IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/44AI/2024/4441324/
Vol. 2024 (22), Article ID 4441324

Application of Large Language Models to Data Extraction from Text with an Unknown Structure*

Maciej HOJDA and Wojciech LORKIEWICZ

Faculty of Computer Science and Telecommunications,
Wrocław University of Science and Technology
Janiszewskiego 11/17, 50-372, Wrocław, Poland
ORCID: 0000-0003-3195-4862
ORCID: 0000-0002-4624-5180

Correspondence should be addressed to: Maciej HOJDA, maciej.hojda@pwr.edu.pl

* Presented at the 44th IBIMA International Conference, 27-28 November 2024 Granada, Spain

Abstract

We present the results of applying Large Language Models to extract meteorological data from weather forecasts provided in a variety of formats. Processing information expressed in natural language is a difficult task, even more so when the goal is the extraction of certain numerical values from sources with unknown (at the design) structure. We apply Large Language Models to this task, and we verify their usefulness when processing various data formats describing the forecast in the natural language and condensed tabular and/or numerical representations. All the data was sourced from real meteorological systems and the output was fixed XML structure. We show that all the models tested in the paper succeed, with varying degrees of efficiency, in extracting basic data from the source forecasts and in encoding extracted information into a predefined XML structure. Finally, we pinpoint main types of errors encountered in the transformation process.

Keywords: Text Processing, Data Extraction, Large Language Model, LLM

Introduction

Language processing is a major field of artificial intelligence that focuses on categorizing, correcting, generating and extracting text. This field of science and engineering alike has a long history, where the first language models were just sequences of words called n-grams (Bahl 1983) and the goal was to predict the next word. While the goal remained mainly unchanged, the field shifted to using artificial neural networks as the main method of modeling. Initially, recurrent neural networks were used for text generation but the solution to vanishing or exploding gradients was solved more recently, with the invention of the long short-term memory (Hochreiter 1997) and finally, with the introduction of the transformer networks (Vasvani 2017). Using graphical processing units as the main computational resource, transformer networks quickly grew in popularity for all, the scientists, the engineers and the hobbyists. It has since became easier to parallelize both learning and inference using neural networks with learnable parameters ranging in the billions. This unprecedented number resulted in the term Large Language Models (**LLM**s) being coined and widely used (Zhao 2024). Since their introduction, LLMs have quickly grown in popularity due to their apparent ability to efficiently solve varying types of text processing tasks.

Cite this Article as: Maciej HOJDA and Wojciech LORKIEWICZ, Vol. 2024 (22) "Application of Large Language Models to Data Extraction from Text with an Unknown Structure " Communications of International Proceedings, Vol. 2024 (22), Article ID 4441324, https://doi.org/10.5171/2024.4441324

The benefits of LLMs have become widespread thanks to the introduction of publicly available tools, mainly chatbots such as ChatGPT (Gill and Kaur 2023) which was initially based on the GPT-3 model with over 175 billion of parameters (Brown 2020). More models soon followed (Rae 2021, Smith 2022, Chowdhery 2022, Touvron 2023a, Touvron 2023b, Abhimanyu, 2024), some of them freely available (open source) and typically to be found on the Hugging Face web portal (Huggingface 2024) either for online inference or for download, for offline use.

In this paper we consider the common problem of extracting data from a given set of external data sources. Most current information systems require that the underlying sources of data utilize a representation that is well-defined and geared towards machines, such as JSON or XML formats, with proper handling mechanism (Application programming interface). However, a plethora of available data is still far from being well structured and organized and is typically represented in a human-readable format like natural language or a mixture of natural language and auxiliary tabular/listing forms. Such data is hard to process using traditional methods and is often left outside of the scope of automated processing. Developments in LLMs allow for a reintroduction of such data sources into modern data pipelines.

The main goal of this paper is to establish the usefulness of LLMs in parsing texts with structures that are not known a-priori. This situation makes it difficult to apply a fixed extraction algorithm, however – as will become apparent – it is a task that LLMs can perform reasonably well. Following this introduction, in the next section we present the main models that will be used for extraction as well as our motivation for picking them. Then we formulate the main problem of the paper in another chapter that is followed by the design and summary of empirical evaluations that we performed. The paper ends with a short concluding section.

Selection of LLMs

As explained previously, in this paper we focus on solutions that can be run locally without the need to forward the data to an external computing provider. There are several reasons for that. Firstly, this makes the process safer in the sense that it eliminates potential data-breaches in transfer and at the provider of the computational infrastructure. This is especially important for cases where the data contains PII (personally identifiable information) or is simply not meant for sharing (such as volatile company data). Secondly, this simplifies the process of adjusting the LLM when needed. Adjusting is understood here as prompt design or fine-tuning (learning) for a downstream task. Moreover, this approach allows for an on-demand generation of responses to the user with little worry about external downtimes.

There are several freely available tools to run LLMs locally, in a chat-like mode (oobabooga 2022), however we prepare our own dedicated python scripts instead. This significantly streamlines the evaluation process. We use the *llama-cpp* library or more specifically, the *llama-cpp-python* bindings which provide the functions necessary to run LLMs in quantized *gguf* format (Gerganov 2024). Quantization is the method of reducing the size of network parameters which permits the model to run faster and be less demanding on hardware.

We make our choice of the models for evaluation based on their popularity and already verified efficiency. Since its introduction in 2023, the Llama family of open-source models has gained warranted popularity, performing comparatively or better than many closed source models (Touvron 2023a, Touvron 2023b). Its third iteration, Llama 3, has become available in the early 2024 (Abhimanyu, 2024) and we use the 3.1 edition of the model. The model is the *instruct* version so it was fine tuned for instruction following. Specifically, we use the gguf quantized versions of the 8B (8 billions of parameters) and 70B: Llama-3.1-8B-Instruct, Llama-3.1-70B-Instruct. The quants are Q5_K_S in both cases. Those models are small enough to be run and fine tuned on consumer hardware but still provide coherent text for queries in general.

Problem Formulation and Data Used

In this paper we consider the problem of extracting meteorological data from selected weather forecasts. Utilized forecasts originate from several services, which follow different data representations patterns. The main goal is to establish the usefulness of LLMs in parsing textual data from semi-structured and unstructured data sources, where representation pattern is not known a-priori.

We focus on data extracted from two popular service providers: 1) www.weather.gov and 2) weather.maniac.com. Both services provide (among other) textual descriptions of weather with varying information (temperature, wind, precipitation etc.). Excerpts of data are presented in Figures 1 and 2. Tested texts can vary, and the type of their structure is not known a-priori. This makes it difficult to process them automatically using traditional methodology. We expect that LLMs, which are trained on widely varying texts, are capable of overcoming this difficulty and can extract the data successfully.

We consider the following scenario: the LLM is asked to extract the meteorological data from the source forecast and encode it in a predefined XML form. The process is completely unsupervised and the LLM must rely on its own capabilities to extract the data and represent it in the output format. We enforce a simple XML format as required for the later ends of batch data processing data pipelines. We provide that format to the LLM as a part of the query. We also provide an example of properly filled XML fields as to ensure that the results are more uniform and easier to parse and validate.

We have extracted and manually processed a total of 30 examples: 20 from 1) with 10 samples each and 10 from 2). We use those examples in the experiments that follow.

```
MDZ021-282000-
Dorchester-
Including the city of Cambridge
801 AM EDT Mon Oct 28 2024

Date             Mon 10/28/24            Tue 10/29/24            Wed 10/30/24
EDT 3hrly     05 08 11 14 17 20 23 02 05 08 11 14 17 20 23 02 05 08 11 14 17 20
UTC 3hrly     09 12 15 18 21 00 03 06 09 12 15 18 21 00 03 06 09 12 15 18 21 00

Max/Min              62 64 65    38 43 52    66 69 71    51 56 62    71 76 77
Temp              57 63 62 53 50 48 46 45 59 66 67 62 59 58 57 58 68 74 74 68
Dewpt             33 31 34 40 42 43 44 45 50 50 50 53 54 55 56 56 59 60 59 60
RH                40 30 35 61 74 83 93100 72 56 54 72 83 90 96 93 73 62 59 76
Wind dir           E SE SE SE SE SE  E  E SE  S  S  S  S  S SW SW SW SW  S
Wind spd           3  4  5  4  3  3  3  3  5  8  6  5  5  5  5  6  9 11  9  8
Wind gust                16 14 14                18 18 18 18 18          19
Clouds            CL FW FW FW FW FW SC SC FW SC FW FW FW SC SC SC FW FW FW FW
PoP 12hr                   0          0          0          0          0
QPF 12hr                   0          0          0          0          0
Snow 12hr             00-00      00-00      00-00
Freeze
```

Figure 1. Excerpt of weather data from www.weather.gov .

```
NWS Forecast for: 20 Miles SSE Butte AK
Issued by: National Weather Service Anchorage, AK
Last Update: 4:45 am AKDT Oct 28, 2024

Winter Weather Advisory

Overnight: A chance of snow. Cloudy, with a steady temperature around 25. South wind around 10 mph.
Chance of precipitation is 50%.

Monday: Snow, mainly after 7am. High near 26. Southeast wind 15 to 25 mph. Chance of precipitation is
90%. New snow accumulation of around an inch possible.


Monday Night: Snow. Low around 24. Southeast wind 30 to 35 mph decreasing to 20 to 25 mph after
midnight. Chance of precipitation is 100%. New snow accumulation of 3 to 5 inches possible.
```

Figure 2. Excerpt of weather data from weather.maniac.com .

Empirical evaluation

Following the Llama 3.1 Instruct prompt style, we have used system and user prompts as seen in Figure 3. Relevant computation parameters are *context size 4096*, *top_k 10*, *max new tokens 2048*. Each used data file contains semi-structured textual description of weather conditions over several days. Excerpts from sample data files are given in Figure 1 and 2. We run tests twice for each data file, that gives us 60 results in total. Excerpts from sample outputs are given in Figure 4.

In general, tested models were capable of providing an output file that resembled the format given in the prompt. In several cases – 2 for the 8B model and (surprisingly) 3 for the 70B model – the output consisted of only one entry, which can be considered a fatal error. However, querying the model a second time did result in the correct format being outputted. This underlines the fact that LLMs are probabilistic at their core and some erroneous output can be mitigated simply by repeating the query. A fully automated tool, however, should be able to do so without user verification or input. The exact format of the output XML file also varied but that can perhaps be blamed on the prompt.

Another type of a common mistake was a result of the ill-fitting format for the data processed – in some cases the temperature or the wind speed were provided as ranges. In those cases, the output consisted either of an average or a dual entry was made. However, in some cases one of the outputs was suppressed entirely. Unfortunately, there were also mistakes that were clearly a result on LLMs weak ability to process data formatted visually, such as the data in Fig. 1. Columns were often misaligned such as can be seen in Fig. 4 (left). This type of mistakes seems difficult to correct for a purely text-based language model. Either mixed models (vision and text) could be applied to such cases or a *n*-shot learning procedure would help to solve the problem. As a reminder *n*-shot learning consists of providing examples of correct processing (*n* of them) within the prompt.

```
System prompt: You are a helpful assistant.

User prompt: Please extract weather data from a given text. Return the results in xml form with fields:
location, date, time, temperature, precipitation, wind speed, wind direction.

Location: provide full name.
Date: in format YYYY-MM-DD.
Time: in format HH:MM.
Temperature: in format XX degrees.
Precipitation: in format XX %.
Wind speed: in format XX mph.
Wind direction: in format XX (example NW or SW).

A single xml entry should have the following format (example):

<data>
        <location>Sweden</location>
        <date>2023-10-15</date>
        <time>15:30</time>
        <temperature>30 degrees</temperature>
        <precipitation>45%</precipitation>
        <speed>50 mph</speed>
        <direction>NW</direction>
</data>

Fill with "unknown" if a value is unknown. Return only the table and nothing else. The text is as
follows.

{text}
```

Figure 3. System and user prompts.

```
<data>
        <location>Dorchester</location>
        <date>2024-10-28</date>
        <time>08:00</time>
        <temperature>57 degrees</temperature>
        <precipitation>0%</precipitation>
        <speed>3 mph</speed>
        <direction>E</direction>
</data>
<data>
        <location>Dorchester</location>
        <date>2024-10-28</date>
        <time>11:00</time>
        <temperature>63 degrees</temperature>
        <precipitation>0%</precipitation>
        <speed>5 mph</speed>
        <direction>SE</direction>
</data>
```

```
<forecast>
    <data>
        <location>20 Miles SSE Butte AK</location>
        <date>2024-10-28</date>
        <time>04:45</time>
        <temperature>25 degrees</temperature>
        <precipitation>50%</precipitation>
        <speed>10 mph</speed>
        <direction>South</direction>
    </data>
    <data>
        <location>20 Miles SSE Butte AK</location>
        <date>2024-10-28</date>
        <time>07:00</time>
        <temperature>26 degrees</temperature>
        <precipitation>90%</precipitation>
        <speed>22.5 mph</speed>
        <direction>Southeast</direction>
    </data>
```

Figure 4. Output excerpts corresponding to Fig. 1 (left) and Fig. 2 (right).

Moreover, a common situation included a weather forecast stating only a low or only a high temperature value for a particular day. In all such cases the model anchored on this numerical value, as the forecasted temperature – missing the fact that it was only lowest or highest temperature point. Similarly, in case of wind speed, the model neglected to provide additional description, like "with gusts as high as X". It seems that the task to enforce a particular XML format in the output tunneled the focus of LLM towards picking only one of the available forecasted values. Unfortunately, such a shortcut resulted in several inconsistencies.

It turns out that establishing precipitation data is a complex task for LLM. In most cases, when the input data contains direct reference to numerical representation of precipitation chance (typically %), the model manages to correctly identify and represent it. However, in a case where only textual description containing words like "rain" or "high/low chance of rain" is available the model fails to identify and represent precipitation data. Most commonly unknown value is provided by the model in such a case. Several times, despite direct reference to "rain" in a particular forecast the model indicated 0% of precipitation in the output. It seems like handling assertions, textual descriptions of chances and converting them to an arbitrary estimation of chances of rain possess a significant difficulty to LLMs.

Finally, perhaps the most aggravating issue noted from the output is the LLMs weak ability to deduce about time. If the time (and date) was stated clearly, then the models have no problem putting them in the output file. However, if time is expressed in a convoluted, relative way, then the entries in the output can become temporally misaligned or altogether omitted. It is, however, well known that LLMs operate poorly on complex temporal data and need further training to be able to do so (Bhuwan 2022, Xiong 2024).

As another conclusion from the research that was carried out, we can note that LLMs struggle when directly used to extract data and enforce a particular machine-friendly format. As such LLMs are still not a silver bullet solution in the area of automated data processing.

Conclusions

In this paper we have tested two Large Language Models for their ability to extract data from text with an unknown structure. Tested models performed reasonably well despite the difficulty of the semantic processing task that they were burdened with. However, there still remains a significant room for improvement. It was seen that the output generated by the models suffered from diverse structure that made it difficult if not impossible to evaluate the data in a strict, numerical manner. To alleviate this problem, we foresee that it will be necessary to either fine-tune the models, either directly or through application of LoRA (QloRA) or to prepare a specific grammar that will force the output into an expected form. Alternatively, another round of queries and answers can be used to put the data into an uniform structure.

References

- Abhimanyu D, et al (2024) "The Llama 3 Herd of Models" arXiv:2407.21783
- Bhuwan D., et al (2022) "Time-Aware Language Models as Temporal Knowledge Bases" Transactions of the Association for Computational Linguistics, 20, 257-273
- Brown T, et al (2020) "Language Models are Few-Shot Learners" arXiv:2005.14165
- Chowdhery A, et al (2022) "PaLM: Scaling Language Modeling with Pathways" arXiv:2204.02311
- Hochreiter S, Schmidhuber J (1997) "Long Short-term Memory" Neural Computation, vol. 9/8, 1735-1780
- Huggingface (2024) "Hugging Face" https://huggingface.co/ [accessed: 2024, October]
- Gerganov G (2024) "Llama cpp" https://github.com/ggerganov/llama.cpp [accessed: 2024, October]
- oobabooga (2022) "text-generation-webui, A Gradio web UI for Large Language Models. Supports transformers, GPTQ, llama.cpp (GGUF), Llama models" https://github.com/oobabooga/text-generation-webui [accessed: 2024, October]
- Rae J, et al (2021) "Scaling Language Models: Methods, Analysis & Insights from Training Gopher" arXiv:2112.11446

- Smith S, et al (2022) “Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model” arXiv:2201.11990
- Touvron H, et al (2023a) “LLaMA: Open and Efficient Foundation Language Models” arXiv:2302.13971
- Touvron H, et al (2023b) “Llama 2: Open Foundation and Fine-Tuned Chat Models” arXiv:2307.09288
- Vaswani A, et al (2017) “Attention Is All You Need”, 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 5998-6008; arXiv:1706.03762
- Xiong S. et al (2024) “Large Language Models Can Learn Temporal Reasoning” arXiv:2041.06853
- Zhao W, et al (2024) “A Survey of Largue Language Models” arXiv:2303.18223v14