Benchmarking Large Language Models for Zero-Shot Automated Information Extraction from Scientific Literature

Master thesis
by

Felix Karg

Institute of Theoretical Informatics

Reviewer:	T.T.-Prof. Dr. Pascal Friederich
Second Reviewer:	Prof. Dr. Jan Niehues
Advisor:	Dr. Tobias Schlöder
Begin:	2023-03-01
Submission:	2023-10-02

Acknowledgements

I want to thank everyone who gave me feedback or ideas during this work. This includes early discussions just as much as involvement in creation of this particular document, through reading over it or offering to do so.

Some of these, which I want to name, though in no particular order:

- Pascal Friederich
- Tobias Schlöder
- Theresa
- Pablo
- Jana
- Konstantin
- Peter

This work acknowledges support by the state of Baden-Württemberg through bwHPC.

Declaration of Authorship

I hereby declare that I have composed this thesis by myself and without any assistance other than the sources given in my list of works cited. This thesis has not been submitted in the past or is currently being submitted to any other examination institution. It has not been published. All direct quotes as well as indirect quotes which in phrasing or original idea have been taken from a different text (written or otherwise) have been marked as such clearly and in each single instance under a precise specification of the source.

I am aware that any false claim made here results in failing the examination.

Karlsruhe, 2nd October 2023

Felix Karg

Approved as examination copy:

Karlsruhe, 2nd October 2023

Contents

List of Figures

Abstract

A majority of materials science knowledge is contained in unstructered text scattered throughout the body of scientific literature, out of reach for increasingly capable but data-starved Machine Learning models that are being used more and more at every step of the materials creation process. Thus, the extraction of such information from unstructured scientific literature and conversion to machine-readable formats has become a grand challenge of Natural Language Processing. Recently, Large Language Models have gained prominence for their general capabilities across Natural Language Processing tasks, including Named Entity Recognition. This work demonstrates that automated information extraction from materials science literature is possible with high accuracy using models of only 13 billion parameters and without fine-tuning. In fact, 13 billion parameter sized variants from both LLaMa and LLaMa 2 achieved an accuracy of 95 to 98% for the extraction of temperature and time information from synthesis paragraphs on the creation of Metal Organic Frameworks. Additionally, the 7 billion parameter sized Falcon achieved an accuracy of 79% on the extraction of solvent information on the same unstructured scientific literature. The smaller size of these models, their open-access availability, and no necessity for additional fine-tuning enables the usage of most consumer hardware, making these capabilities far more accessible than previously expected.

Zusammenfassung

Ein Großteil von materialwissenschaftlichem Wissen ist verteilt über unzählige wissenschaftlichen Arbeiten, wo es unerreichbar für besser werdende, aber datenhungrige Machine Learning Modelle ist, die mehr und mehr in allen Schritten der Herstellung neuer Materialien beteiligt sind. Darum ist die Extrahierung solcher Informationen aus unstrukturierten wissenschaftlichen Artikeln und Konvertierung zu maschinenlesbaren Formaten eine große Herausforderung der Verarbeitung natürlicher Sprache geworden. Zuletzt haben große Sprachmodelle Bekanntheit für ihre allgemeinen Fähigkeiten in der Verarbeitung natürlicher Sprache erlangt, inklusive der Erkennung benannter Entitäten. Diese Arbeit zeigt, dass automatisches Extrahieren aus Arbeiten der Materialwissenschaft mit hoher Genauigkeit möglich ist, sogar von Modellen mit nur 13 Milliarden Parametern und ohne nachzutrainieren. Tatsächlich haben die 13 Milliarden Parameter Varianten von LLaMa und LLaMa 2 Genauigkeiten von 95 bis 98% bei der Extrahierung von Temperatur- und Zeitdauerinformation aus Syntheseabschnitten zur Herstellung von Metall-Organischen-Frameworks erreicht. Zusätzlich hat die 7 Milliarden Parameter große Variante von Falcon eine Genauigkeit von 79% bei der Extrahierung der Lösung in selbigen unstrukturierten wissenschaftlichen Artikeln erreicht. Die kleine Größe dieser Modelle, deren freie Verfügbarkeit, und keine Notwendigkeit für zusätzliches Trainieren ermöglicht die Nutzung mit Verbraucherhardware, was diese Fähigkeiten sehr viel zugänglicher macht, als zuvor angenommen.

1. Introduction

Unstructured scientific literature contains a vast amount of materials science knowledge, which is not provided in a more structured manner elsewhere. This places it squarely out of reach for further processing, which makes it difficult or impossible to build on a large part of prior work. Machine Learning (ML) models are increasingly used in screening steps for materials discovery and property prediction [1–3]. In general, more high-quality data improves the output quality of ML models considerably [4]. Currently, the amount of data accessible to train such models is limited. This is both from older or less known work, but also from recent work that does not publish or contribute to a database.

Over the last year, Large Language Models (LLMs) rose to public prominence since the launch of ChatGPT. Regardless of public perception, recent improvements across Natural Language Processing (NLP) benchmarks are undeniable [5, 6]. In this work, LLMs are used for information extraction of scientific text on synthesisizing Metal Organic Frameworks (MOFs). This work demonstrates that smaller open-access models can achieve high accuracy on information extraction tasks without fine-tuning.

1.1 Scientific Question

There are three main questions this work aims to answer:

1. Can we demonstrate high accuracy in zero-shot automated information extraction from scientific literature using open-access LLMs?
2. How do currently available open-access LLMs compare for this task?
3. How easy is it to fine-tune open-access LLMs for this task, and how much does fine-tuning increase the accuracy?

As part of this work, a highly flexible automated pipeline for the extraction of information from unstructured MOF synthesis paragraphs was created.

1.2 Structure of this Work

This work is built up in the following way: First, Chapter 2 provides background knowledge on NLP and its tasks, primarily in Section 2.1. Section 2.2 goes on to explain how Language Models (LMs) have become a powerful tool in this discipline.

Chapter 3 then provides a basic understanding of the LLMs used, by first defining basic and modern terminology in Section 3.1 before explaining training and fine-tuning in Section 3.2.

The approach chosen in this work will be discussed in more detail Chapter 4, where the implementation is described in Section 4.1, and the models used in this work are described in Section 4.2. Additionally, Section 4.4 describes the origin of the data used for this work.

Chapter 5 is fully dedicated to exploring the results of this work. A high accuracy on the extraction of temperature, duration, and solvent information from unstructured scientific text was achieved, in a zero-shot setting without fine-tuning. This is explained in more detail in Section 5.1, before Section 5.2 analyses some of the most frequent failure modes. Moreover, Section 5.3 provides excerpts of errors encountered during fine-tuning.

The conclusions from this work are drawn in Chapter 6, before Chapter 7 provides an outlook of possible further veins of inquiry to explore.

2. Background

This chapter gives a short introduction to the parts of Natural Language Processing necessary for this work, and highlights other relevant work.

First, Subsection 2.1.1 broadly defines the NLP task of Information Extraction. Subsection 2.1.2 extends this by defining Named Entity Recognition (NER) and how this work expands upon the usual definition of NER. Then, Subsection 2.2.1 highlights early approaches and solutions, Subsection 2.2.2 explains how LMs are being used for such tasks, and finally Subsection 2.2.3 references the usages of LLMs for this use-case.

2.1 Natural Language Processing

NLP is an interdisciplinary field of study between computer science and linguistics, and is primarily concerned with giving computers the ability to "understand" the contents of documents. This often requires contextual nuances to accurately extract information. For this work, primarily the task of Information Extraction explained in the next Subsection 2.1.1 and specifically the subtask NER explained in following Subsection 2.1.2 are of importance.

2.1.1 Information Extraction

Information Extraction is the NLP task of extracting structured (machine-readable) information from unstructured text. In most cases the extended goal of information extraction in the natural sciences is to create a database of the extracted information. Such databases are then used to train further models used to predict attributes or synthesis conditions [2], or to substantially improve automated experimentation [7]. In this work, no such database will be created, as the goal is instead to benchmark the accuracy of various models for such a task. One approach to information extraction is via NER.

2.1.2 Named Entity Recognition

NER seeks to locate and classify named entities (e.g. classify 'water' as solvent) mentioned in unstructured text into pre-defined categories [8] such as temperature, timeframe, or solvent.

The usual NER definition of an entity assumes *rigid designators* [9], i.e. that an entity only has one name to reference it. This work uses the usual definition of NER, but loosens the expectation of rigid designators in the following two ways:

- First, references to temperature can include phrases such as 'at room temperature' or 'in boiling water', and may not just be numbers followed by their unit.
- Second, while Chemistry has normative rules on how molecules are named, it is possible for one molecule to have multiple valid names. In NLP, this is sometimes referred to as the *coreference problem* [10].

2.2 Related Work

2.2.1 Rule-Based Entity Recognition

There have long been rule-based approaches for the recognition of named entities (e.g. temperature) for material science literature. ChemTagger [11], and others [12, 13] clearly demonstrated that systems based on regular expressions can accurately extract information in straightforward, well-defined situations. However, these tools tend to be highly specialized which makes them hard to adapt to new or more complex queries or circumstances.

2.2.2 Language Models for Information Extraction

NER is sometimes modelled as a sequence-to-sequence labeling problem [14, 15]. Zhao et al. fine-tuned a number of pretrained Bidirectional Encoder Representation from Transformers [5] (BERT) instances and achieved an F-Score of 85% [14], demonstrating both high precision and recall (and F-Score is $2 \cdot (precision \cdot recall)/(precision + recall)$, where *precision* is a metric for how many retrieved items are relevant, and *recall* is a metric for how many relevant items where retrieved.). This demonstrated the fundamental capability of LMs for this task. Most other work on NER using LMs has not been on materials science literature but on general purpose text [8].

In this work, NER is not modeled as a sequence-to-sequence labeling problem, which is why F-Score is not an applicable metric for performance. Additionally, a distinction has to be made between a masked language model such as BERT and all well-known LLMs which are causal language models. A causal language model predicts the likelihood of the next token (piece of text) based on a previous sequence of tokens, often called *input*, *prompt* or *context*. In contrast, a masked language model predicts attributes of, or tokens in potentially multiple masked locations as part of a sequence of tokens, taking the full context of all surrounding tokens into account. The output of a masked language model are properties or tokens for each masked location, whereas the outupt of a causal language model is a likelihood distribution for the next token in the sequence. Section 7.6 outlines how a comparison with modern masked language model architectures might look like. Additionally, Section 3.1 describes Language Models in more detail.

2.2.3 LLMs for Information Extraction

LLMs, usually in the form of a causal language model, are a recent phenomenon, and at the forefront of NLP research. There is not much prior work on NER using a LLM. A combined NER and Relation Extraction approach using GPT3 has been tried with some success [15]. The primary focus of Dunn et al. [15] is on relation extraction, which is why F-Scores for the NER task vary between 0.4 and 0.95. This work differs on two counts from [15]: 1) a benchmark and comparison of multiple open-access LLMs is done and 2) the entire focus of this work is on the extended NER task.

A more extensive introduction to LLMs is given in the later Subsection 3.1.4.

3. Methods

This chapter gives an overview of how Language Models work in general. For that, basic and modern terminology and concepts on LM are explained in Section 3.1, before Section 3.2 explains further how training and fine-tuning of these models broadly works.

3.1 Language Model Basics

This Section aims to provide an overview of the language models used in this work as well as a fundamental understanding of the underlying concepts and terminology. First, some basic terminology on LMs is explained in Subsection 3.1.1 before fundamentals of the original transformer architecture are being established in Subsection 3.1.2. Subsection 3.1.3 reintroduces the transformer architecture from a modern perspective, and highlights changes relative to the original architecture. Last but not least, Subsection 3.1.4 provides information on the scaling up and development of modern LLMs.

3.1.1 Basic Terminology

- A *token* is a short piece text, and can be anything from an individual character up to entire words. On average, a token is 3-4 letters.
- The *context length* of a model is the amount of tokens it can process as input concurrently to provide an output.
- A *single-shot* or *multi-shot* setting describes an evaluation setting in which a LLM is being provided with one or multiple examples of the task to fulfill as part of the input. In early LLMs, this increased task accuracy considerably [16].
- *Zero-shot* is, in contrast to single-shot or multi-shot approaches, not providing any previous examples or demonstrations of the task. This approach fully relies on the transfer learning capabilities of the model, which was only subjected to non-task-specific training beforehand. This work tests models in such a setting.

3.1.2 The Transformer Architecture

All modern Language Model are based on what was introduced as the transformer architecture [17] by Google in 2017. This architecture was originally designed for translation, with an encoder and a decoder part (See Figure 3.1 for more details). The text to translate is first encoded to the embedding space by the encoder, after which the decoder will autoregressively (only one token for each full forward pass, where the most recent one will be appended to the output) generate the output, token by token.

This new transformer architecture quickly established itself by outperforming other architectures available at the time with only a fraction of the training cost. An encoder-only transformer architecture, specifically BERT, set a new State of the Art (state-of-the-art) for all NLP benchmarks established at the time [5].

3.1.3 A Modern Transformer Architecture

There have been various attempts at improving the transformer architecture, some of them more successful than others [18–25]. A number of these variations have been adapted, and are used when setting up a new transformer model with few exceptions. Refer to Figure 3.2 for a graphic representation and additional description.

Early transformer architectures used Rectified Linier Unit (ReLU) as an activation function. By comparing different activation functions, it was found that Swish Gated Linear Unit [18] (SwiGLU) empirically works best in most situations. Surprisingly, even though dropout established itself as extremely robust technique against overfitting and forcing generalization [26], few modern LLM architectures make use of it. This may be due to a lack of available high-quality training data.

An even better overall performance was achieved when using the more sophisticated Rotary Positional Encoding [19] (RoPE) instead of previous sinusoidal encoding for positions. Normalizing with RMSNorm as LayerNorm [27], and using it before instead of after each layer resulted in less erratic training. In addition, a further marginally improved performance and parameter count reduction could be achieved by grouping attention query heads together in Grouped Query Attention [20] (GQA).

Because the computational complexity based on context length for the original transformer architecture is $O(n^2)$, numerous approaches attempted to improve that [21, 28–31]. Noteworthy and practically used are in particular sparse attention [28] (in $O(n\sqrt{n})$), and FlashAttention [31]. FlashAttention reintroduced quadratic complexity, but resolved bottlenecks in memory layout and IO throughput, achieving substantial speedups for practical sizes of context length. Attention with linear context-length complexity has been demonstrated [29, 30], but is not without drawbacks and thus not widely used.

This modern architecture as roughly described here and visualized in Figure 3.2, is used with small variations by every recent LLMs [32], and in particular those used for this work, introduced later in Section 4.2. However, models before LLaMa 2 do not use GQA.

For a more comprehensive overview on the modern transformer architecture and modern LLMs, refer to [32].

3.1.4 Large Language Models

GPT2 [33] is a decoder-only LM mostly based on the original transformer architecture, scaled up more than previous models. Compared to other models at the time, GPT2 had up to 15x more parameters than BERT – 1.5 billion. GPT2 indicated that bigger LMs are more capable in general, only constrained by computational resources. Models after that seem to only confirm this, and various scaling laws with diminishing returns have been observed [34, 35]. Models with multiple billion parameters became generally referred to as Large Language Models (LLMs).

GPT3 was the first such LLM, with 176 billion parameters, introduced by OpenAI in 2020 [16]. In the months after, many more models with similar capabilities from different organizations followed. The most well-known models of this wave were BLOOM, OPT and PaLM [36].

The most recent and most capable generation of LLMs were introduced starting early 2023, after the release of ChatGPT sparked worldwide interest in LLMs. This prompted many organizations to research and advance the capabilities of LLMs. This renewed interest enabled fast progress in many different organizations, which culminated in dozens of advances (read more on them in Subsection 3.1.3).

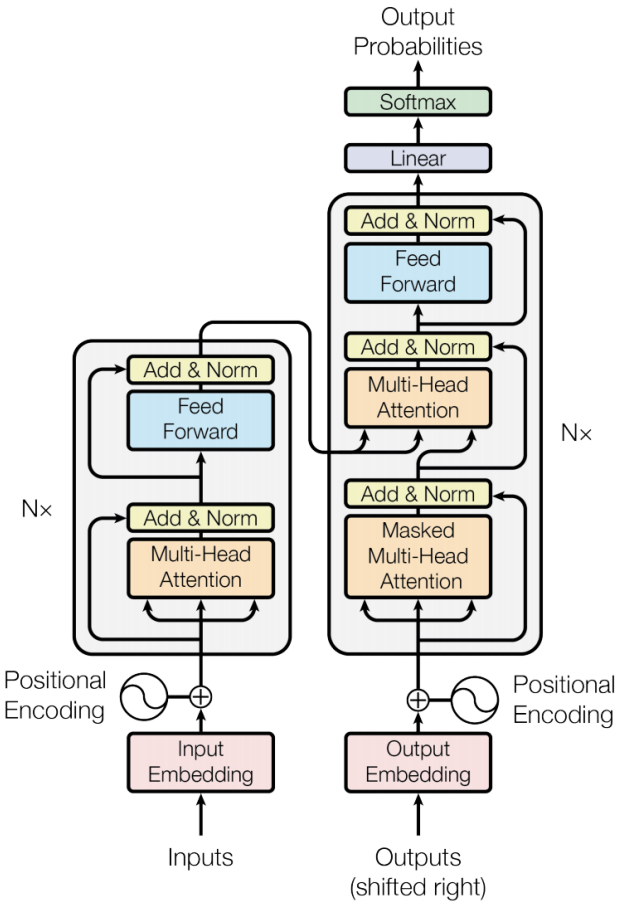

Figure 3.1: Original Transformer Architecture. Here, the left part is called 'encoder', and the right part 'decoder'. In the encoder, tokens are are being encoded to their representation in embedding space and added to a positional encoding, before being passed through alternating layers of self-attention and Multi-Layer Perceptron (MLP) feed-forward layers. The activation function is a ReLU, and the architecture makes use of residual connections and normalization after each layer. The decoder originally had an additional layer of cross-attention to the previously generated input embedding. Causal language models are exclusively autoregressive decoder-only models. Image Source: [17]

Earlier, Chinchilla [35] demonstrated that while achieving impressive capability, such large models tend to be substantially overparametrized and undertrained. Additionally, CoTR [37] demonstrated that existing models have not been properly utilized and are a lot more capable than previously thought.

Thus, newer models, while continuing to advance in capabilities, aren't necessarily larger. In fact, newer models tend to be substantially more capable while being considerably smaller. Such models include the open-access LLaMa and its well-known derivatives Stanford Alpaca and Vicuna, Falcon, as well as most recently LLaMa 2.

For more information on most of the aforementioned models, see the later Section 4.2.

3.2 Training Large Language Models

Most datasets for training ML models consist of two parts: *input* data and *target* or *label* data. In these cases, the model ought to learn the generalizing nonlinear function of providing the labels based on the input. This terminology continues to be used for training LLMs, even though there may be no input, or the label is just additional appended text.

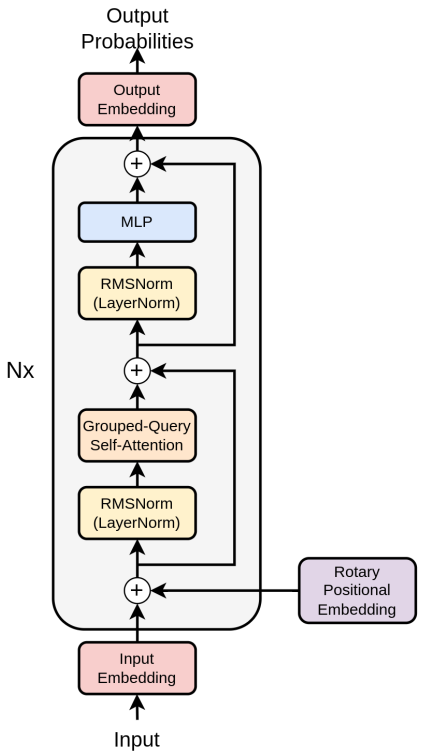

Figure 3.2: Example of a Modern Transformer Architecture. There are a number of differences when compared to the original architecture as seen previously in Figure 3.1: layers are normalizing the residual with RMSNorm instead of having a normalized residual. Multi-head attention was replaced with GQA, and sinusoidal positional embeddings with embeddings from RoPE *on each layer*. Additionally, activation functions in the MLP changed from ReLU to SwiGLU.

Training a Large Language Model heavily depends on the task the model is supposed to learn. First general pretraining is discussed in Subsection 3.2.1, before additional information on on fine-tuning a pretrained model is discussed in Subsection 3.2.2. A specialized version of that, fine-tuning on instructions, is then discussed in Subsection 3.2.3.

3.2.1 Pretraining

The objective of any causal language model is to predict the next token based on the current token sequence. Prediction of the next token only depends on previous tokens in the context. For such scenarios, the reward is modeled as the likelihood of predicting the correct token in the sequence. In most cases (and with few exceptions), the optimizer of choice is AdamW [38], tasked to minimize the Cross-Entropy loss of predicted tokens [32].

Fully training a new model for competitive capability is a massive undertaking requiring many thousands of GPU-hours [39, 40], with all the associated costs and carbon footprint that entails. In order to train a LLM, a high-quality dataset is also needed, many of which are publicly available [41]. More details on pretraining and nuances necessary for large scale distributed training can be found in [42].

3.2.2 Fine-Tuning

Fine-tuning exploits *transfer learning* by continuing to train a previously *pre-trained* model, usually with a lower training rate and on very specialized data [43]. This allows *transferring* previously learned features to accomplish a different task by slightly adapting the model but exploiting existing structures for feature detection. The main benefits include using

substantially less computational resources and training examples [43], both of which tend to be hard to acquire, depending on the task.

As such, fine-tuning and the resulting models are usually highly specialized to their task.

3.2.3 Fine-Tuning on Instructions

Pre-trained or task-specific fine-tuned models have learned context dependent token sequence likelihoods. This is useful for predicting the 'next' token on a wikipedia article or novel, but this does not make it easy to utilize the model in other ways.

This can be solved by instead predicting the most likely token sequence when prompted with instructions instead. Fine-tuning on instructions thus guides the output by shifting the learned context-dependent token sequence likelihood, providing more control over model outputs. This results in users having a preference for the outputs of smaller models, over substantially bigger models, when the smaller model has been instruction fine-tuned [44], as it makes the model more 'useful' in a naive sense. A model fine-tuned on a instruction dataset is commonly referred to as an *instruct*-variant. More details on instruction-based fine-tuning can be found in [42, 44].

4. Approach

This chapter describes all relevant decisions of the approach taken, as well as their reasons.

First, Section 4.1 broadly describes the implementation, as well as libraries and frameworks used. In Section 4.2 descriptions and references to the various models considered for this work can be found. Section 4.3 discusses how the models where prompted in detail, and why little effort and experimentation was done with different prompts. The dataset used, and how it was processed is described in Section 4.4, before it is depicted how the comparison for equality was done in Section 4.5.

4.1 Implementation

All source code for this work can be found at `https://github.com/fkarg/mthesis`, and a tag will mark the state at the time of submission.

It became appearent during literature research that a comparison between different models would be valuable, and that additional models can be expected to be released in time to be included in this work. Thus, almost all code ought to be model-agnostic. The HuggingFace `transformers` [45] library was chosen as a well-established framework providing abstractions to load, manipulate and train any deep learning architecture in a standardized format. Additionally, all open-access LLMs are available directly through the HuggingFace portal.

Dependencies

Most other dependencies used are either straightforward (`torch`, `einops`, `accelerate`, `bitsandbytes` to get the models to run) or common ecosystem choices (e.g. `typer` and `rich` for the cli interface; `pubchempy` to resolve and convert chemical compounds; etc).

Modularity

Proportionally speaking, the main module (with 36%), dataloader (with 23%), and unit conversion module (with 11%) have the highest Lines of Code (LOC) counts. Everything else is split in six more supporting modules.

4.2 Language Models Considered

The following sections describe the various models considered for the benchmark, and the reasons for or against their inclusion. Before that, Subsection 4.2.1 depicts simple criteria for model selection.

See Section 3.1 for an overview and broad categorization of the models mentioned here.

4.2.1 Criteria

First, the model weights need to be available. This is necessary to run the model in a self-hosted manner. Practically, this is a constraint towards open-source or at least open-access models (their license needs to allow for academic research).

Second, at this point dozens of LLMs have been trained, and weights made available. One model was trained purely as a marketing pitch to sell computing systems [46]. To vastly condense the number of models to consider, each base model ought to have demonstrated fundamental capability in other domains. Additionally, due to the nature of the task, it is valuable to include both base models and instruct-based derivatives for comparison.

Third, all else being equal, a smaller model is preferred. This is due to smaller models being less resource intensive to evaluate and fine-tune. Smaller variants of larger models additionally enable faster and less resource intensive iteration during development.

In summary, the following criteria where chosen for model selection in this work:

1. It is possible to get the full model weights.
2. The selected models ought to be decently capable causal language models.
3. Ceteris paribus, a smaller model is better.

4.2.2 OPT, BLOOM (Not Used)

OPT

The initial model set out for this work was OPT [47], a 175 billion parameter open-source LLM trained by Meta, with partially similar capability as GPT3. During early literature research, we encountered the similar but slightly more capable BLOOM.

BLOOM

BLOOM [48] is a 176 billion parameter open-source LLM trained by a cooperation of numerous organizations, spearheaded by HuggingFace and Google. When compared to OPT across NLP benchmarks, BLOOM appears to perform marginally better.

Reasons for Using Neither Model

The original plan for this work would use OPT as the only model. During early literature research, it seemed that BLOOM would be slightly more capable, which changed the intention to compare both. Not soon after, the smaller and seemingly much more capable LLaMa was released, which prompted the decision of creating a model-agnostic pipeline instead, focusing on LLaMa first. See the next Subsection 4.2.3 for more details on LLaMa.

4.2.3 LLaMa (Used)

LLaMa is a family of open-access LLMs provided by Meta with sizes ranging from 7 billion to 65 billion parameters, and capabilities comparable to, and sometimes beating state-of-the-art (including the substantially larger GPT3) at the time of release [39]. LLaMa can be seen as the first culmination of progress on LLMs up to this point, in one place.

LLaMa is not instruction fine-tuned. See Subsection 3.2.3 for more details on instruction fine-tuning. For instruction fine-tuned variants of LLaMa, see Subsection 4.2.4 on Stanford Alpaca or Subsection 4.2.5 on Vicuna.

4.2.4 Alpaca (Not Used)

The Stanford Alpaca Project [49] aims to build and share an instruction fine-tuned LLaMa model. Due to uncertainty with the LLaMa licence which this model is based on (it was fully released a mere two weeks after LLaMa was first announced), no model weights where released officially. Instead, all scripts and training data to fine-tune your own Stanford Alpaca based on existing LLaMa weights where provided. Fine-tuning on a large dataset becomes impractical for larger model variants due to rapidly increasing resource requirements. For this reason, it was decided against including Stanford Alpaca in the benchmark.

4.2.5 Vicuna (Used Partially)

Vicuna is a family of instruction fine-tuned LLaMa-variants, released by Large Model Systems Organization (LMSYS). It is built on top of the training recipe of Stanford Alpaca. However, not all weights of the corresponding LLaMa sizes are available. The largest LLaMa-model (65B) does not have a corresponding Vicuna derivative available. In a tournament format between different LLMs, Vicuna provided user-preferred answers more often than LLaMa and Stanford Alpaca [50], among others. Thus, before the release of Falcon and LLaMa 2, Vicuna was generally seen as the most capable instruct-based model, which is why it was included in this benchmark.

4.2.6 LLaMa 2 (Used)

Meta released LLaMa 2 [51] only a few months after the release of its predecessor. They introduced few fundamental changes when compared to LLaMa. The main differences include making use of GQA for the first time, and training on more tokens. For each size, LLaMa 2 was released in four versions: 1) the base model 2) a 'helpful'-variant trained with human-feedback 3) a 'chat' variant optimized for dialogue and 4) a combined 'helpful' and 'chat' variant.

4.2.7 Falcon (Used)

The Falcon [52] family of language models are created by the Abu Dhabi-based Technology Innovation Institute (TII). Since its release, Falcon is at the top of most benchmarks between open-access models (in each respective parameter size category) [52]. It appears to rival some of the most capable closed-access models such as PaLM in capability.

The better performance of Falcon for most tasks is assumed to mostly be the result of longer training and higher-quality data sets [52].

Recently, a new closed-access 180 billion parameter Falcon variant was announced [53]. Falcon-180B is not included in this benchmark.

4.2.8 GPT4 (Not Used)

GPT4 is the fourth generation Generative Pretrained Transformer (GPT) model from OpenAI [6]. It is the single most capable Language Model we currently know of. However, it is not open-source and only accessible through interfaces provided by OpenAI. Additionally, OpenAI continues to work on, change, and sometimes degrade the capabalities of GPT4 [54]. Even timestamp-versioned, 'unchanging' models have been claimed to measurably change in behaviour [55]. This makes it a hard target for comparison.

GPT4 does not fulfill the criteria of being open-access, and is thus not compared in this work.

4.2.9 Final List

In conclusion, we used the following models and sizes of the aforementioned:

- LLaMa 7B, 13B, 30B, 65B (See Subsection 4.2.3 for more details on the model)
- Vicuna 7B, 13B, 33B (See Subsection 4.2.5 for more details on the model)
- LLaMa 2 7B, 13B, 70B (See Subsection 4.2.6 for more details on the model)
- Falcon 7B, 40B (See Subsection 4.2.7 for more details on the model)
- Falcon-instruct 7B, 40B (See Subsection 4.2.7 for more details on the model)

4.3 Prompts

LLMs are capable of very generic tasks, based on the input they are asked to respond to. *Prompting* a model with a certain input gives more fine-grained control over the output and can provide additional structure, information, and suggestions for solving a specified task. This is particularly eminent in models fine-tuned for instruction-based or chat-based interaction.

In this work, the main effort was put towards creating a pipeline for fine-tuning all models for the specified NER task. This was due to the expectation that even small amounts of fine-tuning would be more effective in guiding model outputs than the best prompt could be. Results of fine-tuning attempts from this work can be found below in Section 5.3.

Structured Output

`guidance` [56], an approach and library originally developed by Microsoft research, allows more effective control over LLMs than traditional prompting or chaining does. In effect, `guidance` is a harness around a LLM, providing support for the model in generating structured information, and making use of additional output structures such as CoT [57] to result in higher-quality outputs.

For this work, specifically the library `jsonformer` [58] is used, which does not provide the full feature suite of the `guidance` library. At the time of deciding, `guidance` did not support models through the HuggingFace `transformers` [45] () library yet.

The schema used for guidance through `jsonformer` is illustrated in the following code examples, which show (1) a schema for guidance, (2) a full prompt as wells as (3) an example output, in Code Example 1, Code Example 2 and Code Example 3 respectively.

Code Example 1

The schema provided for the model to follow. Model output termination would happen after generation of a token for '`"`' for strings or '`,`' for numbers, or a number of other dedicated 'end of generation' tokens, e.g. `<EOS>`. See Code Example 3 for what a possible output of this schema might look like.

```
schema = {
    "type": "object",
    "properties": {
        "additive": {"type": "string"},
        "solvent": {"type": "string"},
        "temperature": {"type": "number"},
        "temperature_unit": {"type": "string"},
        "time": {"type": "number"},
        "time_unit": {"type": "string"},
    },
}
```

Code Example 2

Prompt used to generate output. `"{output}"` delineates where the model provides an answer. See Code Example 3 for what may be filled in.

```
prompt = """{paragraph}\nOutput result in the following JSON schema format:
\n{schema}\nResult: {output}"""
```

Code Example 3

Exemplary output based on the prompt shown in Code Example 2, and schema shown in Code Example 1.

```
output = {
    "additive": "acid",
    "solvent": "water",
    "temperature": 80,
    "temperature_unit": "C",
    "time": 24,
    "time_unit": "h",
}
```

4.4 Data Source

As data source, 778 synthesis paragraphs and their corresponding labels from the publicly accessible database SynMOF_M [2] where used. Each synthesis paragraph describes the creation procedure of a MOF. This work focused on the basic parameters `temperature`, `time` and `solvent`, though additional parameters could be added quickly to the schema discussed in the prior Section 4.3.

The labels from the SynMOF_M database where manually annotated in prior work [2]. All proportional results in the later Chapter 5 are based on the accuracy of over all 778 items. For training purposes, this dataset would have been split in dedicated datasets for test and training.

4.5 Criteria for Equality

This section defines the criteria for determining equality between a result from a LLM and the target label. The criteria for `temperature` and `time` and dealing with unit conversions is described in Subsection 4.5.1. Then, Subsection 4.5.2 detailes how compounds are compared.

4.5.1 Time and Temperature

In the dataset (See Section 4.4 for more details on the data source), all temperature information is encoded in degrees celsius, and all time information in hours. Without a field for the unit (See Section 4.3 for the prompts and structure used), models would use arbitrary units, often those used in the paragraph they are extracting from. Since the task is not accurate unit conversion, but information extraction, a field for `temperature` and `time` units was added.

Unit conversions for `temperature` and `time` happen automatically before comparison and convert to a unified format, degrees celsius and hours respectively. This ensures that durations of both '24h' and '1 day' are seen as equal, even though the strings are different.

4.5.2 Compounds

Instead of names of chemical compounds, the database (See Section 4.4 for more information on the data source) contains the `pubchempy-cid` (compound id) as the labels for solvents and additives (if applicable). Most compounds have multiple different synonymous names they are known by, e.g. 'water' has one `cid` (which is 962) and a list of 319 distinct strings it can be resolved from. Surprisingly, while the list of synonyms includes both 'distilled water' and 'H2O', it does not include 'distilled H2O', which is mentioned verbatim in eight of the 778 synthesis paragraphs, and suggested as an answer by some models. For more details on compound resolution problems, see Subsection 5.2.2.

For each answer provided from the model for `additive` and `solvent`, an attempt at resolving the `cid` is made. If a `cid` is found, it is compared with the label `cid`. An answer is counted as 'wrong' when the resolved `cid` is different to the label, but also when resolving fails. Thus, the answer 'distilled H2O' would be counted as 'wrong', since it could not be resolved to a valid `cid`.

5. Results and Discussion

As mentioned in the previous Chapter 4, the approach of this work is to guide a number of open-access LLMs through prompts to extract `temperature`, `time` and `solvent` data from MOF synthesis paragraphs.

Section 5.1 describes and discusses extraction accuracy across the different model sizes, and potential insights gained there. Building on that, Section 5.2 provides deeper insight in different failure modes, before Section 5.3 provides excerpts of why attempts for fine-tuning ultimately failed.

Note on Differences to Implementation

While all code as explained and partially showcased in Section 4.1 includes the additional category of `additive`, figures in this chapter do not. This decision was made due to mistaken modeling of the task, which resulted in accuracy substantially worse than it probably is when modelled properly. It was modeled as extraction task, similar to `solvent`. However, it ought to have been a classification task, with the options `[None, "base", "acid"]`. Thus, all 7B-sized models had an accuracy of less than 1% (except for Vicuna-7B with 2.5%), and even large models had either below 1% or only around 15% correct (both LLaMa 2 and Falcon, not the instruct-variant). Additionally, a lot of answers would have been `None`, but all models always answered *something*, because it did not seem apparent that `None` is a possible answer and only very large models have the capability of Reflexion [59].

This was only noticed after all models had been run, and was not redone due to time constraints.

5.1 Accuracy Overview

5.1.1 7B Parameter Models

As can be seen in Figure 5.1, accuracy in extracting `temperature` and `time` data varies wildly across models, and good performance in one does not always correlate with good performance in the other.

LLaMa 2 has the highest accuracy for both `temperature` and `time`, but Falcon is more accurate in `solvent` prediction on our dataset. Based on this, it seems that LLaMa 2-7B already achieves close to the highest accuracy for extraction of `temperature`, and is close behind on `time`.

More information on why LLaMa is so much worse in `time` accuracy than `temperature` accuracy can be found in Subsection 5.2.1.

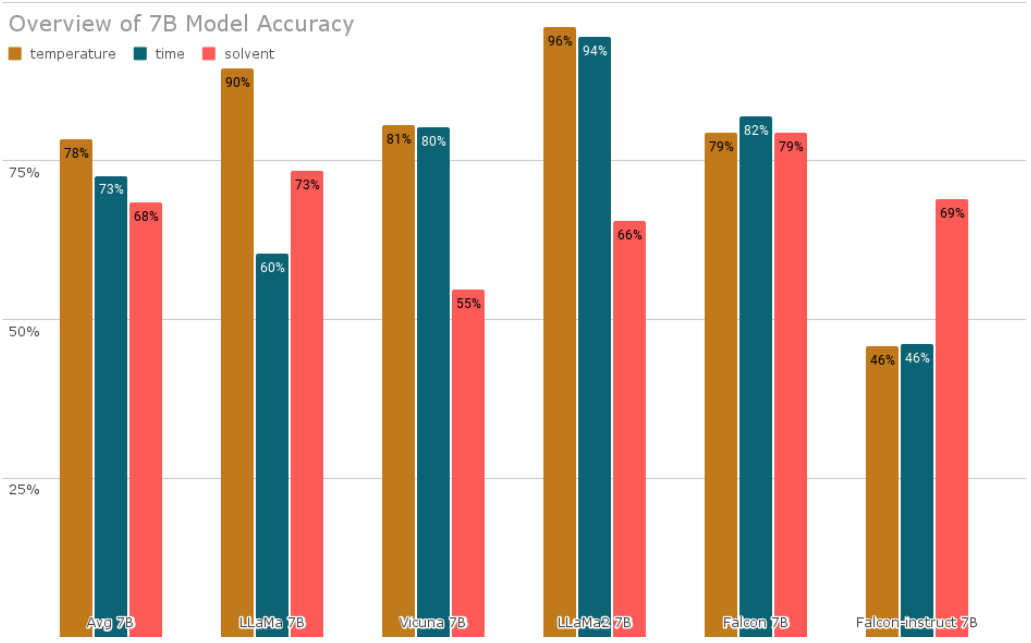

Figure 5.1: Overview of the Accuracy of Models with a Size of 7B parameters. On the left is the average over all 7B sized models. The accuracy of `temperature` and `time` is within 3 percentage points of the other for each model except LLaMa, where the difference is 30 percentage points. On average, the accuracy for `temperature` is 78%, 73% for `time`, and 68% for `solvent`.

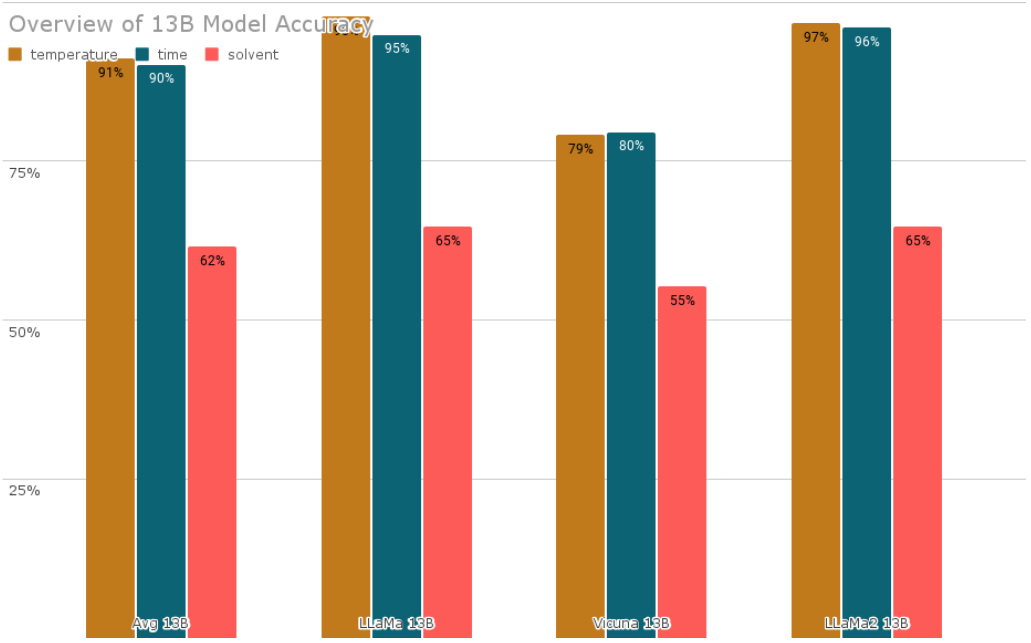

Figure 5.2: Overview of the Accuracy of Models with a Size of 13B parameters. Leftmost is the average over all 13B sized models. Falcon does not have a 13B sized variant. On average, the accuracy for `temperature` is 91%, 90% for `time` and 62% for `solvent`. LLaMa achieves an accuracy of 98% for `temperature`, about 20 percentage points ahead of Vicuna with 79%.

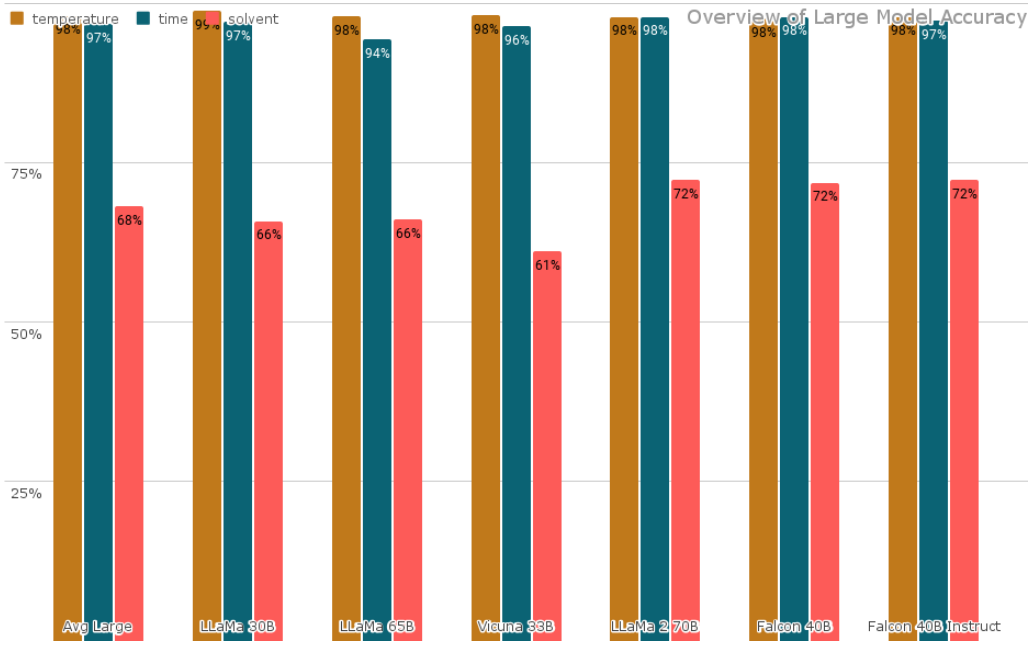

Figure 5.3: Overview of the Accuracy of Models with a Size over 30B parameters. Leftmost is the average over all models upwards of 30B parameters. On average, the accuracy for `temperature` is 98%, 97% for `time` and 68% for `solvent`. LLaMa 2 does not have a 30B or 33B-sized variant. There is also no Vicuna-65B variant made available from LMSYS. LLaMa 2 and both Falcon models achieve 72% accuracy on `solvent` extraction, 6-11 percentage points more than other models.

5.1.2 13B Parameter Models

As can be seen in Figure 5.2, accuracy improved on average, but primarily for LLaMa. In fact, the accuracy of Vicuna-13B even slightly degraded when compared to Vicuna-7B on `temperature` and `time`, and accuracy on everything mostly stayed stagnant for LLaMa 2. LLaMa increased accuracy on `temperature` by 6 percentage points, but particularly on `time` by 33 percentage points.

Vicuna-13B is the weakest in accuracy for `temperature` and `time`, trailing by about 14 percentage points behind LLaMa and LLaMa 2. From what was seen from Falcon-instruct-7B in Subsection 5.1.1, Falcon-instruct-13B should still be worse than Vicuna-13B, but Falcon does not have a 13B variant.

5.1.3 Large Models

As can be seen in Figure 5.3, scores between large models differ marginally at best. All models have an accuracy of 98% or above on `temperature` extraction, and an average accuracy of 97% on `time`.

Surprisingly, the accuracy on `solvent` extraction improved by less than maybe expected. In fact, as can be seen in the previous Figure 5.1 comparing the accuracy of 7B sized models, smaller models were on average similarly capable. Although, variance of accuracy between smaller models included those both more and less capable than its larger siblings. This may be due to accidential success, particularly in the case of Falcon-7B having the highest accuracy on `solvent` extraction across all models and sizes. However, LLaMa-7B similarly having a higher accuracy than any of its bigger siblings indicates something structural that would warrant further analysis.

5.2 Frequent Mistakes

Analysis of cases where the extracted data was incorrect can provide valuable insight in failure modes. Insight on unit confusion, particularly of smaller models, is provided in Subsection 5.2.1. Resolution of solvents and problems with it are looked into in Subsection 5.2.2, before Subsection 5.2.3 gives a short account of a more complex failure mode.

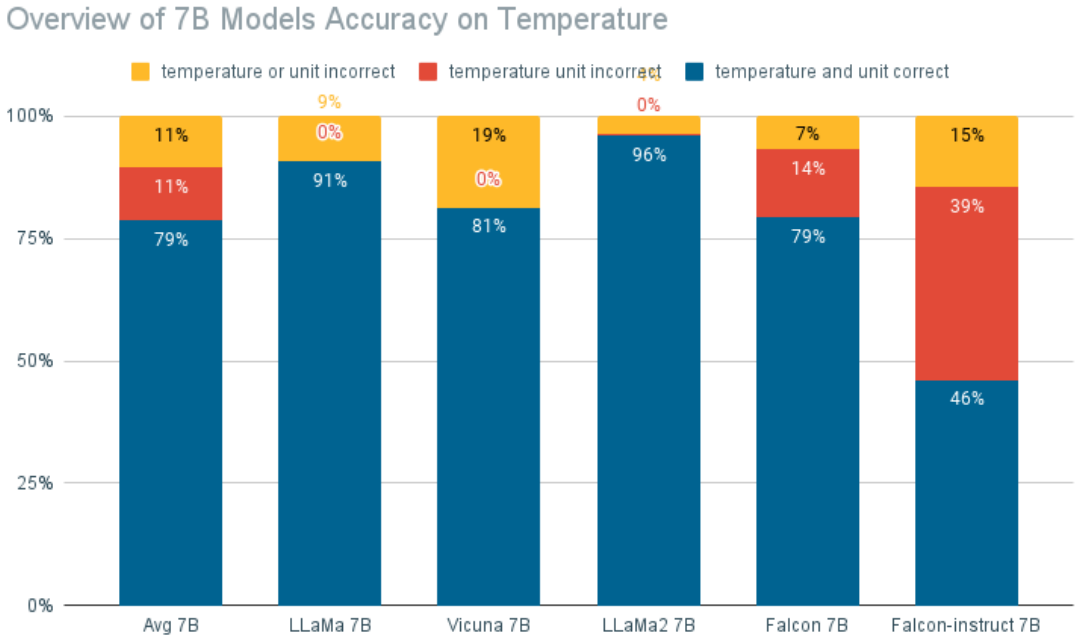

Figure 5.4: Detailed Overview of 7B Models Accuracy on Temperature. On average, 7B sized models had an accuracy of 79% on the extraction of the `temperature` the synthesis was run at. Most models did not have problems with temperature units, apart from Falcon.

5.2.1 Unit Confusion

One interesting observation is that smaller models appear to make a lot of 'easy' mistakes. Not all such simple mistakes can be easily classified, but mistakenly outputting the wrong unit can. For *unit confusion*, the following instances where counted: the answer (on `temperature` or `time`) provided from a model is wrong, but interpreting the same answer with a different unit could get the correct answer. More details on model output structure can be found in Section 4.3.

For example, in one instance Falcon confidently answered 20 °K, when the correct answer would have been 20 °C.

Temperature Units

As can be seen in Figure 5.4, only Falcon experiences unit confusion for temperature – and in particular the instruct-variant. It is unclear why that might be the case exactly, but it seems to be in part of the dataset Falcon was trained on.

Time Units

Unit confusion on `time` seems to affect all models to some degree, as can be seen in Figure 5.5. Interestingly, LLaMa seems to be affected the most, followed by Falcon. This also explains how LLaMa could quickly gain 33 percentage points in accuracy between sizes 7B and 13B (Compare Figure 5.1 and Figure 5.2) – mostly due to using correct time units.

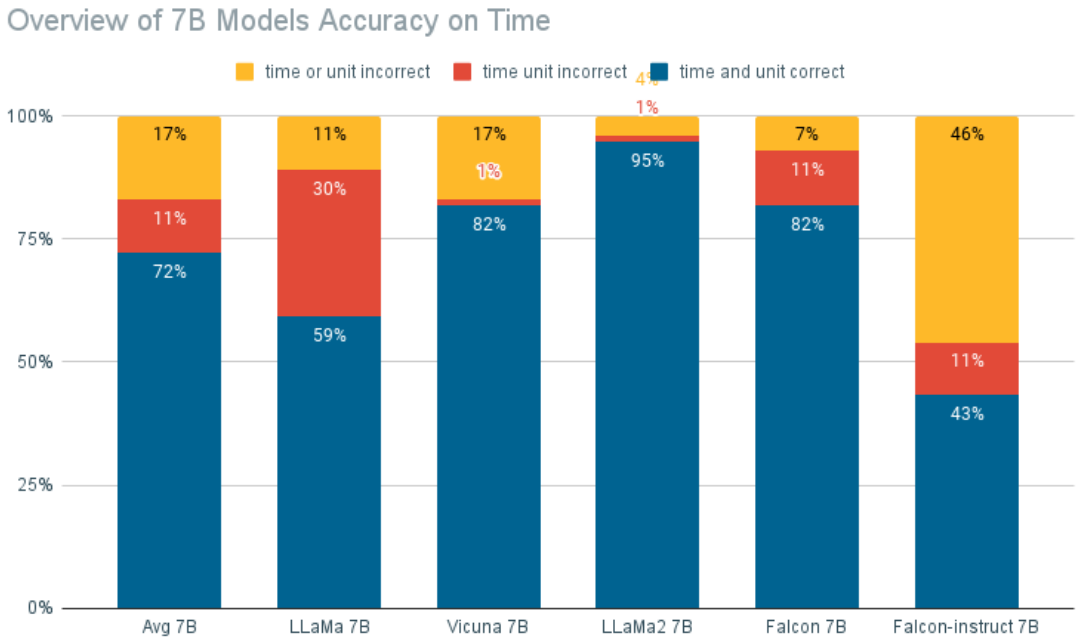

Figure 5.5: Detailed Overview of 7B Models Accuracy on Time. On average, 7B sized models had an accuracy of 72% on the extraction of the synthesis duration, and a third of the mistakes (11% of 28% total mistakes), were simply due to getting the unit of the duration wrong. LLaMa struggled the most, providing inaccurate units for about a third of all answers on `time`. Both Falcon and Falcon-instruct confused units in about 11% of answers. However, Vicuna and LLaMa 2 gave the wrong unit in only 1% of cases.

Bigger-sized Models

Unit confusion on models sized 13B parameters or more is happening in less than 0.5% of cases (in 0-4 answers of 778 data points), and is thus of no further interest.

5.2.2 Solvent Resolution

As mentioned previously in Subsection 4.5.2, compounds may have multiple names that are used in different synthesis paragraphs, but not all of them may be resolved from `pubchempy`. Thus, taking a closer look at how often answers for `solvent` from 7B-sized models could not be resolved in Figure 5.6 reveals an interesting picture.

First, all models seem to be affected by not being able to resolve `solvent` to some degree. The previously mentioned case of 'distilled H2O' not resolving albeit clearly occuring in eight synthesis paragraphs as solvent, indicates that at least some of the answers provided by models may be, in fact, correct. However, this is probably not the case for most of the answers provided by Vicuna-7B, where about 31% of answers did not resolve. This also indicates that the approach taken for compound resolution in this work can be improved.

Second, larger models seem to be affected even more, as can be seen in Figure 5.7. One hypothesis for this observation is that the models are more accurate, and quote the correct `solvent` verbatim, but the answer cannot be resolved. This may be true in particular for the `solvent N,N-DIMETHYLACETAMIDE` (`cid` 31374), where the synthesis paragraphs contain none of its 125 synonyms in 34 cases (or about 4.37% of the dataset).

5.2.3 Orders of Magnitude

In some cases, the following mistake was observed: answers of `time` where clearly supposed to be in seconds (e.g. 864000 seconds are ten days), but the number of zeroes was off

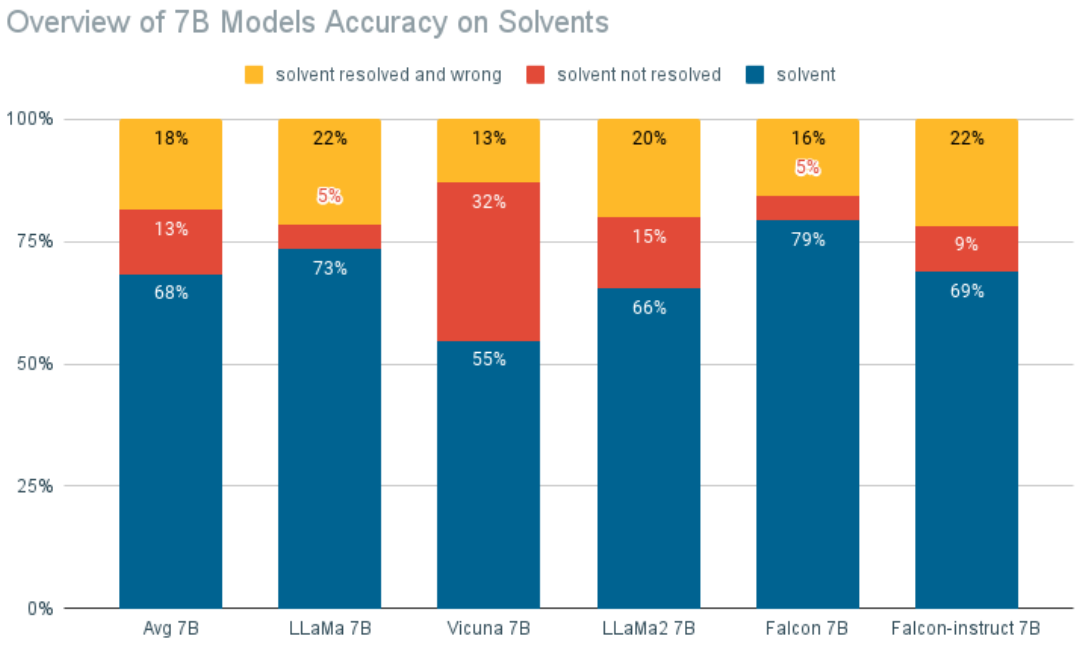

Figure 5.6: Detailed Overview of 7B Models Accuracy on Solvents. On average, accuracy on `solvent` was 68%, and an additional 13% could not be resolved. LLaMa-7B and Falcon-7B had the fewest unresolved answers, at 5% each.

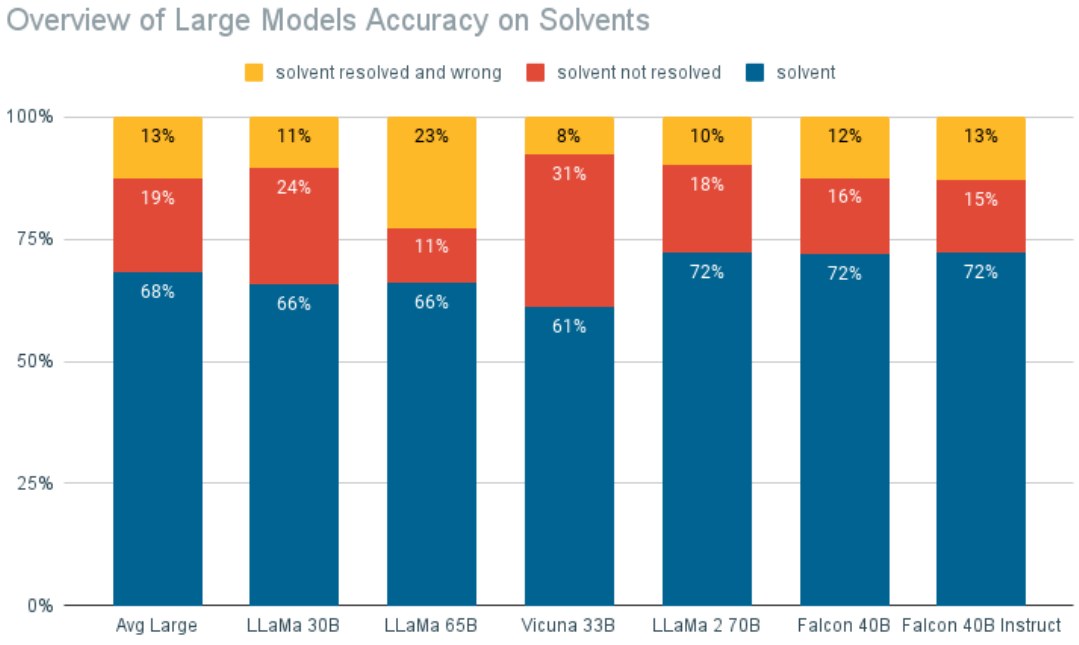

Figure 5.7: Detailed Overview of Accuracy from Models with a Size over 30B parameters on Solvents. On average, accurcy on `solvent` was 68%, and an additional 19% could not be resolved. The remaining 13% resolved to a compound not used as `solvent`.

by one. A similar mistake happened sometimes with `temperature` units in Kelvin. This did not preclude the unit from being wrong. In fact, it sometimes appears that (since the unit information is generated later) the model attempted to increase or decrease its previous answer with a different unit than would be expected, making it closer to correct than simply having the wrong number of zeros – but still wrong. Smaller models seem to have a higher tendency of making these mistakes, but no further analysis was done due to the more complex nature of the error.

5.3 Supervised Fine Tuning

Pretrained LLMs are trained on so much data that they become generally capable of most tasks even with no additional fine-tuning [16], but this usually takes many thousands of GPU-hours [39, 40]. Instead, fine-tuning a pretrained model has substantial benefits: it requires only a fraction of the GPU-hours, and sometimes more importantly, a lot fewer examples for training to achieve similar results to a specifically trained model for most tasks [43].

More information on pre-training a LLM can be found in Section 3.2, and Subsection 3.2.2 has additional details on fine-tuning.

In the end, fine-tuning was abandoned for this work due to lack of success and time constraints. See Chapter 6 for conclusions and Section 7.3 for an outlook on comparisons with fine-tuned models.

What follows are two excerpts of what was encountered when attempting to fine-tune LLMs using th `transformers` library. These excerpts are by no means exhaustive, as many hours of bugfixing where necessary to even get to these points. Further hours, while sometimes yielding different errors, yielded no additional results or insight. Neither did interacting with the community, where a number of people had slightly different but very similar problems, almost always with no answer in any official or unofficial forum.

5.3.1 Excerpt 1: Broken Models

Since for this work a custom dataset is used, it also requires a custom dataloader. Traditionally, specifically for training, a dataloader contains one field for the 'input' of the model, and one for the label, or 'output' of the model, the optimizer should optimize for as a result of the input. Each item of the dataloader is a dictionary of keys, with specific keys for input and output. There is however, only partial and very conflicting documentation on the existance and usage of these keys.

Additionally, LLMs require input to be converted to tokens before being able to process it. In the various examples that can be found, three main keys are of primary importance for dataloaders for LLMs:

- `"input_ids"`
- `"attention_mask"`
- `"labels"`

Code Example 4

Excerpt of what could be found in a custom dataloader. `text` describes any string the model may be provided as input. The tokenizer converts any string to a list of tokens and an attention mask, among other things.

```
text_encodings = tokenizer(text, ...)

return {
    "input_ids": text_encodings["input_ids"],
    "attention_mask": text_encodings["attention_mask"],
    "label": ...
}
```

However, none of these is properly documented or explained. The common usage of the keys `"input_ids"` and `"attention_mask"` is rather straightforward, and few working examples show anything different to what can be seen in Code Example 4. Despite that, `"label"` is not so clear.

Since neither of the three is documented, what remains is looking at use from others, or digging into the code. Multiple tutorials and even official sources (e.g. Microsoft [60]), use it in the following way: `"label": text_encodings["input_ids"]`.

This, in fact, can train a model and result in model weights differing from the original ones. However, all such trained models for this work were broken, as they did not generate anything that was not an `"<EOS>"`-token. These tokens are usually used as a stopping criterion during generation – meaning that the model predicts that a natural separation happens between the previous and the next tokens. This is true in cases where `"<EOS>"` denotes a switch from user-generated to assisstent-generated text, a paragraph or answer ended or similar breaks in continuity (as it will be used for in training data). Nonetheless, this is not the case here. Even when ignoring the *first* such token, no further `"<EOS>"`-tokens should follow immediately after.

At this point, it is unclear what exactly happened. Looking through source code of used modules from the `transformers` library provided little insight. The main hypothesis for the failure is that the optimizer trained the model that *no additional output* is required, since both `"input_ids"` and `"label"` where equal. However, this approach seems to work for other groups in certain circumstances, which substantially reduces the credibility of the hypothesis. So far, no other potential explanation has been found.

Additionally, attempts at mask manipulation were not possible due to the nature of causal language model which all LLMs are. Other sources tokenize different text for `"label"`, though this is more rare and exact usage is inconsistent. No further attempts at this approach where made due to focusing on other, seemingly more promising venues.

5.3.2 Excerpt 2: Broken Libraries

Another attempt made use of the high-level HuggingFace `trl` (Transformer Reinforcement Learning) library, purpose-built for use-cases of (supervised) fine-tuning.

The couple of examples provided, while working with only a few lines of code, obscure the complex and undocumented inner workings of the library. Usually, a `DataCollator` is used to batch-process multiple items from the dataloader, and items from the dataloader are already tokenized. However, the in examples used but otherwise undocumented `DataCollatorForCompletionOnlyLM` (or more specifically, the underlying

`DataCollatorForLanguageModeling`) attempts to abstract away the process of tokenization, so that any dataloader providing text samples could be used, independent of tokenization. This abstraction would generally reduce redundant boilerplate code.

Code Example 5

Counterintuitively, this is not a `KeyError`.

```
|------------- Traceback (most recent call last) ---------------|
|           ...                                                 |
|   372 |   model.train()  # put the model in training mode     |
| > 373 |   trainer.train()                                     |
...
ValueError: You should supply an encoding or a list of encodings
to this method that includes "input_ids", but you provided []
```

After inspecting some of the working examples to learn that the dataloaders provided a `"text"`-key, and adapting other code accordingly to *not* tokenize it beforehand, this resulted in a `ValueError` missing `"input_ids"`, as can be seen in the full message in Code Example 5. When instead tokenizing before the `DataCollatorForCompletionOnlyLM` (providing tokenized `"input_ids"` as key), it would fail, complaining that it will do tokenization itself.

6. Conclusion

Coming back to the initially posed goals of this work from Section 1.1:

First, zero-shot automated information extraction from scientific literature was successfully demonstrated with high accuracy achieved by most models of all sizes, as detailed in Section 5.1.

Second, the capabilities of some of the most capable open-access LLMs was measured and compared specifically for the information extraction task on scientific data, detailed also in Section 5.1. Furthermore, analyzing frequent mistakes in Section 5.2 gave additional insight in failure modes and further venues for inquiry and improvement.

Third, a lot of time and effort was put in attempting to fine-tune models for the information extraction task, which was substantially harder than initially assumed, and eventually abandoned. Section 5.3 has excerpts of the difficulties encountered, which gives some insight to the task of fine-tuning in practice.

This work demonstrates and measures the fundamental capability of various models for information extraction tasks for scientific literature in a zero-shot setting with an easily configurable pipeline. This pipeline can easily be configured to extract additional parameters. Hybrid systems (automatic extraction with manual oversight) could substantially increase data quality for various domains of research even for very complex parameters. This indicates that LLMs can be powerful tools for information extraction, even in setups where the model is self-hosted with no fine-tuning.

Most surprising is the comparatively high accuracy that 13B-sized models are able to achieve, particularly LLaMa and LLaMa 2. Many consumer graphics cards have VRAM of more than 10GB available, which is sufficient to load 13B-sized models with 4-bit quantization and achieve a throughput of 30 to 40 tokens per second, depending on the specific GPU used [61]. Thus, high-accuracy information extraction capabilities are much more accessible than previously expected, which could have far-reaching implications for the future of NLP.

7. Outlook

This work aimed to answer a number of questions surrounding the use of LLMs for automated information extraction from scientific literature. A number of questions have been answered, and insights where gained. This newfound knowledge provides the opportunity to ask better questions.

7.1 Further Analysis

While a certain amount of analysis on failure modes was done, there remain numerous avenues unexplored. One such avenue is the full exploration of `solvent` resolution cases. How many of the model-provided answers are actually right, but couldn't be resolved? Of those that could be resolved but are wrong, what did the models answer, and why?

A secondary avenue for exploration could be the more accurate querying for `additive` and additional parameters.

Answering these questions could provide substantial insight on the problem of NER with LLMs.

7.2 Prompts

For this work, the most effort was put in attempting to fine-tune the models used. Consequently, not much effort was put in designing the best prompt for a zero-shot setting. Better prompts could potentially be automatically generated to work well across models [62]. Using tools from mechanistic interpretability [63] it should be possible to generate highly-specialized prompts [64] for each model as well. They may not generalize, however.

Additionally, the more sophisticated `guidance` [56] library now allows integration of `transformers` models. Using it instead of `jsonformer` should enable more fine-grained control and allow for better outputs overall.

7.3 Fine-Tuning

While attempted, working fine-tuned models where not achieved during this work. As mentioned before, a model fine-tuned for a specific task should have better performance than a generally pretrained model [14].

However, as demonstrated in this work, fine-tuning may not be necessary depending on the task. A detailed comparison with fine-tuned models would still be able to provide substantial insights in failures of current models. Additionally, the question of how much more accurate existing open-access Large Language Models could become on information extraction tasks remains unanswered.

7.4 Different Frameworks

In recent months an ecosystem of LLM orchestration frameworks emerged. Orchestration frameworks often build on top of the abstractions provided by the `transformers` library, but bring additional tooling for using LLMs particularly in agentic use-cases. Additionally, interfacing, chaining and fine-tuning of LLMs is supposed to be easier. The most prominent orchestration frameworks are currently LangChain [65] and HayStack [66], though with the current pace of new frameworks emerging, this may change over time.

7.5 Next-Gen Models

While this work compared a number of open-access LLMs, a comparison with closed-access models such as Falcon-180B or GPT4 with or without fine-tuning would provide additional insights on relative capabilities.

7.6 Comparison to Masked Language Models

The task of knowledge extraction may be well suited to masked language models with modern BERT-derived architectures such as T5 [67]. Thus, such a a comparison would be of particular interest.

Glossary

BLOOM **B**igScience **L**arge **O**pen-science **O**pen-access **M**ultilingual LM, a 176 billion parameter open-source LLM created through a cooperation between Google, HuggingFace and various smaller organisations [48]. 10, 16

causal language model A causal language model predicts the likelihood of the next token based on a sequence of tokens (input). By sampling one of the predicted tokens and appending it to the input, output can be generated autoregressively. This in contrast to e.g. a masked language model. 8, 11, 12, 16, 28, 36

ChatGPT A chat interface for GPT3.5 or GPT4 from OpenAI, which makes it easier to interface with the model. Subsection 3.2.3 provides more information on instruct-based models, chat-variants like ChatGPT are an extension of. 5, 10

F-Score $2 \cdot (precision \cdot recall)/(precision + recall)$, where *precision* is a metric for how many retrieved items are relevant, and *recall* is a metric for how many relevant items where retrieved. 8

Falcon One of the LLMs used. Created by the TII. See Subsection 4.2.7 for details. 1, 3, 11, 17, 18, 21–26, 34

Google Now called Alphabet, search engine giant and AI powerhouse, well known for training supersized machine learning models with unrealistic hardware requirements for just about anyone else. 9, 16, 35, 36

GPT2 The second generation **G**enerative **P**retrained **T**ransformer LM from OpenAI [33]. 10, 36

GPT3 The third generation **G**enerative **P**retrained **T**ransformer LM from OpenAI [16]. 8, 10, 16, 35, 36

GPT4 The fourth generation **G**enerative **P**retrained **T**ransformer LM from OpenAI [6]. Currently their most capable model. 17, 34–36

HuggingFace American deep learning ecosystem startup, having created the well established `transformers` framework which provides useful abstractions of most existing open-access Machine Learning models. 15, 16, 18, 28, 35, 37

Large Model Systems Organization The Large Model Systems Organization eponymously develops large models and systems that are open, accessible, and scalable, e.g. Vicuna. 17, 23

LLaMa A LLM from Meta. See Subsection 4.2.3 for details. 1, 3, 11, 16–18, 21–26, 31, 35, 36

LLaMa 2 One of the LLMs used. It is the successor of LLaMa, also created by Meta. See Subsection 4.2.6 for details. 1, 3, 10, 11, 17, 18, 21, 23, 25, 31

masked language model A masked language model predicts all masked (often missing) tokens in a sequence based on the context provided by the surrounding tokens. This in contrast to e.g. a causal language model. 8, 34, 35

Meta Previously known as Facebook, Meta is a deep learning powerhose and regularly open-sources new state-of-the-art machine learning models. 16, 17, 35, 36

Microsoft Tech Giant, well-known for its operating system. Microsoft recently started intensive cooperation with OpenAI through a $10 Billion USD investment, and started integrating GPT4 and other models throughout their services. 18, 28

OpenAI American AI company, trailblazer at the frontier of scaling deep learning architectures and corresponding algorithmic breakthroughs. Their currently most well-known models are the GPT family of models, particularly GPT2, GPT3 and GPT4. 10, 17, 35, 36

OPT Open Pretrained Transformer, a 175 billion parameter open-source LM from Meta research [47]. 10, 16

PaLM An extremely capable closed-access LLM from Google [36]. 10, 17

percentage point One percentage point denotes an absolute increase of 1% in the underlying measure. 22–24

Stanford Alpaca A LLM based on LLaMa. See Subsection 4.2.4 for details. 11, 16, 17

Technology Innovation Institute Abu Dhabi-based machine learning research institute. 17, 35

Vicuna One of the LLMs used. Based on LLaMa. See Subsection 4.2.5 for details. 11, 16–18, 21–23, 25

Acronyms

`transformers` HuggingFace `transformers` [45] 15, 18, 27, 28, 33, 34

BERT Bidirectional Encoder Representation from Transformers [5] 8–10, 34

GPT Generative Pretrained Transformer 17, 36

GQA Grouped Query Attention [20] 10, 12, 17

LLM Large Language Model 1, 5, 7–12, 15–19, 21, 27, 28, 31, 33–36

LM Language Model 5, 7–10, 17, 35, 36

LOC Lines of Code 15

ML Machine Learning 1, 3, 5, 11

MLP Multi-Layer Perceptron 11, 12

MOF Metal Organic Framework 1, 5, 19, 21

NER Named Entity Recognition 1, 7, 8, 18, 33

NLP Natural Language Processing 1, 5, 7–9, 16, 31

ReLU Rectified Linier Unit 10–12

RoPE Rotary Positional Encoding [19] 10, 12

state-of-the-art State of the Art 9, 16, 36

SwiGLU Swish Gated Linear Unit [18] 10, 12

Bibliography

1. Saal, J. E., Oliynyk, A. O. & Meredig, B. Machine Learning in Materials Discovery: Confirmed Predictions and Their Underlying Approaches. *Annual Review of Materials Research* **50,** 49–69. doi:10.1146/annurev-matsci-090319-010954 (2020).

2. Luo, Y., Bag, S., Zaremba, O., Cierpka, A., Andreo, J., Wuttke, S., Friederich, P. & Tsotsalas, M. MOF Synthesis Prediction Enabled by Automatic Data Mining and Machine Learning**. en. *Angewandte Chemie International Edition* **61,** e202200242. ISSN: 1521-3773. doi:10.1002/anie.202200242. https://onlinelibrary.wiley.com/doi/abs/10.1002/anie.202200242 (2023-02-01) (2022).

3. Choudhary, K., DeCost, B., Chen, C., Jain, A., Tavazza, F., Cohn, R., Park, C. W., Choudhary, A., Agrawal, A. & Billinge, S. J. Recent Advances and Applications of Deep Learning Methods in Materials Science. *npj Computational Materials* **8,** 59. doi:10.1038/s41524-022-00734-6 (2022).

4. Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J. W. & Sifre, L. *An Empirical Analysis of Compute-Optimal Large Language Model Training* en. in *Advances in Neural Information Processing Systems* (2022-10). https://openreview.net/forum?id=iBBcRUlOAPR (2023-01-18).

5. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv preprint arXiv:1810.04805.* arXiv: 1810.04805 (2018).

6. OpenAI. *GPT-4 Technical Report* 2023. https://cdn.openai.com/papers/gpt-4.pdf (2023-03-14).

7. Shi, Y., Prieto, P. L., Zepel, T., Grunert, S. & Hein, J. E. Automated Experimentation Powers Data Science in Chemistry. *Accounts of Chemical Research* **54,** 546–555. doi:10.1021/acs.accounts.0c00736 (2021).

8. Li, J., Sun, A., Han, J. & Li, C. A Survey on Deep Learning for Named Entity Recognition. *IEEE Transactions on Knowledge and Data Engineering* **34,** 50–70. ISSN: 1041-4347, 1558-2191, 2326-3865. doi:10.1109/TKDE.2020.2981314. arXiv: 1812.09449 [cs]. http://arxiv.org/abs/1812.09449 (2023-09-21) (2022-01).

9. LaPorte, J. in *The Stanford Encyclopedia of Philosophy* (eds Zalta, E. N. & Nodelman, U.) Winter 2022 (Metaphysics Research Lab, Stanford University, 2022). https://plato.stanford.edu/archives/win2022/entries/rigid-designators/ (2023-09-21).

10. Hobbs, J. R. Coherence and Coreference*. en. *Cognitive Science* **3,** 67–90. ISSN: 1551-6709. doi:10.1207/s15516709cog0301_4. https://onlinelibrary.wiley.com/doi/abs/10.1207/s15516709cog0301_4 (2023-09-21) (1979).

11. Hawizy, L., Jessop, D. M., Adams, N. & Murray-Rust, P. ChemicalTagger: A Tool for Semantic Text-Mining in Chemistry. *Journal of Cheminformatics* **3,** 17. ISSN: 1758-2946. doi:10.1186/1758-2946-3-17. https://doi.org/10.1186/1758-2946-3-17 (2023-02-01) (2011-05).

12. Beard, E. J., Sivaraman, G., Vazquez-Mayagoitia, A., Vishwanath, V. & Cole, J. M. Comparative Dataset of Experimental and Computational Attributes of UV/Vis Absorption Spectra. en. *Scientific Data* **6,** 307. ISSN: 2052-4463. doi:10.1038/s41597-019-0306-0. https://www.nature.com/articles/s41597-019-0306-0 (2023-02-20) (2019-12).

13. Huang, S. & Cole, J. M. A Database of Battery Materials Auto-Generated Using ChemDataExtractor. en. *Scientific Data* **7,** 260. ISSN: 2052-4463. doi:10.1038/s41597-020-00602-2. https://www.nature.com/articles/s41597-020-00602-2 (2023-02-20) (2020-08).

14. Zhao, X., Greenberg, J., An, Y. & Hu, X. T. *Fine-Tuning BERT Model for Materials Named Entity Recognition* in *2021 IEEE International Conference on Big Data (Big Data)* (2021-12), 3717–3720. doi:10.1109/BigData52589.2021.9671697.

15. Dunn, A., Dagdelen, J., Walker, N., Lee, S., Rosen, A. S., Ceder, G., Persson, K. & Jain, A. Structured Information Extraction from Complex Scientific Text with Fine-Tuned Large Language Models. *arXiv:2212.05238.* doi:10.48550/arXiv.2212.05238. arXiv: 2212.05238 [cond-mat]. http://arxiv.org/abs/2212.05238 (2023-02-01) (2022-12).

16. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G. & Askell, A. Language Models Are Few-Shot Learners. *Advances in neural information processing systems* **33,** 1877–1901 (2020).

17. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. & Polosukhin, I. Attention Is All You Need. *Advances in neural information processing systems* **30** (2017).

18. Shazeer, N. Glu Variants Improve Transformer. *arXiv preprint arXiv:2002.05202.* arXiv: 2002.05202 (2020).

19. Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B. & Liu, Y. RoFormer: Enhanced Transformer with Rotary Position Embedding. *arXiv:2104.09864.* arXiv: 2104.09864 [cs]. http://arxiv.org/abs/2104.09864 (2023-04-03) (2022-08).

20. Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F. & Sanghai, S. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. *arXiv preprint arXiv:2305.13245.* arXiv: 2305.13245 (2023).

21. Bolya, D., Fu, C.-Y., Dai, X., Zhang, P. & Hoffman, J. Hydra Attention: Efficient Attention with Many Heads. *arXiv:2209.07484.* arXiv: 2209.07484 [cs]. http://arxiv.org/abs/2209.07484 (2023-03-16) (2022-09).

22. Sukhbaatar, S., Grave, E., Bojanowski, P. & Joulin, A. Adaptive Attention Span in Transformers. *arXiv:1905.07799.* doi:10.48550/arXiv.1905.07799. arXiv: 1905.07799 [cs, stat]. http://arxiv.org/abs/1905.07799 (2023-04-02) (2019-08).

23. Lu, Y., Li, Z., He, D., Sun, Z., Dong, B., Qin, T., Wang, L. & Liu, T.-Y. Understanding and Improving Transformer From a Multi-Particle Dynamic System Point of View. en. *arXiv:1906.02762.* arXiv: 1906.02762 [cs, stat]. http://arxiv.org/abs/1906.02762 (2023-01-18) (2019-06).

24. Ye, X., He, Z., Heng, W. & Li, Y. Toward Understanding the Effectiveness of Attention Mechanism. en. *AIP Advances* **13,** 035019. ISSN: 2158-3226. doi:10.1063/5.0141666. https://aip.scitation.org/doi/10.1063/5.0141666 (2023-04-03) (2023-03).

25. Wu, Y., Rabe, M. N., Hutchins, D. & Szegedy, C. Memorizing Transformers. *arXiv:2203.08913.* arXiv: 2203.08913 [cs]. http://arxiv.org/abs/2203.08913 (2023-02-21) (2022-03).

26. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. & Salakhutdinov, R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The journal of machine learning research* **15,** 1929–1958 (2014).

27. Ba, J. L., Kiros, J. R. & Hinton, G. E. Layer Normalization. *arXiv:1607.06450.* doi:10.48550/arXiv.1607.06450. arXiv: 1607.06450 [cs, stat]. http://arxiv.org/abs/1607.06450 (2023-03-08) (2016-07).

28. Child, R., Gray, S., Radford, A. & Sutskever, I. Generating Long Sequences with Sparse Transformers. *arXiv:1904.10509.* doi:10.48550/arXiv.1904.10509. arXiv: 1904.10509 [cs, stat]. http://arxiv.org/abs/1904.10509 (2023-03-02) (2019-04).

29. Wu, C., Wu, F., Qi, T., Huang, Y. & Xie, X. Fastformer: Additive Attention Can Be All You Need. *arXiv:2108.09084.* arXiv: 2108.09084 [cs]. http://arxiv.org/abs/2108.09084 (2023-02-27) (2021-09).

30. Hua, W., Dai, Z., Liu, H. & Le, Q. *Transformer Quality in Linear Time* in *International Conference on Machine Learning* (PMLR, 2022), 9099–9117.

31. Dao, T., Fu, D. Y., Ermon, S., Rudra, A. & Ré, C. Flashattention: Fast and Memory-Efficient Exact Attention with Io-Awareness. *arXiv preprint arXiv:2205.14135.* arXiv: 2205.14135 (2022).

32. Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Akhtar, N., Barnes, N. & Mian, A. A Comprehensive Overview of Large Language Models. *arXiv:2307.06435.* arXiv: 2307.06435 [cs]. http://arxiv.org/abs/2307.06435 (2023-09-24) (2023-09).

33. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. Language Models Are Unsupervised Multitask Learners. en. *published on GitHub* (2019).

34. Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., Aslanides, J., Henderson, S., Ring, R., Young, S., Rutherford, E., Hennigan, T., Menick, J., Cassirer, A., Powell, R., van den Driessche, G., Hendricks, L. A., Rauh, M., Huang, P.-S., Glaese, A., Welbl, J., Dathathri, S., Huang, S., Uesato, J., Mellor, J., Higgins, I., Creswell, A., McAleese, N., Wu, A., Elsen, E., Jayakumar, S., Buchatskaya, E., Budden, D., Sutherland, E., Simonyan, K., Paganini, M., Sifre, L., Martens, L., Li, X. L., Kuncoro, A., Nematzadeh, A., Gribovskaya, E., Donato, D., Lazaridou, A., Mensch, A., Lespiau, J.-B., Tsimpoukelli, M., Grigorev, N., Fritz, D., Sottiaux, T., Pajarskas, M., Pohlen, T., Gong, Z., Toyama, D., d'Autume, C. d. M., Li, Y., Terzi, T., Mikulik, V., Babuschkin, I., Clark, A., Casas, D. d. L., Guy, A., Jones, C., Bradbury, J., Johnson, M., Hechtman, B., Weidinger, L., Gabriel, I., Isaac, W., Lockhart, E., Osindero, S., Rimell, L., Dyer, C., Vinyals, O., Ayoub, K., Stanway, J., Bennett, L., Hassabis, D., Kavukcuoglu, K. & Irving, G. Scaling Language Models: Methods, Analysis & Insights from Training Gopher. *arXiv:2112.11446.* arXiv: 2112.11446 [cs]. http://arxiv.org/abs/2112.11446 (2023-04-09) (2022-01).

35. Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O. & Sifre, L. Training Compute-Optimal Large Language Models. *arXiv:2203.15556.* doi:10.48550/arXiv.2203.15556. arXiv: 2203.15556 [cs]. http://arxiv.org/abs/2203.15556 (2023-02-06) (2022-03).

36. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C. & Gehrmann, S. Palm: Scaling Language Modeling with Pathways. *arXiv preprint arXiv:2204.02311.* arXiv: 2204.02311 (2022).

37. Zhang, Z., Zhang, A., Li, M., Zhao, H., Karypis, G. & Smola, A. Multimodal Chain-of-Thought Reasoning in Language Models. *arXiv preprint arXiv:2302.00923.* arXiv: 2302.00923 (2023).

38. Loshchilov, I. & Hutter, F. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101.* arXiv: 1711.05101 (2017).

39. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E. & Lample, G. LLaMA: Open and Efficient Foundation Language Models. *arXiv:2302.13971.* doi:10.48550/arXiv.2302.13971. arXiv: 2302.13971 [cs]. http://arxiv.org/abs/2302.13971 (2023-03-14) (2023-02).

40. Scao, T. L., Wang, T., Hesslow, D., Saulnier, L., Bekman, S., Bari, M. S., Biderman, S., Elsahar, H., Muennighoff, N., Phang, J., Press, O., Raffel, C., Sanh, V., Shen, S., Sutawika, L., Tae, J., Yong, Z. X., Launay, J. & Beltagy, I. What Language Model to Train If You Have One Million GPU Hours? *arXiv:2210.15424.* arXiv: 2210.15424 [cs]. http://arxiv.org/abs/2210.15424 (2023-04-09) (2022-11).

41. *RedPajama-Data: An Open Source Recipe to Reproduce LLaMA Training Dataset* Together. 2023-05. https://github.com/togethercomputer/RedPajama-Data (2023-05-03).

42. Tirumala, K., Simig, D., Aghajanyan, A. & Morcos, A. S. D4: Improving LLM Pretraining via Document De-Duplication and Diversification. *arXiv:2308.12284.* doi:10.48550/arXiv.2308.12284. arXiv: 2308.12284 [cs]. http://arxiv.org/abs/2308.12284 (2023-09-24) (2023-08).

43. Gaddipati, S. K., Nair, D. & Plöger, P. G. Comparative Evaluation of Pretrained Transfer Learning Models on Automatic Short Answer Grading. *arXiv preprint arXiv:2009.01303.* arXiv: 2009.01303 (2020).

44. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J. & Lowe, R. Training Language Models to Follow Instructions with Human Feedback. *arXiv:2203.02155.* doi:10.48550/arXiv.2203.02155. arXiv: 2203.02155 [cs]. http://arxiv.org/abs/2203.02155 (2023-02-16) (2022-03).

45. *Huggingface Transformers* 2023. https://huggingface.co/docs/transformers/index (2023-09-16).

46. Dey, N., Gosal, G., Zhiming, Chen, Khachane, H., Marshall, W., Pathria, R., Tom, M. & Hestness, J. *Cerebras-GPT: Open Compute-Optimal Language Models Trained on the Cerebras Wafer-Scale Cluster* en. 2023-04. https://arxiv.org/abs/2304.03208v1 (2023-04-08).

47. Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T. & Zettlemoyer, L. OPT: Open Pre-trained Transformer Language Models. *arXiv:2205.01068*. arXiv: 2205.01068 [cs]. http://arxiv.org/abs/2205.01068 (2023-02-01) (2022-06).

48. Workshop, B. *et al.* BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. *arXiv:2211.05100*. doi:10.48550/arXiv.2211.05100. arXiv: 2211.05100 [cs]. http://arxiv.org/abs/2211.05100 (2023-02-16) (2022-12).

49. *Tatsu-Lab/Stanford_alpaca: Code and Documentation to Train Stanford's Alpaca Models, and Generate the Data.* 2023. https://github.com/tatsu-lab/stanford_alpaca (2023-09-15).

50. Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E. & Stoica, I. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *arXiv:2306.05685*. doi:10.48550/arXiv.2306.05685. arXiv: 2306.05685 [cs]. http://arxiv.org/abs/2306.05685 (2023-09-18) (2023-07).

51. Touvron, H., Martin, L. & Stone, K. Llama2: Open Foundation and Fine-Tuned Chat Models. en (2023).

52. ZXhang, Y. X., Haxo, Y. M. & Mat, Y. X. Falcon LLM: A New Frontier in Natural Language Processing. *AC Investment Research Journal* **220** (2023).

53. tii. *Falcon180B: World's Most Powerful Open LLM | Technology Innovation Institute* en. 2023-09. https://www.tii.ae/news/technology-innovation-institute-introduces-worlds-most-powerful-open-llm-falcon-180b (2023-09-28).

54. Chen, L., Zaharia, M. & Zou, J. How Is ChatGPT's Behavior Changing over Time? *arXiv:2307.09009*. arXiv: 2307.09009 [cs]. http://arxiv.org/abs/2307.09009 (2023-09-15) (2023-08).

55. jw1224. *HN: This Was Using Gpt-4-0314, Fixed and Datestamped. The Counting Shortcut Still Af... | Hacker News* https://news.ycombinator.com/item?id=37533703 (2023-09-16).

56. *Guidance* guidance-ai. 2023-09. https://github.com/guidance-ai/guidance (2023-09-09).

57. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V. & Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* **35,** 24824–24837 (2022).

58. *1rgs/Jsonformer: A Bulletproof Way to Generate Structured JSON from Language Models* 2023. https://github.com/1rgs/jsonformer/tree/main (2023-09-23).

59. Shinn, N., Labash, B. & Gopinath, A. Reflexion: An Autonomous Agent with Dynamic Memory and Self-Reflection. *arXiv:2303.11366*. doi:10.48550/arXiv.2303.11366. arXiv: 2303.11366 [cs]. http://arxiv.org/abs/2303.11366 (2023-03-29) (2023-03).

60. *DeepSpeedExamples/Applications/DeepSpeed-Chat/Training/Utils/Data/Data_utils.Py at Bae2afb8417697407ffe7cf6a21388a840679059 · Microsoft/DeepSpeedExamples* en. 2023. https://github.com/microsoft/DeepSpeedExamples/blob/bae2afb8417697407ffe7cf6a213applications/DeepSpeed-Chat/training/utils/data/data_utils.py (2023-09-16).

61. *HardwareRequirements for LLaMA and Llama-2 Local Use (GPU, CPU, RAM)* en-US. 2023-07. https://www.hardware-corner.net/guides/computer-to-run-llama-ai-model/ (2023-10-02).

62. Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H. & Ba, J. Large Language Models Are Human-Level Prompt Engineers. *arXiv:2211.01910.* doi:10.48550/arXiv.2211.01910. arXiv: 2211.01910 [cs]. http://arxiv.org/abs/2211.01910 (2023-02-16) (2022-11).

63. Conmy, A., Mavor-Parker, A. N., Lynch, A., Heimersheim, S. & Garriga-Alonso, A. Towards Automated Circuit Discovery for Mechanistic Interpretability. *arXiv:2304.14997.* arXiv: 2304.14997 [cs]. http://arxiv.org/abs/2304.14997 (2023-09-18) (2023-07).

64. Rumbelow, J. & mwatkins. SolidGoldMagikarp (plus, Prompt Generation). en. https://www.lesswrong.com/posts/aPeJE8bSo6rAFoLqg/solidgoldmagikarp-plus-prompt-generation (2023-08-28) (2023-01).

65. *Langchain Introduction* en. 2023. https://python.langchain.com/docs/get_started/introduction (2023-09-18).

66. *Haystack Introduction* en. 2023. https://haystack.deepset.ai/overview/intro/ (2023-09-18).

67. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. & Liu, P. J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. en (2020).