An Empirical Study to Use Large Language Models to Extract Named Entities from Repetitive Texts

Angelica Lo Duca[a]
Institute of Informatics and Telematics of the National Research Council, via G. Moruzzi, 1, 56124 Pisa, Italy

Keywords: Large Language Models, Prompt Engineering, Named Entities Extraction.

Abstract: Large language models (LLMs) are a very recent technology that assists researchers, developers, and people in general to complete their tasks quickly. The main difficulty in using this technology is defining effective instructions for the models, understanding the models' behavior, and evaluating the correctness of the produced results. This paper describes a possible approach based on LLMs to extract named entities from repetitive texts, such as population registries. The paper focuses on two LLMs (GPT 3.5 Turbo and GPT 4), and runs some empirical experiments based on different levels of detail contained in the instructions. Results show that the best performance is achieved with GPT 4, with a high level of detail in the instructions and the highest costs. The trade-off between costs and performance is given when using GPT 3.5 Turbo when the level of detail is medium.

1 INTRODUCTION

The advent of Large Language Models (LLMs) (Brown et al., 2020) has opened the way for new strategies to automate different tasks, including extracting information from repetitive texts, with potential benefits for humans (Weber et al., 1980). In this context, numerous industrial and academic initiatives are present in various sectors, such as healthcare (Gebreab et al., 2024), Robotic Process Automation (RPA) (Fani Sani et al., 2023), and finance (Olaoye and Jonathan, 2024). These initiatives leverage the advanced capabilities of LLMs to process and interpret vast amounts of unstructured text data, thereby facilitating more efficient research and development processes. However, using LLMs is still immature, as it is a very recent technology that requires formulating adequate instructions, understanding the models' behavior, and a rigorous evaluation of the produced outputs (Zhang et al., 2023). In addition, the challenges associated with deploying LLMs involve ethical and societal considerations. For instance, integrating LLMs into workflows should consider the potential biases in the training data, which can perpetuate stereotypes and unfair decisions if not adequately addressed (Bang et al., 2023).

[a] https://orcid.org/0000-0002-5252-6966

This paper focuses on a practical experiment using LLMs to extract named entities and their relations from repetitive texts and convert them into a tabular format. Traditional rule-based approaches (Waltl et al., 2018) often struggle to handle the nuances and variations present in repetitive texts. In contrast, LLMs have demonstrated their ability to generalize patterns and adapt to diverse textual formats (Pakhale, 2023).

This paper improves on previous work, which used an approach that combined rule-based and manual correction by a domain expert to extract named entities from a registry of births (Lo Duca et al., 2023). This combination of rule-based and manual corrections reached an accuracy of about 0.99. The main drawback of this approach was the presence of humans who signaled and modified rules in the event of detected errors. This human presence made the algorithm particularly slow, time-consuming, and resource-consuming.

This paper follows the line of research of few-shot prompting, which is a prompt engineering technique that tries to obtain satisfactory results from LLMs by providing examples as input (Ma et al., 2023). In particular, this paper applies an LLM-based approach to perform the same task without human intervention. Compared to the rule-based approach, the LLM-based approach is generated via machines

Lo Duca, A.
An Empirical Study to Use Large Language Models to Extract Named Entities from Repetitive Texts.
DOI: 10.5220/0013066500003825
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 20th International Conference on Web Information Systems and Technologies (WEBIST 2024)*, pages 417-424
ISBN: 978-989-758-718-4; ISSN: 2184-3252
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

and does not rely on human correction. The objective is to see whether LLMs can still generate acceptable results while reducing time and resources. This approach was partially described in a previous paper (Lo Duca et al., 2024), where only Generative Pretrained Transformer 3.5 Turbo (GPT-3.5-turbo) was used as an LLM. Instead, this paper compares GPT 3.5 Turbo and GPT 4 (Achiam et al., 2023) to define different scenarios. The objective is to investigate the effectiveness of these models in handling repetitive textual data and to evaluate the impact of varying levels of detail in the provided instructions. This empirical study aims to assess the efficacy of LLMs in extracting named entities from repetitive texts without domain-specific adaptations or manipulations. Due to their extensive training on diverse datasets, this approach is grounded in the belief that LLMs can inherently generalize across various domains and extract named entities effectively.

To assess the performance of our approach, some empirical experiments are described, each employing different instruction levels. Results show that the best performance is obtained when instructions maintain a high level of detail. However, in some cases, a moderate level of detail reaches the same performance as the high level. The paper also includes a discussion on associated costs.

The remainder of the paper is organized as follows: Section 2 reviews the related literature, and Section 3 describes the experiments followed in this paper. Section 4 discusses results. Finally, Section 5 gives conclusions and future work.

2 RELATED WORK

Extracting named entities and their relations from repetitive texts is a task belonging to the domain of Named Entity Recognition (NER). NER is a well-known Natural Language Processing (NLP) task that identifies and classifies named entities in texts, such as persons, locations, organizations, dates, etc. Traditional NER techniques include rule-based, learning-based, and hybrid approaches (Waltl et al., 2018; Goyal et al., 2018; Humbel et al., 2021). The literature about NER is vast and reviewing all the works is beyond the scope of this paper. In this section, we describe only some representative works that help to contextualize the research done in this paper.

2.1 Traditional Approaches to NER

Rule-based approaches rely on predefined rules and patterns crafted based on the language's linguistic properties. These rules identify entities based on their grammatical and contextual features. For instance, in our previous paper, we used a rule-based approach to perform the same task as in this paper (Lo Duca et al., 2023). Other examples include the research done by Eftimov et al., who presented a rule-based NER method called drNER for extracting evidence-based dietary recommendations from text (Eftimov et al., 2017), and MeTAE (Medical Texts Annotation and Exploration), a platform for the annotation of medical entries (Ben Abacha and Zweigenbaum, 2011). Learning-based approaches utilize machine learning algorithms to learn from annotated datasets. These approaches can range from traditional machine learning methods to more advanced deep learning techniques. A notable example is the work by Pooja and Jagadeesh, who used a deep learning-based approach using Bidirectional Long Short Term Memory (BiLSTM), Bidirectional Encoder Representations from Transformers (BERT), and Conditional Random Field (CRF) models for biomedical NER (Pooja and Jagadeesh, 2024). Hybrid approaches combine the strengths of both rule-based and learning-based methods. They often employ machine learning models to capture complex patterns and use rule-based systems to handle well-defined entities. An example of a hybrid approach is presented by Ji et al., who applied a BiLSTM-CRF model with an attention mechanism and post-processing rules for NER in Chinese electronic medical records (Ji et al., 2019).

While rule-based approaches can be very effective for well-structured entities, learning-based approaches offer greater flexibility and adaptability to diverse and complex entity patterns. Hybrid approaches leverage both advantages to achieve optimal performance on NER tasks. The choice of approach often depends on the task's specific requirements, the text's nature, and the availability of annotated data.

2.2 LLM-Based Approaches to NER

Given the growing interest in LLMs, a thriving literature is also developing around the use of LLMs to perform NER tasks. One way to use LLMs for NER is to transform the sequence labeling task of NER to a generation task that can be easily adapted by LLMs. Wang et al. propose a method to encode the input sentence and the entity type as a single prompt and decode the entity span as the output (Wang et al., 2023). They also introduce a self-verification strategy to address the hallucination issue of LLMs (Braverman et al., 2020), where they may generate spurious entities that do not exist in the input sentence. Another way to use LLMs for NER is

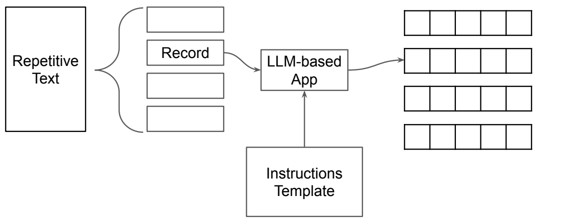


Figure 1: The architecture of the proposed system.

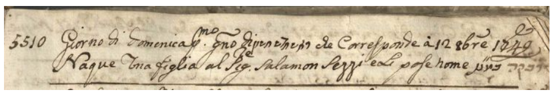
to leverage their ability to access external knowledge sources, such as Wikipedia or other knowledge bases. Malmasi et al. present a large multilingual dataset for NER that covers three domains (Wiki sentences, questions, and search queries) across 11 languages and multilingual and code-mixing subsets (Malmasi et al., 2022). Some works have also performed NER without labeled data or pre-trained models. For example, Luo et al. propose a fully unsupervised NER model that only relies on pre-trained word embeddings (Luo et al., 2019). Another approach uses natural language prompts to guide LLMs to perform NER without fine-tuning or labeling data (Ashok and Lipton, 2023). Shen et al. propose a simple method that applies an LLM to encode queries and documents into dense vectors and then computes their cosine similarity (Shen et al., 2023).

Compared to the current literature, this paper describes an empirical study to evaluate the performance of two popular LLMs, without any domain- adaptation or other manipulations, to extract named entities from repetitive texts. The objective is to evaluate if LLMs can be used as they are for this type of task without any adaptation.

3 EXPERIMENTS

This paper uses an LLM-based application to extract named entities and their relationships from repetitive texts. Figure 1 illustrates the implemented system architecture for the task. First, the workflow splits the repetitive text into single records. Next, each record and an instructions template containing text describing the task to perform are provided as input to the LLM-based app. As an output, the LLM-based app generates data in a tabular format.

This architecture is used to compare the performance of two LLMs: GPT 3.5 Turbo and GPT 4. For each LLM, three levels of instruction templates are defined: simple, medium, and detailed, based on the level of detail they describe. Each instruction template is provided as an input to the LLM-based app. Results are compared to evaluate the minimum detail required to achieve satisfactory results. In total, six experiments have been run.



Original Text in Italian
5510 Giorno di domenica primo giorno di Rosh Chodesh Cheshvan che corresponde a 12 ottobre 1749. Naque una figlia al Signore Salamon Sezzi e si pose nome Ribqa bemazal tov

5510 Sunday first day of Rosh Chodesh Cheshvan which corresponds to October 12, 1749. A daughter was born to Lord Salamon Sezzi and she was named Ribqa bemazal tov

Figure 2: An example of a record in the Registry of Births of the Historical Archive of the Jewish Community of Pisa.

3.1 Case Study

The case study considered the Registry of Births of the Historical Archive of the Jewish Community of Pisa (Lo Duca et al., 2023). This registry contains 262 records related to the members of the Pisa Jewish community, their date of birth, their sex, and their father's name. The birth registry started on October 12, 1749, and ended on November 1809. Each record contains the date of birth, indicated according to the Jewish and Gregorian systems (for a baby born on March 10th, the registry also shows the day of Rosh Chodesh Cheshvan 5510). The original document is written in Italian, and all the records have more or less the same structure. Figure 2 shows a sample record within its original version (the manuscript at the top of the figure), the Italian transcription (in the middle), and the English translation (at the bottom). The figure shows the text translated into English for convenience, although tests have been done using the original Italian language.

3.2 Model Instructions

Three levels of instructions were defined: simple, medium, and detailed, based on the level of detail they describe. Each instruction was provided as an input to the models. Results were compared to evaluate the minimum detail required to achieve satisfactory results.

3.2.1 Simple Scenario

The following snippet of code shows the simple instructions used in this first experiment:

```
For each line extract:
- extract child name, father name,
gender, date of birth and format as CSV

Instructions:
```

```
- If you find a son, set gender to M
- If you find a daughter, gender is F
- Do not include besiman tov
in the child's name

Answer by formatting the output in CSV.
```

Only the information to extract and some record-specific details were provided as an input, such as how to extract the sex and to omit the words *besiman tov* from the child's name since this is a Jewish expression the model may not know.

3.2.2 Medium Scenario

This scenario slightly complicated the instruction by adding an example of how to format the output, as shown in the following snippet of code:

```
For each line extract:
- extract child name, father name, gender,
date of birth and format as CSV

Instructions:
- If you find a son, set gender to M
- If you find a daughter, gender is F
- Do not include besiman tov
in the child's name

Follow this example:
Input:
<5510 Giorno di domenica primo giorno
di Rosh Chodesh Cheshvan che
corresponde a 12 ottobre 1749.
Naque una figlia al Signore Salamon Sezzi
e si pose nome Ribqa bemazal tov>

Output:
Ribqa,Salamon Sezzi,F,1749-10-17

Answer by formatting the output in CSV.
```

This instruction provides an example in angular brackets, defines the desired output, and shows how to format it. For instance, it asks the model to format the date following the yyyy-mm-dd notation.

3.2.3 Detailed Scenario

The detailed scenario complicated the instruction by adding what the model should do if some information is missing, as described in the following piece of code:

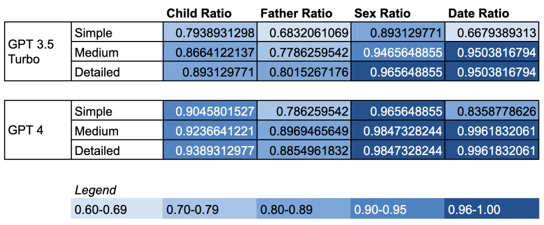
```
For each line extract:
- extract child name,father name,
gender, date of birth and format as CSV

Instructions:
- If you find a son, set gender to M
- If you find a daughter, gender is F
- Do not include besiman tov in
the child's name
- If the child's name is not present,
add only a comma
- If the father's name is not present,
add only a comma
- If the date of birth is not present,
add only a comma

Follow this example:
Input:
<5510 Giorno di domenica primo giorno
di Rosh Chodesh Cheshvan
che corresponde a 12 ottobre 1749.
Naque una figlia al Signore Salamon Sezzi
e si pose nome Ribqa bemazal tov>

Output:
Ribqa,Salamon Sezzi,F,1749-10-17

Answer by formatting the output in CSV.
```

3.3 Metrics

In total, six experiments were run. Results generated by each model were compared with a ground truth generated using the improved rule-based approach described in a previous paper (Lo Duca et al., 2023). Thanks to the presence of a human using the rule-based approach, previous results identified all the records correctly; thus, the output produced by that approach is used as a ground truth for this paper.

In this paper, the following metrics are measured:

- *Father Ratio:* the ratio between the number of entities recognized correctly as a father and the total number of records;
- *Child Ratio:* the ratio between the number of entities recognized correctly as a child and the total number of records;
- *Sex Ratio:* the ratio between the number of records where the child's sex is correctly identified and the total number of records;
- *Date Ratio:* the ratio between the number of records where the child's date of birth is correctly identified and the total number of records.
- *Total Ratio:* the ratio between the sum of the number of entities recognized correctly as a father, the number of entities recognized correctly as a child, the number of records where the child's sex is correctly identified, the number of records where the



		Child Ratio	Father Ratio	Sex Ratio	Date Ratio
GPT 3.5 Turbo	Simple	0.7938931298	0.6832061069	0.893129771	0.6679389313
	Medium	0.8664122137	0.7786259542	0.9465648855	0.9503816794
	Detailed	0.893129771	0.8015267176	0.965648855	0.9503816794
GPT 4	Simple	0.9045801527	0.786259542	0.965648855	0.8358778626
	Medium	0.9236641221	0.8969465649	0.9847328244	0.9961832061
	Detailed	0.9389312977	0.8854961832	0.9847328244	0.9961832061

Legend

0.60-0.69	0.70-0.79	0.80-0.89	0.90-0.95	0.96-1.00

Figure 3: The Father, Child, Sex, and Date ratios for all the experiments. The darker the cell color, the better the model performance. The father is the hardest entity to extract in all scenarios, while the easiest is the date. This is because the parent entity sometimes has a different structure from the classic one, while the date is almost always in the same format.

child's date of birth is correctly identified and the total number of records multiplied for four.

3.4 Results

Figure 3 shows the Father, Child, Sex, and Date ratios for all the experiments. As expected the best model is GPT 4 with detailed instructions. However, the Sex and Date ratios are the same for GPT 4 with medium and detailed instructions. Adding details to the instructions does not improve the model's performance. It is also interesting to notice that GPT 3.5 Turbo with detailed instructions performs better than GPT 4 with simple instructions. This suggests the importance of instructions in obtaining the correct outputs.

Figure 4 shows the Total Ratio for all the experiments. Only GPT 4 with medium and detailed instructions has a Total Ratio greater than 0.95. The only surprise is given by GPT 3.5 Turbo medium and detailed, which outperforms GPT 4 simple. This confirms the importance of formulating instructions properly.

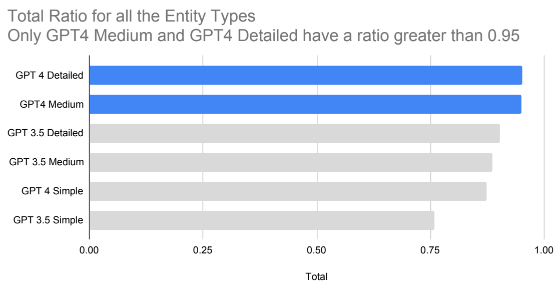
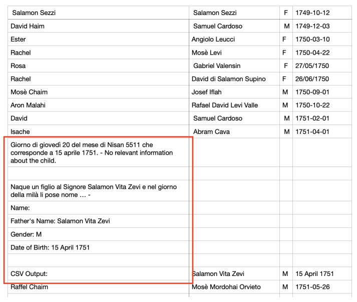


Figure 4: The Total Ratio for all the experiments.

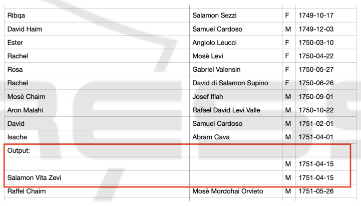
4 DISCUSSION

The experiments show that GPT 4 detailed is the best model. However, any model can reach a ratio of 100%, so it is essential to understand which types of errors each model makes. The first consideration involves the formatting style. Any of the GPT 3.5 Turbo models can generate a perfect CSV file. Figure 5 shows an example of a formatting error when using GPT 3.5 Turbo. If the model cannot identify the entities correctly, it writes additional text.



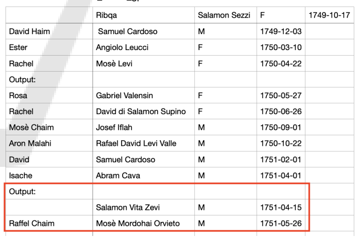
Salamon Sezzi	Salamon Sezzi	F	1749-10-12
David Haim	Samuel Cardoso	M	1749-12-03
Ester	Angiolo Leucci	F	1750-03-10
Rachel	Mosè Levi	F	1750-04-22
Rosa	Gabriel Valensin	F	27/05/1750
Rachel	David di Salamon Supino	F	26/06/1750
Mosè Chaim	Josef Iflah	M	1750-09-01
Aron Malahi	Rafael David Levi Valle	M	1750-10-22
David	Samuel Cardoso	M	1751-02-01
Isache	Abram Cava	M	1751-04-01
Giorno di giovedì 20 del mese di Nisan 5511 che corresponde a 15 aprile 1751. - No relevant information about the child.			
Naque un figlio al Signore Salamon Vita Zevi e nel giorno della milà li pose nome ... -			
Name:			
Father's Name: Salamon Vita Zevi			
Gender: M			
Date of Birth: 15 April 1751			
CSV Output:	Salamon Vita Zevi	M	15 April 1751
Raffel Chaim	Mosè Mordohai Orvieto	M	1751-05-26

(a) GPT-3.5 Turbo Simple.



Ribqa	Salamon Sezzi	F	1749-10-17
David Haim	Samuel Cardoso	M	1749-12-03
Ester	Angiolo Leucci	F	1750-03-10
Rachel	Mosè Levi	F	1750-04-22
Rosa	Gabriel Valensin	F	1750-05-27
Rachel	David di Salamon Supino	F	1750-06-26
Mosè Chaim	Josef Iflah	M	1750-09-01
Aron Malahi	Rafael David Levi Valle	M	1750-10-22
David	Samuel Cardoso	M	1751-02-01
Isache	Abram Cava	M	1751-04-01
Output:			
		M	1751-04-15
Salamon Vita Zevi		M	1751-04-15
Raffel Chaim	Mosè Mordohai Orvieto	M	1751-05-26

(b) GPT-3.5 Turbo Medium.

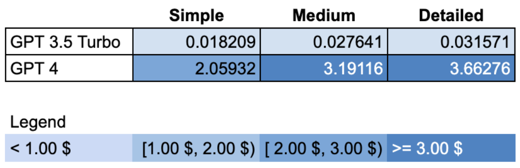


	Ribqa	Salamon Sezzi	F	1749-10-17
David Haim	Samuel Cardoso	M	1749-12-03	
Ester	Angiolo Leucci	F	1750-03-10	
Rachel	Mosè Levi	F	1750-04-22	
Output:				
Rosa	Gabriel Valensin	F	1750-05-27	
Rachel	David di Salamon Supino	F	1750-06-26	
Mosè Chaim	Josef Iflah	M	1750-09-01	
Aron Malahi	Rafael David Levi Valle	M	1750-10-22	
David	Samuel Cardoso	M	1751-02-01	
Isache	Abram Cava	M	1751-04-01	
Output:				
	Salamon Vita Zevi	M	1751-04-15	
Raffel Chaim	Mosè Mordohai Orvieto	M	1751-05-26	

(c) GPT-3.5 Turbo Detailed.

Figure 5: Formatting problems with GPT 3.5 Turbo. When the model does not find an entity in a record: (a) GPT-3.5 Turbo simple adds a long text that specifies what it did not find. (b) GPT-3.5 Turbo medium uses three lines for the missing entity. This causes formatting problems for the following lines. (c) GPT-3.5 detailed uses two lines for the missing entity but does not cause formatting problems for the following lines.

As the instructions become more detailed, the model makes fewer formatting errors (as shown in Figure 5), which include additional text or the " symbol. A more in-depth study should be done on how to eliminate these types of errors starting from input instructions. This aspect would lead to a complica-



	Simple	Medium	Detailed
GPT 3.5 Turbo	0.018209	0.027641	0.031571
GPT 4	2.05932	3.19116	3.66276

Legend

< 1.00 $	[1.00 $, 2.00 $)	[2.00 $, 3.00 $)	>= 3.00 $

Figure 6: Total costs for each model.

tion of the instructions and could be studied as future work. Anyway, this type of problem is completely solved when using GPT 4. However, GPT 4 simple introduces another formatting problem, which consists of adding extra quotes (”) to some rows, such as in the following example: ”Ribqa,Salamon Sezzi,F,12 ottobre 1749”. This problem is completely solved with GPT 4 medium and detailed.

The second consideration refers to the type of entities not recognized. All the models cannot properly recognize the father’s name with the following structure: Name Surname *del fu* Name Surname, where the first pair Name Surname refers to the father’s name, *del fu* is an Italian expression to literary indicate *of the past* and the second Name Surname is the grand father’s name. Among all the records, there are six records with this expression. Only GPT 4 medium and GPT 3.5 Turbo detailed can recognize 3 of the six entities with this expression. GPT 3.5 Turbo simple recognizes two entities, while GPT 4 detailed only one. The remaining models do not recognize any entity of this type.

The third consideration regards costs. OpenAI (which GPT 3.5 Turbo and GPT 4 belong to) calculates costs based on the model type and the number of input and output tokens. The sum of the input and output costs gives the total cost. Based on the prices updated on 2024 June 19, the output cost C_o for GPT-3.5 Turbo is 1.5 \$/ 1M tokens, and for GPT-4, it is 120 \$ / 1M tokens. We calculate the total output cost for a model as follows:

$$C_{output} = \frac{T_o \times N_r \times C_o}{10^6} \tag{1}$$

where T_o is the number of tokens in output, N_r is the number of records (262) and C_o is the output cost. We consider an average output size of 32 characters. If a token is four characters, we can set $T_o = 8$.

The input cost C_i for GPT-3.5 Turbo is 0.5 \$/ 1M tokens, and for GPT-4, it is 60 \$ / 1M tokens. We calculate the total input cost for a model as follows:

$$C_{input} = \frac{T_i \times N_r \times C_i}{10^6} \tag{2}$$

where T_i is the number of tokens in input, it is given by the sum of the record size (an average of 171 records) and the length of the instruction divided by the token size (4). The sum of the input and output costs gives

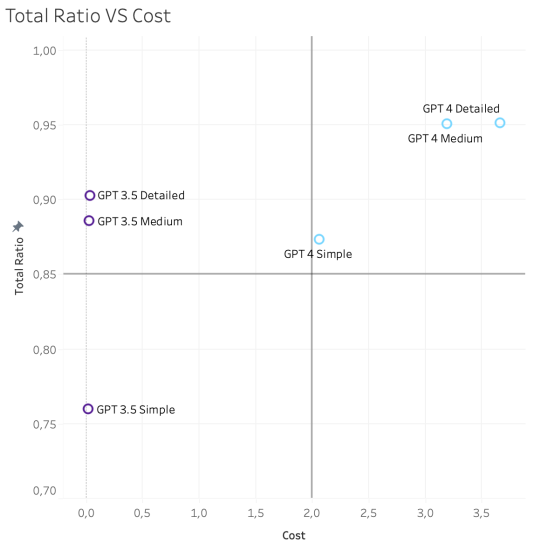


Figure 7: Trade-off between Ratio and Costs.

the total cost for each model. Figure 6 shows the total costs in Dollars for each scenario.

While the cost of GPT 3.5 Turbo is minimal (less than one dollar) for all scenarios, the cost of GPT 4 explodes as the number of input tokens increases. Figure 7 shows the trade-off between ratio and costs. GPT 4 models are in the top right part of the chart, meaning they reach high performance but are expensive. GPT 3.5 Turbo detailed and medium are in the top left part of the chart, meaning they have high performance but are cheaper. GPT 3.5 Turbo simple is the most affordable but has the worst performance. Based on this chart, the best models are GPT 3.5 Turbo detailed and medium.

When choosing between models, in addition to costs, the task’s complexity, token length, accuracy needs, and frequency should also be considered. For simple tasks like summarization or basic queries, GPT-3.5 Turbo medium and detailed are cost-effective choices, while more complex or high-stakes tasks requiring deeper reasoning or precision justify the higher cost of GPT-4 medium and detailed. If speed and frequent usage are priorities, cheaper models are preferable, but for specialized domains or high accuracy, GPT-4 may be necessary despite its cost.

A final consideration concerns optimizing instructions for maximum performance and cost efficiency. However, to achieve maximum performance, the model should know the context in which it operates well, and the instructions provided as input must be as clear as possible. A separate analysis should be done to evaluate the model’s efficiency, response times, and human difficulty in writing optimized instructions. These aspects will be the subject of future work and study.

The main limitation of this study is that the analyzed records have a standardized format. Thus, the observed results may differ significantly in different scenarios where the input text is less rigidly structured. However, the overall architecture described in 1 remains valid, although more research should be done on the instructions to be provided as input to the model.

5 CONCLUSIONS AND FUTURE WORK

This paper explored the application of LLMs, specifically GPT 3.5 Turbo and GPT 4, for extracting named entities from repetitive texts. This investigation aimed to study the effectiveness of these models in handling such structured texts by defining different types of instructions with an increasing level of detail. This paper has demonstrated that all the tested LLMs reach a total ratio greater than 0.75. In all cases, costs should also be considered while choosing the best model.

This paper has investigated two specific models: GPT 3.5 Turbo and GPT 4. As new models are released continuously, future work could include comparing them and the costs of models released by different providers, such as Google and Meta.

Future work could also use the best scenario at scale and implement an LLM-based app that receives input from repetitive text and an output example and returns the formatted CSV text as output. In addition, instruction optimization could be investigated, with a more detailed analysis of the model to use based on the task requirements.

REFERENCES

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. (2023). Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Ashok, D. and Lipton, Z. C. (2023). Promptner: Prompting for named entity recognition. *arXiv preprint arXiv:2305.15444*.

Bang, J., Lee, B.-T., and Park, P. (2023). Examination of ethical principles for llm-based recommendations in conversational ai. In *2023 International Conference on Platform Technology and Service (PlatCon)*, pages 109–113. IEEE.

Ben Abacha, A. and Zweigenbaum, P. (2011). Automatic extraction of semantic relations between medical entities: a rule based approach. *Journal of biomedical semantics*, 2:1–11.

Braverman, M., Chen, X., Kakade, S., Narasimhan, K., Zhang, C., and Zhang, Y. (2020). Calibration, entropy rates, and memory in language models. In *International Conference on Machine Learning*, pages 1089–1099. PMLR.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Eftimov, T., Koroušić Seljak, B., and Korošec, P. (2017). A rule-based named-entity recognition method for knowledge extraction of evidence-based dietary recommendations. *PloS one*, 12(6):e0179488.

Fani Sani, M., Sroka, M., and Burattin, A. (2023). Llms and process mining: Challenges in rpa: Task grouping, labelling and connector recommendation. In *International Conference on Process Mining*, pages 379–391. Springer.

Gebreab, S. A., Salah, K., Jayaraman, R., ur Rehman, M. H., and Ellaham, S. (2024). Llm-based framework for administrative task automation in healthcare. In *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–7. IEEE.

Goyal, A., Gupta, V., and Kumar, M. (2018). Recent named entity recognition and classification techniques: a systematic review. *Computer Science Review*, 29:21–43.

Humbel, M., Nyhan, J., Vlachidis, A., Sloan, K., and Ortolja-Baird, A. (2021). Named-entity recognition for early modern textual documents: a review of capabilities and challenges with strategies for the future. *Journal of Documentation*, 77(6):1223–1247.

Ji, B., Liu, R., Li, S., Yu, J., Wu, Q., Tan, Y., and Wu, J. (2019). A hybrid approach for named entity recognition in chinese electronic medical record. *BMC medical informatics and decision making*, 19:149–158.

Lo Duca, A., Abrate, M., Marchetti, A., and Moretti, M. (2024). Genealogical data-driven visits of historical cemeteries. *Informatics*, 11(1).

Lo Duca, A., Marchetti, A., Moretti, M., Diana, F., Toniazzi, M., and D'Errico, A. (2023). Genealogical data mining from historical archives: The case of the jewish community in pisa. *Informatics*, 10(2).

Luo, Y., Zhao, H., and Zhan, J. (2019). Named entity recognition only from word embeddings. *arXiv preprint arXiv:1909.00164*.

Ma, H., Zhang, C., Bian, Y., Liu, L., Zhang, Z., Zhao, P., Zhang, S., Fu, H., Hu, Q., and Wu, B. (2023). Fairness-guided few-shot prompting for large language models. *Advances in Neural Information Processing Systems*, 36:43136–43155.

Malmasi, S., Fang, A., Fetahu, B., Kar, S., and Rokhlenko, O. (2022). Multiconer: A large-scale multilingual dataset for complex named entity recognition. *arXiv preprint arXiv:2208.14536*.

Olaoye, G. and Jonathan, H. (EasyChair, 2024). The evolving role of large language models (llms) in banking. EasyChair Preprint no. 13367.

Pakhale, K. (2023). Comprehensive overview of named entity recognition: Models, domain-specific applications and challenges. *arXiv preprint arXiv:2309.14084*.

Pooja, H. and Jagadeesh, M. P. (2024). A deep learning based approach for biomedical named entity recognition using multitasking transfer learning with bilstm, bert and crf. *SN Computer Science*, 5(5):482.

Shen, T., Long, G., Geng, X., Tao, C., Zhou, T., and Jiang, D. (2023). Large language models are strong zero-shot retriever. *arXiv preprint arXiv:2304.14233*.

Waltl, B., Bonczek, G., and Matthes, F. (2018). Rule-based information extraction: Advantages, limitations, and perspectives. *Jusletter IT (02 2018)*, 4.

Wang, S., Sun, X., Li, X., Ouyang, R., Wu, F., Zhang, T., Li, J., and Wang, G. (2023). Gpt-ner: Named entity recognition via large language models. *arXiv preprint arXiv:2304.10428*.

Weber, A., Fussler, C., O'hanlon, J., Gierer, R., and Grandjean, E. (1980). Psychophysiological effects of repetitive tasks. *Ergonomics*, 23(11):1033–1046.

Zhang, S., Dong, L., Li, X., Zhang, S., Sun, X., Wang, S., Li, J., Hu, R., Zhang, T., Wu, F., et al. (2023). Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*.