the DeepMatcher versions of the AmWa and AmGo datasets here, as the model only uses the product name from these, so the results would be the same as those presented in Table 2. The results show that even in this case, where the datasets already contain some descriptions of the products (only short descriptions in the case of AmGo and AmWa, but relatively long descriptions in the ProMap datasets), the generated descriptions still improve the results slightly in 3 out of the 4 experiments. The only exception is the ProMapEn dataset, where the results are better without the generated description. Judging by the scores on training data, it seems that the model slightly overfits on this data and the problem is more pronounced when the generated descriptions are included.

6 Conclusion

In this paper, we have investigated the performance of large language models on the product matching task and compared them to a model fine-tuned for this specific task. The results show that the best modern LLMs can beat this domain-specific model in most cases, except for the very simple AmGoPm and AmWaPm datasets. This is an interesting result, considering the models were not specifically trained for this task. In most cases, the Gemma3 LLM shows the best performance over most of the datasets.

The newer models generally show improved performance compared to the older baseline ones.

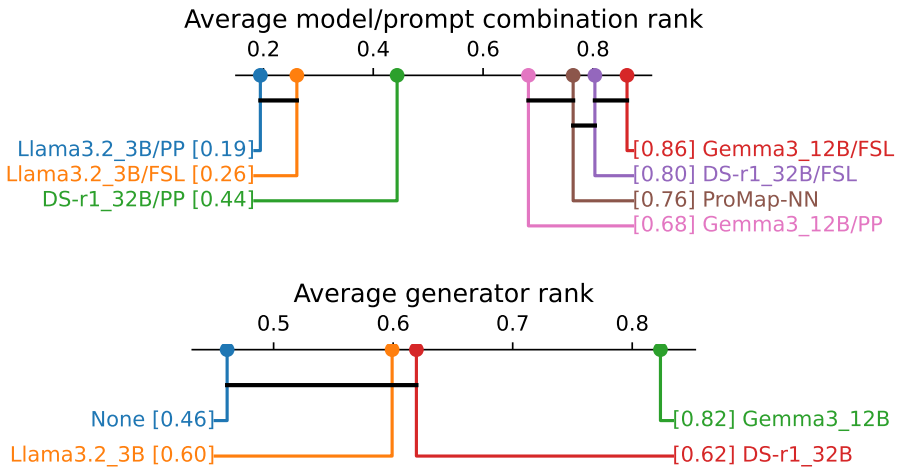


Figure 2: Critical difference plots for different model and prompt combination (top) and for different description generating models (bottom). Average normalized ranks (ordered in ascending order by F1 score – higher is better).

Dataset	F1	Prec.	Rec.
AmGoPm	0.97	0.98	0.95
+ DS-r1_32B	**0.99**	**0.99**	**0.98**
+ Gemma_12B	0.98	0.98	**0.98**
+ Llama3.2_3B	0.98	**0.99**	0.97
AmWaPm	0.90	**0.90**	0.91
+ DS-r1_32B	0.91	0.88	0.93
+ Gemma_12B	**0.92**	0.88	**0.96**
+ Llama3.2_3B	**0.92**	0.89	0.95
ProMapCZ	0.74	0.72	0.77
+ DS-r1_32B	**0.76**	0.71	**0.82**
+ Gemma_12B	0.73	**0.73**	0.73
ProMapEN	**0.68**	**0.66**	0.70
+ DS-r1_32B	0.62	0.57	0.68
+ Gemma_12B	0.63	0.55	**0.74**
+ Llama3.2_3B	0.64	0.64	0.63

Table 4: The effect of generated descriptions on the performance of the feature-based model while using all the data available in the datasets.

While the older models are sensitive to the length of the prompt and typically have worse results when longer descriptions are included, for the newer models, including additional information brings clear advantages. However, the age of the models is not the only important aspect – the size of the models is also important, as is demonstrated by the Llama3.2 model with only 3B parameters. It was unable to get good results in most of the experiments.

We have also demonstrated that LLMs can be successfully used to enrich product records with descriptions generated from the information available in the dataset. The enriched records generally lead to better results.

Regarding the different types of prompts tested, the few-shot-learning one that also contains an example of a matching and a non-matching pair of products significantly improves the performance of the models. This is not that surprising – including examples is a well-known prompting technique. We have, however, also shown that for the smallest model this may have a negative effect.

For the ProMapCZ dataset, the performance of Bing is better than the performance of all the models, including the new ones, however, the difference is not large. We believe this may be caused by the fact that Czech is still relatively under-represented in LLMs, and Bing is a much larger model than those that we used for the experiments with the newer ones. On the other hand, a huge advantage of the newer models is that they can be used offline without the need to pay API fees.

The results presented here for the baseline models were obtained during July and August 2023. The results for the newer models were obtained in June 2025. This shows that the current relatively small open-weight models can beat much larger models from only two years ago. We can expect that further development of the models may lead to even better results. It would also be interesting to evaluate the models on multilingual datasets, where each product in the pair would be in a different language. We are, however, not aware of any dataset that would provide such product pairs.

7 Limitations

The main limitation of this work concerns the reproducibility of the baseline experiments. Most of them were conducted in July and August 2023 using ChatGPT3.5, Bard, Bing Chat, and Llama2. ChatGPT3.5 is no longer publicly available in the version used in our experiments, and the web interfaces of Bard and Bing Chat have evolved substantially since then. Moreover, the experiments with the proprietary models relied on manual interaction through web interfaces rather than on stable APIs. Consequently, the exact model behaviours cannot be reproduced precisely.

Additionally, we do not systematically check the quality of the generated descriptions and evaluate them only by their effect on downstream product-matching performance. While this is a practical and task-relevant evaluation, it does not directly assess the factual accuracy or linguistic quality of the generated descriptions.

Another limitation is the possibility of data contamination, which affects how the results should be interpreted. The proprietary baseline models were accessed without an API, which means that the inputs may have been subject to the providers' terms and conditions and potentially used for further model training. Prior work has documented that closed-source LLMs can be exposed to benchmark data through such indirect leakage, artificially boosting their apparent performance (Balloccu et al., 2024). Although the modern models evaluated in the main experiments are open-weights models run locally, the possibility that they or related proprietary models from the same providers have seen similar product descriptions during training cannot be fully ruled out. Therefore, the strong results reported for the models, especially those from large commercial providers, should be interpreted with caution.

8 Acknowledgements

KM was supported by Center for Foundations of Modern Computer Science (Charles Univ. project UNCE 24/SCI/008) and by SVV project number 260 821. MP was partially supported by PEER project that has received funding from the EU Research and Innovation Programme, under Grant Agreement number 101120406. The paper reflects only the authors' view and the EC is not responsible for any use that may be made of the information it contains. Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

References

Andre Luiz Firmino Alves, Claudio de Souza Baptista, Luciano Barbosa, and Clecio B. M. Araujo. 2024. Cross-lingual learning strategies for improving product matching quality. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*, SAC '24, page 313–320, New York, NY, USA. Association for Computing Machinery.

Simone Balloccu, Patrícia Schmidtová, Mateusz Lango, and Ondrej Dusek. 2024. Leak, cheat, repeat: Data contamination and evaluation malpractices in closed-source LLMs. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 67–93, St. Julian's, Malta. Association for Computational Linguistics.

Sanjib Das, AnHai Doan, Paul Suganthan G. C., Chaitanya Gokhale, Pradap Konda, Yash Govind, and Derek Paulsen. 2016. The magellan data repository. https://sites.google.com/site/anhaidgroup/projects/data.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *Preprint*, arXiv:2501.12948.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30.

Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, Alexis Chevalier, and Julius Berner. 2023. Mathematical capabilities of chatgpt. *arXiv:2301.13867*.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 197 others. 2025. Gemma 3 technical report. *Preprint*, arXiv:2503.19786.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2).

Dimitrios Karapiperis, Christos Tjortjis, and Vassilios Verykios. 2026. A comprehensive survey of deep learning for entity resolution. *ACM Comput. Surv.* Just Accepted.

Tom Kocmi, Sweta Agrawal, Ekaterina Artemova, Eleftherios Avramidis, Eleftheria Briakou, Pinzhen Chen, Marzieh Fadaee, Markus Freitag, Roman Grundkiewicz, Yupeng Hou, Philipp Koehn, Julia Kreutzer, Saab Mansour, Stefano Perrella, Lorenzo Proietti, Parker Riley, Eduardo Sánchez, Patricia Schmidtova, Mariya Shmatova, and Vilém Zouhar. 2025. Findings of the WMT25 multilingual instruction shared task: Persistent hurdles in reasoning, generation, and evaluation. In *Proceedings of the Tenth Conference on Machine Translation*, pages 414–435, Suzhou, China. Association for Computational Linguistics.

Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Marzena Karpinska, Philipp Koehn, Benjamin Marie, Christof Monz, Kenton Murray, Masaaki Nagata, Martin Popel, Maja Popović, and 3 others. 2024. Findings of the WMT24 general machine translation shared task: The LLM era is here but MT is not solved yet. In *Proceedings of the Ninth Conference on Machine Translation*, pages 1–46, Miami, Florida, USA. Association for Computational Linguistics.

Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endow.*, 3(1–2):484–493.

Ming Li, Han Chen, Chenguang Wang, Dang Nguyen, Dianqi Li, and Tianyi Zhou. 2025a. RuleR: Improving LLM controllability by rule-based data recycling. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 926–943, Albuquerque, New Mexico. Association for Computational Linguistics.

Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yuyao Zhang, Peitian Zhang, Yutao Zhu, and Zhicheng Dou. 2025b. From matching to generation: A survey on generative information retrieval. *ACM Trans. Inf. Syst.*, 43(3).

Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. 2023. Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology*, page 100017.

Brady D Lund and Ting Wang. 2023. Chatting about chatgpt: how may ai and gpt impact academia and libraries? *Library Hi Tech News*, 40(3):26–29.

Kateřina Macková and Martin Pilát. 2024a. Extended promap datasets for product mapping. *Electronic Commerce Research*.

Kateřina Macková and Martin Pilát. 2024b. Promap: Product mapping datasets. In *Advances in Information Retrieval*, pages 159–172, Cham. Springer Nature Switzerland.

James Manyika. 2023. An overview of bard: an early experiment with generative ai. Technical report, Technical report, Google AI.

Yusuf Mehdi. 2023. Reinventing search with a new AI-powered Microsoft Bing and Edge, your copilot for the web. Microsoft Blog.

Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, New York, NY, USA. Association for Computing Machinery.

Navapat Nananukul, Khanin Sisaengsuwanchai, and Mayank Kejriwal. 2024. Cost-efficient prompt engineering for unsupervised entity resolution in the product matching domain. *Discover Artificial Intelligence*, 4(1):56.

Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can foundation models wrangle your data? *Proc. VLDB Endow.*, 16(4):738–746.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Ralph Peeters and Christian Bizer. 2024. Entity matching using large language models. *Preprint*, arXiv:2310.11244.

Aaron Steiner, Ralph Peeters, and Christian Bizer. 2025. Fine-tuning large language models for entity matching. In *2025 IEEE 41st International Conference on Data Engineering Workshops (ICDEW)*, pages 9–17.

Nigar M Shafiq Surameery and Mohammed Y Shakor. 2023. Use chat gpt to solve programming bugs. *International Journal of Information Technology & Computer Engineering (IJITC) ISSN: 2455-5290*, 3(01):17–22.

Yiming Tan, Dehai Min, Yu Li, Wenbo Li, Nan Hu, Yongrui Chen, and Guilin Qi. 2023. Evaluation of chatgpt as a question answering system for answering complex questions. *arXiv:2303.07992*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023a. Llama: Open and efficient foundation language models. *arXiv:2302.13971*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023b. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Murong Yue. 2025. A survey of large language model agents for question answering. *Preprint*, arXiv:2503.19213.

Haopeng Zhang, Philip S. Yu, and Jiawei Zhang. 2025. A systematic survey of text summarization: From statistical methods to large language models. *ACM Comput. Surv.*, 57(11).

Zeyu Zhang, Paul Groth, Iacer Calixto, and Sebastian Schelter. 2024. Anymatch – efficient zero-shot entity matching with a small language model. *Preprint*, arXiv:2409.04073.

Supplementary Materials

A Details about older LLMs

ChatGPT (Liu et al., 2023) is an LLM developed by OpenAI that was introduced in November 2022. Its original purpose was to create human-like conversational dialogue and help with simple tasks such as responding to questions and composing various written content. It was soon discovered that it could also be used for many other NLP tasks such as information retrieval, text summarization, question answering, or even simple mathematical computations and code generation (Surameery and Shakor, 2023; Tan et al., 2023; Frieder et al., 2023; Lund and Wang, 2023).

Bard (Manyika, 2023) is an LLM developed by Google that was introduced in March 2023. It can also generate articles, poetry, and essays or solve simple tests and exams. The main difference between ChatGPT and Bard is the data source and its purpose. ChatGPT was developed as a next-generation chatbot, whereas Bard is designed specifically to assist research. Therefore, Bard can browse the Internet and continually get new information. Bard's input is limited to 4k characters, with a similar limit (5k) to its output.

LLaMA (Touvron et al., 2023a) is a family of LLMs developed by Meta AI introduced in February 2023. In this group of models we consider Llama2, specifically in the 7B version. LLaMA's model weights are publicly released to the research community under a non-commercial license and, therefore, this LLM can be used offline.

Bing Chat is developed by Microsoft and was introduced in February 2023 as a part of the Bing search engine. Bing Chat is based on the Microsoft Prometheus model, which has been built on top of the GPT-4 large language model. GPT-4 is a generative AI model created by OpenAI based on transformer architecture and trained on a large corpus of books. It was designed as a chat tool, to write different types of text content such as poems, songs, essays, and reports similar to ChatGPT. Similarly to Bard, it was designed to include browsing the internet and getting real-world data. Additionally, along with information about searched queries, it provides related links to the website containing relevant information and includes images and logos using its Image Creator. As with all the previous language models and chatbots, Bing is multilingual and supports hundreds of languages.

B Detailed Description of the Baseline Experiments

This section contains an extended description of the baseline experiments. These pilot experiments were performed in July and August 2023 with several widely used LLMs, before the recent generation of open-weight models became available. Although some of these models (such as ChatGPT3.5) are no longer publicly available in the exact version we used, the results remain relevant: in production settings, it is not practical to rebuild the entire pipeline every time a new model is released, so comparing the older generation with newer models is a realistic evaluation scenario.

We evaluated two datasets from the extended ProMap Datasets collection – ProMapCzExt and ProMapEnExt – on all four LLMs in this category: ChatGPT, Bard, LLaMA and Bing. We also evaluated the ProMap version of the Amazon-Walmart and Amazon-Google datasets on ChatGPT and Bing. As most of these models are published without the possibility of being fine-tuned for particular tasks, we directly asked them whether the product pairs match without any training. We used only the test part of the datasets. We evaluated their results

by measuring F1, precision, and recall. We ran two types of experiments. In the first one, we used only the names of the products; in the other one, we used names with short descriptions. This was required by the limitations of the models regarding the input size. We evaluated each dataset three times with each model to reach more objective results and to test the consistency of the models. We report the average of the values in the table below. After each evaluation, we created a new conversation with the model to ensure that the results from the previous run do not influence the results of the new one. We tested two types of prompts – Product Matching Championship and Pair-by-Pair approach. In the end, we also used the three predictions for voting to evaluate the scenario in which the models are queried multiple times about each of the products and the most common answer is used.

B.1 Prompts

An important consideration while using LLMs is how the prompts are formulated. We tried a number of different prompts, but the structure did not seem that important as long as the model was able to understand the question. In the end, we formulated two prompts. One of them is formulated as a Product Matching Game:

> Hello, ChatGPT, let's play a game called Product Mapping Championship. The rules are simple. I will give you a list of pairs of products from two e-shops and for each pair, you will try to guess whether the products are the same or not. If they are the same print 1 and if not print 0. I will start with product names. Product pairs are separated by enter and both products in pairs are separated by tabulator. Ready to play? Please while printing the response repeat both product names and 1 whether they match or 0 if not.

For the experiments with the product names only, we then followed with a list of products in the form:

> PRODUCT1: First product name from pair 1
> PRODUCT2: second product name from pair 1
> <empty line>
> PRODUCT1: First product name from pair 2
> PRODUCT2: Second product name from pair 2

For experiments with the product names and short descriptions, we also added the description to the prompt.

This form of the prompt has the advantage that we can get the predictions for several pairs of products at the same time. However, some models were unable to work with this kind of prompt due to some limitations (limited input size, or they did not understand the instructions). So, we included also a second much simpler prompt – a Pair-by-Pair one, and we repeated the same question for each product pair:

> Are these two products the same? Return 1 if yes or 0 if no.
> PRODUCT1: First product name
> PRODUCT2: Second product name

The example is for the experiment with names only, for the experiment with names and descriptions, we again added the description to the prompt.

In all cases, for the experiments on the Czech datasets, the prompts were translated to Czech.

B.2 Model Outputs

These older models do not support any structured output (such as JSON according to a JSON Schema), and while the instructions specify that the models should answer only 0 or 1, most of the time, the models include some additional information, such as a more detailed explanation of the differences. This complicates any automated processing of the answers and, together with the fact that some models do not provide any free API, means that we did most of these experiments manually, using only simple scripts to generate the prompts based on the pairs of products in the datasets. These scripts, together with the labels provided by the models are available as supplementary materials[3].

B.3 Results

The results of the experiments with the older models are presented in Table 5 showing the performance of all the models on all the datasets in all the runs. We use the suffix "-N" in the model name to indicate the experiments with names only, and the suffix "-D" to indicate the experiments with names and descriptions. The type of prompt is indicated by the letter after the "/": "/C" for the Product Matching Championship and "/P" for the Pair-by-Pair prompt.

ChatGPT with the Product-Matching-Championship prompt is very close to the baseline ProMap-NN model when using only

[3] https://figshare.com/s/f918e978205b5a9c27ab

	ProMapCZ			ProMapEN		
model	F1	prec.	recall	F1	prec.	recall
ProMap-NN-N	0.74	0.70	0.79	0.69	0.63	0.77
ProMap-NN-D	0.77	0.78	0.76	0.70	0.61	0.82
Bard-D/P	0.74	0.78	0.70	0.71	0.68	0.73
Bard-N/P	0.74	0.61	0.94	0.69	0.68	0.70
Bing-N/C	0.68	0.79	0.59	0.65	0.71	0.59
Bing-N/P	0.79	0.77	0.81	0.61	0.75	0.51
ChatGPT-D/C	0.45	0.54	0.40	0.53	0.49	0.59
ChatGPT-N/C	0.72	0.71	0.74	0.64	0.68	0.60
ChatGPT-N/P	0.70	0.80	0.62	0.49	0.86	0.35
LLaMA-N/P	0.39	0.34	0.46	0.48	0.44	0.53
Bard-D/P-vote	0.76	0.80	0.72	0.75	0.72	0.78
Bard-N/P-vote	0.75	0.62	0.94	0.67	0.66	0.68
Bing-N/PC-vote	0.80	0.78	0.83	0.62	0.78	0.51
ChatGPT-D/C-vote	0.45	0.60	0.36	0.59	0.58	0.59
ChatGPT-N/C-vote	0.74	0.78	0.71	0.67	0.72	0.62
LLaMA-N/P-vote	0.42	0.29	0.45	0.50	0.48	0.53
	AmWaPm			AmGoPm		
model	F1	prec.	recall	F1	prec.	recall
ProMap-N	0.92	0.90	0.94	0.99	1.00	0.98
ChatGPT-N/C	0.67	0.95	0.52	0.79	0.99	0.66
ChatGPT-N/P	0.67	0.98	0.51	0.66	0.98	0.50
Bing-N/C	0.78	0.97	0.66	0.90	0.98	0.84
Bing-N/P	0.56	0.95	0.39	0.70	0.99	0.54
ChatGPT-N/C-vote	0.69	0.98	0.54	0.83	0.99	0.71
ChatGPT-N/P-vote	0.68	0.99	0.52	0.66	1.00	0.49
Bing-N/C-vote	0.80	0.98	0.68	0.92	1.00	0.86

Table 5: Results of the baseline experiments. For the voting version, we test the better from Product Matching Championship and Pair-by-Pair versions. For Bing, it is Product-Matching-Championship on ProMapEN and Pair-by-Pair prompting for ProMapCZ.

names of the products on the ProMapCZ dataset. When the descriptions are also included, its performance however drops significantly (from F1 score 0.72 to 0.45), while the baseline score grows from 0.74 to 0.77. It seems that the longer description confuses the model rather than helping it. The results with the pair-by-pair prompt are very similar to the Product-Matching-Championship ones. In the ProMapEN dataset, the situation is similar – the best result is obtained with names only and with the Product-Matching-Championship prompt.

On the Amazon-Google dataset, ChatGPT is able to obtain F1 scores of around 0.80 for the Product Matching Championship prompt and around 0.65 for the Pair-by-Pair prompt. On the Amazon-Walmart dataset, both prompts get similar F1 scores of 0.67. Results on these datasets are thus much worse than the baseline model. All the runs of both the models have very high precision and very low recall, meaning that the model struggles to identify matching pairs of products, but can distinguish non-matching pairs.

Overall, ChatGPT typically has similar or better results with the Product Matching Championship prompt compared to the Pair-by-Pair one. Given that it is also easier to use and allows an evaluation of multiple pairs of products at the same time, we would recommend using this prompt with ChatGPT.

Bard is unfortunately unable to work with the Product-Matching-Championship prompt, so we only used the Pair-by-Pair one. The results do not seem to change much with the inclusion of the product descriptions. Bard's results are very close to the baseline in all the tested cases.

Llama2 is a relatively small model and is thus also not able to work with the more complicated Product-Matching-Championship prompt. Llama2 also officially does not support Czech, which leads to its results on ProMapCZ being much worse than the baseline ProMap-NN model (Llama2 only got F1 score of 0.39 – the worst result of all the models). On ProMapEN, its performance is also quite poor, again obtaining the worst results of all the tested models. Due to its poor performance on the ProMap datasets, we did not evaluate it on the AmWaPm and AmGoPm datasets at all.

We tested both prompts with Bing as the model has the same baseline LLM as ChatGPT. Due to the relatively long time it takes this model to generate the answer (partially because it often uses the internet to look up the specifications of the products), we only ran the experiment with product names and omitted the one with names and short descriptions.

Bing with the Pair-by-Pair prompt provided the best result of all the models on the ProMapCZ dataset, reaching an F1 score of 0.79, which is better than the ProMap-NN baseline. The results are better than ChatGPT, which should not be surprising, as Bing is based on the newer version of the GPT model. On ProMapEn, Bing got an F1 score of 0.61, lower than the 0.69 of the baseline model. Bing got slightly better results with the Product-Matching-Championship prompt on the ProMapEN dataset, but much worse result on the ProMapCZ dataset.

On the Amazon-Walmart and Amazon-Google datasets, the Pair-by-Pair prompt gave much weaker results than the Product Matching Championship. Compared to the baseline model, the results on these datasets are again much worse, showing that the baseline is quite strong for datasets with rather simple non-matches.

B.3.1 Voting Models

As we ran each of the experiments three times, we can also use the data to evaluate a situation in which we ask the model three times about each of the products and take the most common prediction. The results of these experiments are also in Table 5 and use the "-vote" suffix. They are similar to the ones obtained by each of the models. The best results using only names were reached by Bing with F1 score of 0.80 on ProMapCz and by ChatGPT and Bard with an F1 score of 0.67 on ProMapEn. The LLaMA model again showed very poor and inconsistent results for both Czech and English. When we compare the models in the experiment with names and short descriptions, the best result was obtained by Bard with an F1 score of 0.76 on ProMapCz and 0.75 on ProMapEn. ChatGPT with descriptions has much lower score than ChatGPT with names only even in this mode.

B.3.2 Overall Comparison

Regarding the overall comparison of the individual older LLMs, the best results for the Czech ProMapCz dataset were obtained with Bing. However, the best results for the ProMapEN dataset were obtained by Bard. On the other hand, while Bing had the highest F1 score, it was the slowest one. Bing and LLaMA were both the models with the most limited input character size. Bing was also the model that refused to follow required instructions, especially when using the Pair-by-Pair prompt. Therefore, even though Bing has the highest F1 score, ChatGPT is much more user-friendly and faster and evaluating ChatGPT is significantly easier especially while using the Pair-by-Pair prompt. Bard was the only model where short descriptions helped to improve results. For ChatGPT, other information about the products only worsened the results and Bing and LLaMA were even not able to respond while including short descriptions due to its strict input size limits.

C Implementation Details

C.1 Description Generation Prompt

The prompt used to generate the product descriptions is shown below.

> Create a more detailed description of the following product given the information from the following JSON dictionary. Answer using JSON.
> `<empty line>`
> `<JSON representation of product info>`

The JSON representation of the product is directly a JSON dictionary whose keys are the column names from the dataset and the values from these columns. We use the capability of the current models to generate the output in a given JSON Schema and request an output that contains the product name, product brand, product price, and the generated product description. The generated product descriptions are processed in the following experiments in the same way as if they were extracted from the webpages. The specific JSON Schema used can be found in the source codes.

C.2 Models and Runtime

We use three LLMs to generate these product descriptions – Llama3.2 (a small 3B parameter model, digest `a80c4f17acd5`), Gemma 3 (a 12B parameter model, digest `f4031aab637d`), and DeepSeek-r1 (a 32B parameter model, digest `edba8017331d`). All these models are run through the Ollama API and use the Q4_K_M quantization. All the models use the default parameters from Ollama, except for the input context size, which was set to default 4k tokens for the description generation experiments, and to 12k or 32k tokens for the matching experiments (the larger value was used only for the few-shot learning experiments).

Llama3.2 is a newer version of the Llama2 model. It is developed by Meta AI and was introduced in September 2024. The model is available in four different sizes; we use the 3B variant as a representative of a modern small model.

Gemma3 (Gemma Team et al., 2025) is developed by Google DeepMind and released in March 2025. It is intended for working on resource-limited devices and excels in tasks such as question answering, summarization, and reasoning. Several sizes of the model are available; we use the 12B version as an example of a modern middle-sized model.

DeepSeek-r1 (DeepSeek-AI et al., 2025) was released in January 2025 by DeepSeek-AI and was able to beat even larger and more complex models in many different tasks. We use the 32B version of the model distilled from Qwen.

The experiments were run on a heterogeneous cluster with various types of GPUs. Most of the experiments were run using nVidia A40 or nVidia A100; some of them were run on an nVidia H100. The specific GPU should not have any effect on the final results as long as the whole model fits in the GPU memory. The runtime of the experiments was in the order of hours per dataset (most of the time was spent generating the descriptions).

C.3 Product Matching Prompts

The prompt used for the Pair-by-Pair matching experiments is shown below.

> Are these two products the same? Answer using JSON.
> First product: `<product1 JSON>`
> Second product: `<product2 JSON>`

The prompt used for the few-shot-learning matching experiments, extended by one positive and one negative example, is shown below.

> Are these two products the same? Answer using JSON.
> EXAMPLE 1:
> `<empty line>`
> First product: `<pos. ex., product 1>`
> Second product: `<pos. ex., product 2>`
> `<empty line>`
> Output: `{"same_product": true}`
> `<empty line>`
> EXAMPLE 2: `<empty line>`
> First product: `<neg. ex., product 1>`
> Second product: `<neg. ex., product 2>`
> `<empty line>`
> Output: `{"same_product": false}`
> `<empty line>`
> INPUT: `<empty line>`
> First product: `<product 1>`
> Second product: `<product 2>`
> `<empty line>`
> Output:

C.4 Statistical Evaluation Details

This section contains detailed results of the statistical evaluation. That were used to create the critical difference plot in the main paper. Table 6 contains the pairwise comparisons of different models and prompt combinations on the product matching task. Table 7 contains the pairwise comparisons of the different models as description generators. In both cases, the difference Conover test with Bonferroni correction is used as a posthoc test after Friedman test showed that the differences between the groups are statistically significant.

Table 6: Details of statistical evaluation for different LLM/prompt combination and the ProMap-NN model. p-values of Conover test with Bonferroni correction. Model names in columns are abbreviated (full names are in rows).

	DS/FSL	DS/PP	G3/FSL	G3/PP	L3.2/FSL	L3.2/PP	PM-NN
DS-r1_32B/FSL	1.0000	0.0000	1.0000	0.0488	0.0000	0.0000	1.0000
DS-r1_32B/PP	0.0000	1.0000	0.0000	0.0000	0.0001	0.0000	0.0000
Gemma3_12B/FSL	1.0000	0.0000	1.0000	0.0002	0.0000	0.0000	0.0297
Gemma3_12B/PP	0.0488	0.0000	0.0002	1.0000	0.0000	0.0000	1.0000
Llama3.2_3B/FSL	0.0000	0.0001	0.0000	0.0000	1.0000	1.0000	0.0000
Llama3.2_3B/PP	0.0000	0.0000	0.0000	0.0000	1.0000	1.0000	0.0000
ProMap-NN	1.0000	0.0000	0.0297	1.0000	0.0000	0.0000	1.0000

Table 7: Details of statistical evaluation for different LLMs as description generators. p-values of Conover test with Bonferroni correction.

	DS	Gemma3	Llama3.2	None
DS-r1	1.0000	0.0249	1.0000	0.1133
Gemma3	0.0249	1.0000	0.0045	0.0000
Llama3.2	1.0000	0.0045	1.0000	0.4147
None	0.1133	0.0000	0.4147	1.0000