

CS499 Research Thesis

Jason William Zalewski

Faculty Advisor: Kevin Chenchuan Chang

University Of Illinois Urbana-Champaign

12/10/2023

Contents

List of Figures	iv
List of Tables	v
1 Introduction	2
1.1 What is the problem?	2
1.2 Why is it important?	2
1.3 Why is it difficult?	3
1.4 What has been done?	3
1.5 What have you done?	4
1.6 What is the result?	4
1.7 Comparisons of LLM models	4
2 Related Work	5
2.1 DEPTA	5
2.2 Prompt engineering	5
2.3 Chain of thought untruthful explanations	6
2.4 Emotional language model prompts	6
3 Solutions & Techniques	7
3.1 Motivation	7
3.2 Design	7

3.3	Framework	8
4	Experiments	9
4.1	Classifying DEPTA regions that contain professor names	9
4.1.1	Classification prompt	9
4.1.2	DEPTA classification example 1	10
4.1.3	DEPTA classification example 2	10
4.1.4	DEPTA classification results	11
4.2	Extracting the professor’s name, position, and research interests for each pro- fessor	11
4.2.1	Professor information extraction prompt	11
4.2.2	Professor information extraction grading explanation	15
4.2.3	Professor information extraction example 1	15
4.2.4	Professor information extraction example 2	16
4.2.5	Professor information extraction example 3	18
4.3	Professor information extraction scoring results	21
4.4	Comparing the capabilities of LLMs Gpt3.5 and Beluga for information ex- traction	21
5	Future Work	22
5.1	Developing a generalized approach for programming with LLMs	22
5.2	Enhancing predictability and handling uncertainty	22
5.3	Implementing a verification and correction System	23
5.4	Exploring advanced prompt engineering	23
5.5	Investigating context management techniques	23
5.6	Enhancing model comparisons	23
6	Summary	24
6.1	LLM capabilities for information extraction	24

6.2	Overcoming LLM limitations	24
6.2.1	Chain-of-Thought prompting	24
6.2.2	Integration of external tools	25
6.2.3	Handling context length	25
6.2.4	Model comparison insights	25
	Bibliography	26

List of Figures

- 3.1 Illustration of the DEPTA method integration and data processing workflow. 8

List of Tables

4.1	Extraction Accuracy Scores for UIUC, MIT, and CMU	21
-----	---	----

Abstract

This thesis presents an innovative approach to unsupervised web information extraction using Large Language Models (LLMs), focusing on the extraction of faculty data from various university websites. The research entails the development and implementation of methods that enable LLMs, specifically GPT-3.5 Turbo and Beluga (a fine-tuned version of Llama2), to autonomously gather and process information from web sources. The extraction process involves using the DEPTA method for organizing web content into structured regions, thereby enhancing the efficiency of data extraction. The study also involves a comparative analysis of the two models, highlighting their individual strengths and limitations in web information extraction. The results demonstrate the potential of LLMs in reducing manual labor to traditional data extraction methods. This thesis aims to establish a foundational framework for the advancement and application of LLMs in the realm of unsupervised web information extraction. It seeks to provide a general blueprint for utilizing LLMs across a diverse range of web sources.

The source code used in this work can be found here:

<https://github.com/FireBirdJZ/jason-zalewski-llm-web-extraction/tree/main>

Acknowledgements

This thesis would not have been possible without the guidance and support of several individuals whose contributions I deeply appreciate.

First and foremost, I extend my heartfelt gratitude to Professor Kevin Chenchuan Chang for his invaluable advice, relentless support, and mentorship throughout my research journey. His expertise and insights have profoundly influenced my understanding and approach to the subject matter.

I am also immensely thankful to Zhijia Chen for his generous assistance with the Pydepta tool. His willingness to explain his modifications and code played a key role in the development of my solution

Chapter 1

Introduction

1.1 What is the problem?

The core focus of this thesis lies in harnessing the capabilities of Large Language Models (LLMs) for unsupervised web information extraction, with a broader aim of general applicability across various types of web pages. While the research predominantly centers around the extraction of information from faculty web pages, the methodologies and insights derived are designed to be adaptable for a wider range of web sources. This involves crafting techniques and strategies that enable LLMs to effectively identify, extract, and process diverse information sets found across the internet. The case study of faculty web pages serves as a specific example through which the potential and challenges of LLMs in broader web information extraction are explored and addressed.

1.2 Why is it important?

Utilizing Large Language Models (LLMs) for web information extraction significantly reduces manual effort in data collection. This technology streamlines the process, shifting the challenge of problem-solving and adaptation from human effort to automated systems. While it doesn't necessarily speed up processing time, it offers a more adaptable and less

labor-intensive alternative to manual methods. In the context of big data, where the volume and variety of information are overwhelming, this approach provides a flexible and efficient solution for gathering diverse web data.

1.3 Why is it difficult?

The primary difficulty lies in managing the non-deterministic outputs of LLMs and adapting programs to accommodate different models with varying behaviors and capacities. Another difficulty has to do with the models occasionally misinterpreting the task and attempting to create a program for solving the extraction task, rather than performing the extraction directly. Another significant challenge is ensuring that these models, with their differing context lengths and requirements for specific instructions (like chain-of-thought prompting), can effectively process web page texts of varying lengths and complexities. At times, the text provided to the large language model can be ambiguous, making it challenging for the model to discern the correct information to extract. This is particularly noticeable in instances where a job title includes a word that could also be interpreted as a name. In such cases, the model requires sufficient contextual information to accurately identify and extract the intended data. Some mitigation to this can be adding direct instructions for patterns that continually occur, for example for UIUC faculty web page: “Watch Out For: pronouns such as he, they, or her. Don’t add these anywhere to the JSON object”.

1.4 What has been done?

Prior efforts in this domain have included experiments with simpler web pages, leading to the identification of LLMs’ strengths in analyzing and extracting relevant information from large text blocks. However, these models often struggle with processing excessive information at once, leading to disorderly or incomplete extraction.

1.5 What have you done?

In my research, I focused on utilizing large language models (LLMs) for unsupervised information extraction, specifically targeting faculty web pages. By comparing the performance of two advanced models, GPT-3.5 Turbo and Beluga (a fine-tuned version of Llama2), I assessed their capabilities in this context. A key aspect of my approach involved employing the DEPTA method to systematically organize web page content into structured regions, facilitating targeted extraction. Additionally, I developed techniques for classifying these regions and extracting specific data, such as professors' names, positions, and research interests, using the LLMs, thereby streamlining the process of information gathering from complex web sources.

1.6 What is the result?

The result was being able to get the majority of professors' names, positions, and research interests into JSON objects from specific faculty webpages, notably from UIUC, CMU and MIT. The process highlighted differences in models' responses to varying prompts and their capabilities in handling detailed extractions. Some future work is still needed in dealing with edge cases and harder examples as the model can sometimes hallucinate results or get confused with the directions in the prompt.

1.7 Comparisons of LLM models

The research revealed that the Beluga model required more direct and detailed instructions to perform effectively, especially benefiting from chain-of-thought prompting. In contrast, GPT-3.5 Turbo demonstrated more flexibility in generating outputs with less specific guidance.

Chapter 2

Related Work

2.1 DEPTA

In this project, we apply the DEPTA method to organize web page content into clear, structured records. This method, based on the research in "Web Data Extraction Based on Visual Information and Partial Tree Alignment" by Fan et al. [1], uses the layout and visual elements of web pages to identify and group related data. Essentially, it finds parts of a web page that look similar (like sections listing professors' names and details) and groups them together. This makes it easier to extract specific information, such as faculty names and research interests, from complex web pages.

2.2 Prompt engineering

The project's approach to prompt engineering for Large Language Models (LLMs) aligns with principles detailed in Microsoft Azure's advanced prompt engineering guide [2]. By strategically crafting prompts, we directed LLMs to produce context-specific responses, a technique crucial for accurate web data extraction. This involved clear, structured instructions within prompts and a methodical breakdown of tasks. The project also incorporated grounding of text, as highlighted by Andrej Karpathy in his talk at Microsoft Build 2023 [3].

Grounding refers to referencing specific text within a prompt, leveraging the LLM’s memory for focused processing of web content. Such tailored prompts significantly enhanced the LLMs’ capability to parse and extract structured data from complex web pages and to minimize hallucinating.

2.3 Chain of thought untruthful explanations

The project also incorporates chain-of-thought (CoT) prompting into a web data extraction system, as discussed in "Language Models Don’t Always Say What They Think: Unfaithful Explanations in Chain-of-Thought Prompting," [4] enhances the Large Language Models’ (LLMs) ability to process and structure complex web data. This method, which involves LLMs articulating their reasoning step by step, is valuable for designing ordered instructions. However, the research acknowledges the risk of CoT explanations being systematically unfaithful, potentially misrepresenting the model’s true reasoning process.

2.4 Emotional language model prompts

In the benchmark suite, I experimented with the concept from "Large Language Models Understand and Can Be Enhanced by Emotional Stimuli," [5] applying emotional cues in specific test scenarios. This trial focused on enhancing the suite’s modules, particularly in extracting nuanced faculty data and comparing the responsiveness of different LLMs. While not used universally, this selective approach allowed for targeted assessment of models’ ability to process emotional stimuli alongside factual data extraction. The results provided valuable insights into each model’s comprehension and adaptability.

Chapter 3

Solutions & Techniques

3.1 Motivation

The primary motivation for my approach was to harness the capabilities of large language models (LLMs) for extracting specific information from web pages. Initial experiments with simpler web pages revealed that LLMs excel in analyzing large text blocks to discern and extract relevant details. However, challenges arose when LLMs had to process excessive information in one go, leading to missed or disorderly extraction. The key was to balance the amount of text provided, ensuring the model could effectively establish relationships without being overwhelmed.

3.2 Design

To address these challenges, I integrated the DEPTA method, aiming to organize web content into structured regions. DEPTA categorizes similar structured data, such as faculty records on a webpage, allowing for targeted LLM processing. This structured approach, however, exposed a limitation: processing all records in a region simultaneously could result in incomplete extraction. To mitigate this, I developed a strategy of initially identifying relevant regions containing professor names with a simple "yes" or "no" prompt. Subsequently,

the LLM would examine three records at a time, striking a balance between comprehensiveness and accuracy. The extraction focused on key data points: Professor Names, Positions, and Research Interests, with a fallback to output 'null' for missing information.

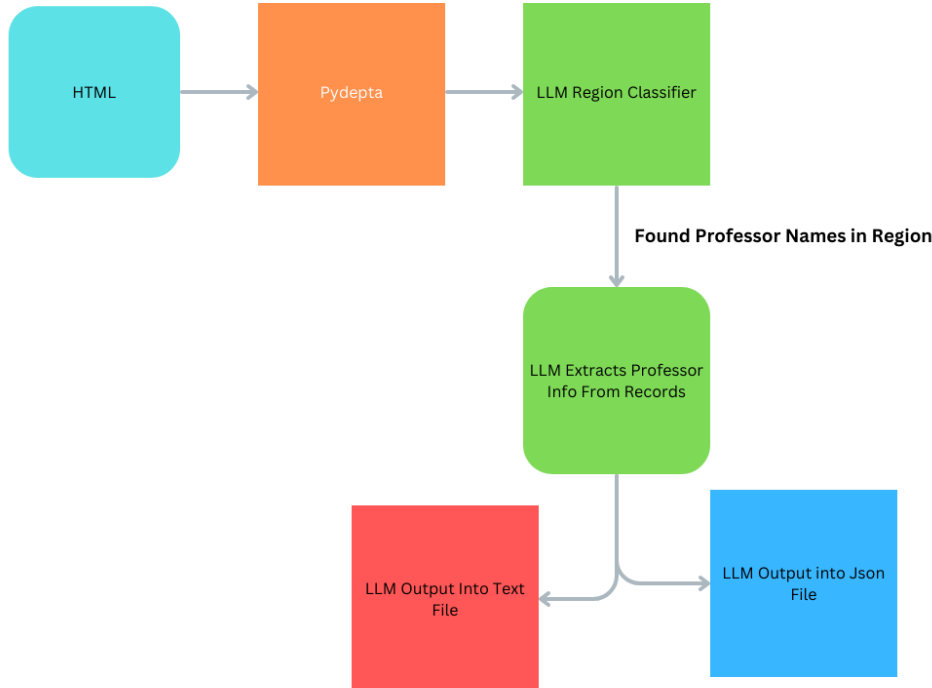


Figure 3.1: Illustration of the DEPTA method integration and data processing workflow.

3.3 Framework

The framework involved careful prompt engineering to guide LLMs effectively. Prompts were broken down into steps, each accompanied by an example to clarify the expected output. This approach was important when aiming for JSON formatted outputs, often requiring iterative refinement. Additionally, prompts included directives to prevent common pitfalls, such as avoiding the generation of unrelated content or use of inappropriate pronouns in specific contexts. Recognizing the occasional inaccuracies in LLM-generated JSON structures, future iterations of the program will focus on parsing LLM responses and manually constructing JSON objects for reliability.

Chapter 4

Experiments

The evaluation of this project primarily relies on qualitative analysis, complemented by a quantitative grading scale. This scale, detailed later in the experiments section, assesses the accuracy of LLM extractions from faculty pages of UIUC, MIT, and CMU.

4.1 Classifying DEPTA regions that contain professor names

What we are trying to classify here is to have the large language model to try and find Professor names inside a region so we know if we should extract information from it or to skip the region. In the prompt the region is the DEPTA region we want to classify.

4.1.1 Classification prompt

If the given list has at least one Professor's name, output '

- Yes' or 'No'. To determine this, examine each item in the
- list and identify if it mentions a Professor. A
- Professor's name is likely to be followed by academic
- titles or positions. Apply this criterion to each item in

→ the list. For example, the list provided contains names
→ followed by titles like Research Professor, Professor
→ Emeritus, and specific Professorial chairs, which
→ indicate these are Professors' names. Therefore, the
→ conclusion is 'Yes', the list contains at least one
→ Professor's name.- {region}

4.1.2 DEPTA classification example 1

A 'Yes' example DEPTA region from the UIUC faculty page:

```
['Han Zhao', None, 'Assistant Professor']  
['Craig Zilles', 'he/him/his', 'Professor and Severns Faculty Scholar']
```

This particular region contains 2 Professor names which are Han Zhao and Craig Zilles, so the output from the LLM should be 'Yes'.

4.1.3 DEPTA classification example 2

A 'No' example DEPTA region from the UIUC faculty page:

```
['Undergraduate Advising Office', 'Graduate Advising Office']  
['Faculty Support Contacts', 'Communications & Engagement Team']  
['Instructional Development Team', 'Business Office']
```

This particular region does not contain any Professor names so the output from the LLM should be 'No'.

4.1.4 DEPTA classification results

Even on harder regions this prompt and method for classifying regions has worked very well for the UIUC, MIT, and CMU web pages. I was able to correctly classify every region from these 3 web pages.

The Full Depta output for the web pages used can be found in the folder here:

```
https://github.com/FireBirdJZ/jason-zalewski-llm-web-extraction/tree/main/src/segmentation/pydepta/pydepta/output_depta
```

4.2 Extracting the professor's name, position, and research interests for each professor

The large language model now knows which Region has Professor names to extract from from the previous classification step. We now look at 3 records at a time inside a region to extract all of the Professor Names, Positions, and Research Interests. The `combined_records_str` is the current 3 records inside a region we are looking at.

4.2.1 Professor information extraction prompt

Text to Process: {combined_records_str}+

Understanding the Task:

The task is to create JSON objects for each professor using the

→ provided text data. Each JSON object should include keys

→ for 'name', 'position', and 'research_interests'.

Analyzing the Text:

The provided text data contains details about professors in

→ list format. The first element is the professor's name,

- ↪ the second element (if present) is their position, and
- ↪ the third element (if present) is their research
- ↪ interests.

Note: Research interests are often specific and technical, like

- ↪ 'Computer Architecture', 'Multicore Processors & Cloud
- ↪ Computing', or 'Artificial Intelligence + Machine
- ↪ Learning'. Ensure to capture these accurately.

Handling Missing Information:

If any of these elements are missing (position or research

- ↪ interests), the value should be set to 'null'.

Watch Out For:

pronouns such as he, they, or her. Don't add these anywhere to

- ↪ the JSON object.

Step-by-Step Process with Examples:

Step 1: Extracting Names

Extract the first element from each list for the name.

Example:

Given ['Devin H. Bailey', None, 'Associate Professor'], the '

- ↪ name' key will be "Devin H. Bailey".

Given ['Jeff Hamming', 'Armen Wolak (1965) Professor', '

- ↪ jham@csail.mit.edu', '(617) 103-9122', 'Office: 12-345',

- 'Algorithms', 'AI for Healthcare', 'Game Theory'] the '
- name' key will be Jeff Hamming. Don't get confused by
- professor positions that look like names such as Armen
- Wolak.

Step 2: Extracting Positions

Extract the second element for the position, using 'null' if it

- is missing.

Example:

Given ['Devin H. Bailey', None, 'Associate Professor'], the '

- position' key will be 'Associate Professor'.

Given ['Jeff Hamming', 'Armen Wolak (1965) Professor', '

- jham@csail.mit.edu', '(617) 103-9122', 'Office: 12-345',
- 'Algorithms', 'AI for Healthcare', 'Game Theory'] the '
- position' key will be 'Armen Wolak (1965) Professor'.

Step 3: Extracting Research Interests

Extract the third element for research interests, using 'null'

- if it is missing. Ensure to capture all available
- information, resorting to 'null' only when certain
- information is genuinely missing.

Example:

Given ['Devin H. Bailey', None, 'Associate Professor'], the '

- research_interests' key will be 'null' since the research
- interests are missing.

Given ['Jeff Hamming', 'Armen Wolak (1965) Professor', '

- jham@csail.mit.edu', '(617) 103-9122', 'Office: 12-345',

→ 'Algorithms', 'AI for Healthcare', 'Game Theory'] the '
→ research_interests' key will be 'Algorithms, AI for
→ Healthcare, Game Theory'.

Creating Individual JSON Objects:

For each professor, create a JSON object (dictionary in Python)

→ with the keys 'name', 'position', and '
→ research_interests'.

Ensure to replace missing elements with 'null'.

Building the Final JSON Output:

Combine these individual JSON objects into a single JSON array.

The final output should be a clean JSON array without any

→ additional nesting.

Final Example:

For 'Devin H. Bailey', the JSON object will be:

```
{  
  "name": "Devin H. Bailey",  
  "position": Associate Professor,  
  "research_interests": "null"  
}
```

For 'Jeff Hamming', the JSON object will be:

```
{  
  "name": "Jeff Hamming",  
  "position": "Armen Wolak (1965) Professor",
```

```
    "research_interests": "Algorithms , AI for Healthcare , Game  
    ↪ Theory"  
}
```

Continue this process for Rest of Professors , following the
↪ same steps. Don't write the Final Example JSON Object.

4.2.2 Professor information extraction grading explanation

For evaluating extraction accuracy, each JSON object (each representing one professor) is given a score out of 3 points: 1 point for correctly identifying the name, 1 point for accurately finding the professor's position, and 1 point for correctly listing the professor's research interests. Any unnecessary objects that shouldn't have been created in the output will also be counted separately, but not perfectly generated JSON, added commas, or explanations of chain of thought prompts will not be counted.

4.2.3 Professor information extraction example 1

For 3 CMU DEPTA records:

```
['Acar, Umut', 'Associate Professor ', '9101 Gates and  
Hillman Centers(412) 268-6791', 'uacar@andrew.cmu.edu']  
['Ada, Anil', 'Associate Teaching Professor ', '6215 Gates and  
Hillman Centers(412) 268-3835', 'aada@andrew.cmu.edu']  
['Akoglu, Leman', 'Associate Professor, Affiliated Faculty ', '2118C  
Hamburg Hall(412) 268-3043', 'lakoglu@andrew.cmu.edu']
```

Listing 4.1: Model output in JSON format

```
1  [  
2    {  
3      "name": "Acar, Umut",  
4      "position": "Associate Professor",  
5      "research_interests": null  
6    },  
7    {  
8      "name": "Ada, Anil",  
9      "position": "Associate Teaching Professor",  
10     "research_interests": null  
11   },  
12   {  
13     "name": "Akoglu, Leman",  
14     "position": "Associate Professor, Affiliated Faculty",  
15     "research_interests": null  
16   }  
17 ]
```

This output would get 9 points(perfect score for 3 records worth 3 points each); all of the names match the given DEPTA records, the positions also match the DEPTA records, and the research interests are correctly labeled as null.

4.2.4 Professor information extraction example 2

For 3 UIUC DEPTA records:

```
['Gerald F. DeJong, II', None, 'Professor Emeritus']  
['Payam Delgosha', None, 'Research Assistant Professor']  
['Jana Diesner', None, 'Associate Professor, Information Sciences']
```

Listing 4.2: Model output in JSON format

```
1 [  
2   {  
3     "name": "Gerald F. DeJong, II",  
4     "position": "Professor Emeritus",  
5     "research_interests": "null"  
6   },  
7   {  
8     "name": "Payam Delgosha",  
9     "position": "Research Assistant Professor",  
10    "research_interests": "null"  
11  },  
12  {  
13    "name": "Jana Diesner",  
14    "position": "Associate Professor",  
15    "research_interests": "Information Sciences"  
16  }  
17 ]
```

This output would be worth 8 points; everything matches the Depta records except Jana Diesner’s research interests should be labeled “null” not “Information Sciences”.

4.2.5 Professor information extraction example 3

A very wrong answer For 3 different UIUC DEPTA records:

```
['Brian P. Bailey', None, 'Professor']
['Michael Bailey', None, 'Adjunct Professor']
['Arindam Banerjee', None, 'Founder Professor in Engineering']
```

Listing 4.3: Model output in JSON format

```
1 To create the JSON objects for each professor using the
    ↪ provided text data, we will follow the step-by-step
    ↪ process outlined in the given text. Here is the
    ↪ implementation in Python:
2
3 '''python
4 import json
5
6 # List of professors
7 professors = [
8     ['Devin H. Bailey', None, 'Associate Professor'],
9     ['Jeff Hamming', 'Armen Wolak (1965) Professor', '
    ↪ jham@csail.mit.edu', '(617) 103-9122', 'Office: 12-34
    ↪ 5', 'Algorithms', 'AI for Healthcare', 'Game Theory']
    ↪ ,
10     # Add more professors here...
11 ]
12
13 # Initialize an empty list to store the JSON objects
14 json_objects = []
```

```

15
16 # Process each professor's information
17 for professor in professors:
18     # Extract name
19     name = professor[0]
20
21     # Extract position or set to 'null' if missing
22     if len(professor) >= 2:
23         position = professor[1]
24     else:
25         position = None
26
27     # Extract research interests or set to 'null' if missing
28     if len(professor) >= 3:
29         research_interests = ', '.join(professor[2:])
30     else:
31         research_interests = None
32
33     # Create JSON object for the professor
34     json_object = {
35         'name': name,
36         'position': position,
37         'research_interests': research_interests
38     }
39
40     # Append JSON object to the list
41     json_objects.append(json_object)

```

```

42
43 # Convert the list of JSON objects to a JSON array
44 json_array = json.dumps(json_objects)
45
46 # Print the final JSON array
47 print(json_array)
48 '''
49
50 This code will process the provided text data and generate a
    ↪ JSON array containing the JSON objects for each professor
    ↪ . The JSON array will be printed as the final output.

```

This output would be worth 0 points. Instead of extracting the faculty information into JSON the language model tried to create a python program to solve the task instead. This also counted as 3 unnecessary generated objects as the output was useless and took up potential space for 3 Professors records that could have been extracted instead. This failed extraction could have potentially been prevented by putting a statement inside the prompt stating: “Do not create a python program to solve this task!”.

The outputs used in the scoring can be found in the files starting with the name v2.2 in the folder:

https://github.com/FireBirdJZ/jason-zalewski-llm-web-extraction/tree/main/src/segmentation/pydepta/pydepta/video_test_comparing_models

4.3 Professor information extraction scoring results

Uni Webpage	Score	Total Points	Unnecessary Objects
UIUC	635	681 (93%)	13
MIT	198	225 (88%)	2
CMU	129	150 (86%)	0

Table 4.1: Extraction Accuracy Scores for UIUC, MIT, and CMU

Note: For the CMU web page, -15 points were given for flipping First Names and Last Names(96% accuracy for the extraction if you ignore this).

4.4 Comparing the capabilities of LLMs Gpt3.5 and Beluga for information extraction

Due to the extensive length of prompts and responses, specific examples are not included in this document. However, all outputs from the experiments with GPT-3.5 and Beluga are accessible via the following link:

https://github.com/FireBirdJZ/jason-zalewski-llm-web-extraction/tree/main/src/segmentation/pydepta/pydepta/llm_benchmark_suite

The findings indicate that Beluga required more specific instructions, including chain-of-thought prompting, for effective task completion compared to GPT-3.5. Both models demonstrated difficulty in accurately counting the number of records in a region, with neither model correctly identifying 20 records. Additionally, incorporating emotional reinforcement into prompts did not enhance their counting performance.

Chapter 5

Future Work

5.1 Developing a generalized approach for programming with LLMs

The focus here would be on crafting a more universal method for integrating LLMs into information extraction applications. This approach would encompass creating templates or frameworks that can adapt to various models and scenarios, thereby reducing the need for extensive customizations for each unique case.

5.2 Enhancing predictability and handling uncertainty

One of the main challenges faced in this project was the unpredictable nature of LLM outputs. To address this, further research could explore strategies for managing uncertainty in LLM responses. Small changes in a prompt can lead to unexpected dramatic changes in what LLMs output. This might involve developing algorithms or techniques that can more accurately anticipate and handle the range of outputs that LLMs can generate. This could improve the robustness and reliability of applications using these models.

5.3 Implementing a verification and correction System

A system that can verify the accuracy of LLM outputs and suggest corrections could help improve the efficacy of the extraction process. This would involve not only identifying incorrect or irrelevant responses (such as the model writing a program instead of extracting information) but also suggesting modifications to prompts or procedures to prevent similar errors in future runs.

5.4 Exploring advanced prompt engineering

Continued exploration in the field of prompt engineering could yield more efficient ways to guide LLMs towards desired outputs. This includes experimenting with various prompt structures, lengths, and styles to determine the most effective ways to communicate tasks to LLMs.

5.5 Investigating context management techniques

Investigating Context Management Techniques: Given the varying context length capabilities of different models, it would be helpful to develop methods for effectively managing and utilizing context in LLM-based applications. This could involve techniques for dynamically adjusting the amount of context provided based on the model's limitations and the complexity of the task.

5.6 Enhancing model comparisons

Further research could involve more nuanced comparisons of various LLMs, particularly focusing on how different models handle the same prompts and tasks. This can provide deeper insights into the strengths and weaknesses for each model, guiding their optimal use.

Chapter 6

Summary

6.1 LLM capabilities for information extraction

My research deepened my understanding of LLMs, especially their powerful inference abilities. They proved highly effective in analyzing and extracting detailed information from web content. However, their unpredictability and varying performances across different models were major challenges.

6.2 Overcoming LLM limitations

To address the limitations of LLMs, I employed several strategies:

6.2.1 Chain-of-Thought prompting

This technique helped in guiding the models to produce more accurate and relevant outputs. The effectiveness of the prompts can be further increased by incorporating step-by-step examples of the task's solution and including cautionary statements. These additions aid the model in avoiding potential pitfalls and improve its overall performance in executing the task.

6.2.2 Integration of external tools

I used tools like DEPTA to structure web content, making it easier for LLMs to process and extract information.

6.2.3 Handling context length

Dealing with large web pages required creative solutions, like segmenting content or experimenting with vector databases, to manage the context length limitations of LLMs.

6.2.4 Model comparison insights

Comparing different models, such as Beluga and GPT-3.5 Turbo, offered insights that each model has its strengths and requires specific approaches. For example, Beluga needed more detailed prompts, while GPT-3.5 Turbo was more flexible with less specific input. In addition, both models need external tools to cover their weaknesses such as counting the amount of records in a region, something both models struggled to find the right answer to.

Bibliography

- [1] Y. Zhai and B. Liu, “Web data extraction based on partial tree alignment,” in *Proceedings of the 14th International Conference on World Wide Web - WWW '05*, 2005. [Online]. Available: <https://dl.acm.org/doi/10.1145/1060745.1060761>
- [2] E. Urban and suhridpalsule, “Prompt engineering techniques with azure openai - azure openai service,” 2023, accessed: 2023-12-16. [Online]. Available: <https://learn.microsoft.com/en-us/azure/ai-services/openai/concepts/advanced-prompt-engineering?pivots=programming-language-chat-completions>
- [3] A. Karpathy, “Highlights from 2023,” <https://build.microsoft.com/en-US/sessions/db3f4859-cd30-4445-a0cd-553c3304f8e2>, Microsoft Build, 2023, accessed: 2023-12-16.
- [4] M. Turpin, J. Michael, E. Perez, and S. R. Bowman, “Language models don’t always say what they think: Unfaithful explanations in chain-of-thought prompting,” 2023. [Online]. Available: <https://arxiv.org/abs/2305.04388>
- [5] C. Li, J. Wang, Y. Zhang, K. Zhu, W. Hou, J. Lian, F. Luo, Q. Yang, and X. Xie, “Large language models understand and can be enhanced by emotional stimuli,” 2023. [Online]. Available: <https://arxiv.org/abs/2307.11760>
- [6] A. Borzunov, D. Baranchuk, T. Dettmers, M. Ryabinin, Y. Belkada, A. Chumachenko, P. Samygin, and C. Raffel, “Petals: Collaborative inference and fine-tuning of large models,” 2023. [Online]. Available: <https://arxiv.org/abs/2209.01188>

- [7] Z. Chen, “zhijiachen/pydepta: A python implementation of depta,” <https://github.com/ZhijiaCHEN/pydepta/tree/master>, 2022, accessed: 2023-12-16.
- [8] N. Friedman, “Natbot,” <https://github.com/nat/natbot/blob/main/natbot.py>, 2022, accessed: 2023-12-16.
- [9] OpenAI, “Chatgpt,” AI Language Model, 2023, accessed: 2023-12-25.
- [10] “Langchain,” <https://www.langchain.com/>, accessed 25 Dec. 2023.
- [1] [2] [3] [4] [5] [6] [7] [8] [9] [10]