Extraction of numerical entities from scientific literature using large language models

Esteban Felipe Cáceres Gelvez

Master of Science

School of Informatics

University of Edinburgh

2023

Abstract

Research papers' publications keep increasing over time. Nevertheless, reading papers is a complicated task: it is hard to keep up with the number of publications, and information is scattered across sections in an unstructured PDF file. If a database with structured data from scientific papers existed, we could create several impactful applications across different STEM areas. Thus, this work proposes using Large Language Models (LLMs) to extract numeric entities from scientific text across 26 research fields. For this, we 1) discuss why LLMs among different deep-learning techniques are ideal for this task, 2) build our own dataset with a self-proposed labelling approach, and 3) fine-tune nine models using Parameter Efficient Tuning. Results show LLMs have great retrieval capabilities with recall and F1 scores of 90.17 and 87.56, respectively, outperforming, for example, ChatGPT's 10-shot learning. A web application is built with the results of the model, and 18 academics evaluated it in an online survey, demonstrating the usefulness of the product as well as the existence of a niche of users whose search experience could be highly improved.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 874307
Date when approval was obtained: 2023-07-10
The participants' information sheet and a consent form are included in the appendix B and appendix C, respectively.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Esteban Felipe Cáceres Gelvez*)

Acknowledgements

To my beloved parents, Eduardo and Ana, and my dear brother Sebastián.

Table of Contents

Chapter 1

Introduction

Advantages for science have been observed in recent years as the number of scientific papers has experienced a rapid increase. However, this surge in data presents a challenge for researchers to keep abreast of the latest findings during literature reviews. For topics with extensive volumes of research, manually reviewing each document becomes impractical and time-consuming, leading to a higher likelihood of overlooking significant results [26]. Consequently, this situation may lead to less-than-optimal or even erroneous conclusions.

Results from the scientific literature are unstructured by default. That is, metrics, numbers, and conclusions are not reported in a fixed, organized format. On the contrary, they are spread across different document sections, such as the abstract, the discussion or the conclusion sections, depending on the author's chosen structure. Furthermore, research papers are published in PDF files, a file format that is not optimal for performing automatic analysis or for extracting data easily. Tables, for example, that are a structured type of data, are still reported in a PDF file, difficulting the extraction and analysis of the results.

There are several useful applications if a structured database of scientific results existed. For example, suppose you're a scientific researcher in the area of materials science, and you must find -amongst all scientific repositories of papers- all polymers whose tensile strength exceeds 20 MPa. A normal approach would be to type some keywords in a search engine, get a list of papers and then browse each one of the articles. Then, for each article, if the quantity is not reported in the abstract, you would need to go to the discussion or conclusions section. This is definitely a task that will take plenty of time for you as a researcher. A structured database integrated with a search engine would accelerate this process. Also, if we extracted this information faster, we could

accelerate the design and discovery of new compounds, drugs and materials due to the recent adoption of machine learning and data-driven approaches in materials science to guide experiments [5].

Having results stored in a structured manner could also bring benefits outside academia. For instance, if a plastic bags manufacturer intends to shift to biomaterials instead of plastic, they could use an academic search engine to ask specific questions like ”What is the highest reported tensile strength of a biofilm?” and receive precise answers, like 420 MPa, rather than a list of papers. Or, if a government representative wants to know how good machine learning models are in health applications for a possible pilot project, he or she could ask this search engine to retrieve all F1 scores or precisions or accuracy scores in papers related to health and medicine and immediately see scores above 95%. A search engine of this kind could bridge the gap between public government, the private sector and academia.

Computer science techniques have been used to retrieve data from scientific papers automatically. Among these methods, text mining (TM) has emerged as one of the most commonly employed techniques. Text mining, a specialized area within natural language processing (NLP), aims to develop algorithms capable of interpreting textual patterns and extracting meaningful information comprehensibly for humans. Furthermore, recent advancements such as ChatGPT have been harnessed to carry out information extraction tasks, leveraging the capabilities of Large Language Models (LLMs). Here, models are trained to generate text given a sequence of words or sentences.

In this study, we propose using Large Language Models to create a system that automatically extracts three key pieces of information from the abstract of a scientific paper: the quantity, the corresponding unit of measurement, and the reported property or variable. To accomplish this task, we propose a new way of labelling text so that an LLM can successfully extract text in a structured way. Additionally, we intend to utilize our model to index a substantial dataset of 270000 papers, thereby creating an interactive search engine that allows users to engage with the extracted results.

1.1 Problem statement

We have identified one problem regarding the recent usage of AI and ML techniques in the information extraction area. These models, as we will notice in the literature review, focus on only one domain. This approach presents a significant challenge since creating

separate models for each subtopic within the vast scientific literature is impractical. The considerable training and labelling efforts required for such projects often become prohibitive. Moreover, these specialized models may encounter difficulties when dealing with scientific papers from diverse areas. To address this limitation, our research aims to train and evaluate our model in papers across 26 different areas of science. The primary objective of our study is to develop a more general model capable of efficiently and accurately extracting the mentioned entities across these domains. Therefore, we will build an artificial intelligence model that will extract all variables, numeric values and units of measurement present in STEM-related datasets built by ourselves.

In addition, we will try to answer the following questions. First, which method, in-context learning or parameter-efficient fine-tuning, performs best for information extraction regarding F1 score, precision and recall? Also, we will discuss how the dataset size and the number of parameters influence the model performance for our task.

1.2 Research Hypothesis and Objectives

The primary objective of this project is to develop and evaluate a model capable of detecting all reported quantities within a given scientific abstract. For each identified quantity, the pipeline will extract the numerical value, the corresponding unit of measurement, and the variable or property being measured. For example, in the sentence: "the reported tensile strength for the material is of 948 MPa", the model should detect one variable, and for that variable, it will extract three elements: its name (tensile strength), its value (948) and its unit of measurement (MPa). Please refer to the dataset section for a deeper explanation of this. To accomplish the mentioned goal, the following activities are proposed:

- Create a dataset with a list of sentences from abstracts across the different areas of research selected.
- Label each one of those sentences for model training.
- Train several Large Language Models using the created dataset.
- Test the models' performance and generate evaluation metrics using the test data.
- Build a search engine where users can explore the outputs and detections of the model.

- Survey academics and students so they evaluate and give feedback on the search engine and its usefulness.

We hypothesise that large language models can generalize correctly using a relatively small set of training data, no matter the variety of the data.

1.3 Motivation

This project was developed at a time when AI research was advancing rapidly. Only a few months ago, ChatGPT was released, creating a breakthrough in computer science and society in general. The advent of these large language models has opened up new avenues for developing innovative scientific search engines. For instance, Bing now generates AI-generated answers directly related to the user's query. Websites like Connected Papers offer results in the form of graphs displaying paper similarities, relevance, citations, and publication dates. Elicit, another platform, delivers specific answers to natural user questions, exclusively utilizing open scientific papers as sources. Moreover, Elicit provides AI-generated abstract summaries and a diverse range of metadata for each paper. These advancements illustrate the positive evolution of search engines. Thus, we ourselves want to propose a new perspective on how scientific papers should be explored.

Most previous works focus only on creating a model and exposing evaluation metrics inherent to the model itself. Furthermore, only a few papers create a user-friendly way to interact with the data or the outcome of the system. Thus, we propose creating a web application for exploring data, as well as applying a survey so that real academic users let us know their opinions on the website and the results.

By addressing these research objectives, we aim to advance the state-of-the-art in information extraction from scientific literature significantly and enhance the utility of large language models for this task.

Chapter 2

Background

2.1 Concepts

There are some key concepts involved in the development of this project. In this section, we will go through them. First, we will discuss how Large Language Models (LLMs) work, as well as show an overview of the techniques that exist for fine-tuning them and some common tasks that an LLM may solve.

2.1.1 Evolution in language models research

A language model (LM) [10] is a probabilistic representation of natural language capable of estimating the likelihood of word sequences. It derives these probabilities from text datasets on which it underwent training. In simpler terms, it can predict the next word in a sequence based on previous words. It has been proven useful in tasks such as machine translation, speech recognition and text generation [7]. In [43], authors state four research areas around language models. We will discuss each one of them in the following paragraphs.

The first one is Statistical Language models (SLM), where the predictions are based on the Markov assumption; that is, the forecasting is grounded in the recent context. A commonly used example of SLM is N-gram models [2]. Although useful in some applications, N-grams pose a problem when trying to take into account more context or a longer history of words when making predictions. As the order of the language model increases, the number of possible combinations of word sequences also increases, leading to more transition probabilities that need to be estimated. This can make the estimation process more complex and computationally demanding.

A second area is Neural Language models (NLM). Here, neural networks are used to understand and model the probability of word sequences. Furthermore, they bring a key innovation, which is the use of distributed representation of words [4]. In traditional statistical language models, words are represented as discrete entities. This can be problematic, as it does not consider the meaning of words or how they relate to each other. NLMs, on the other hand, represent words as vectors of real numbers, allowing the neural networks to understand the interactions between words better. Word2vec [25, 24] became one of the most popular and successful methods for learning word embeddings. Nonetheless, within a word2Vec embedding, the word's positional information is disregarded, resulting in the amalgamation of diverse contexts of a given word into a singular vector [40]. So, in the sentences i) "We sat on the river bank and had a picnic" and ii) "I asked the bank for a loan", when word2vec tries to create the vector embedding for the word "bank", it will collapse the two different contexts into one single vector.

Because of this drawback, a different approach emerged, representing the third area of research: Pre-trained language models (PLMs). In this area, the aim is to capture context-sensitive word representations. This was achieved at first with ELMo [27] by initially pre-training a bidirectional LSTM (biLSTM) network, as opposed to acquiring static word representations. This Bidirectional LSTM (BiLSTM) is a type of recurrent neural network (RNN) that can process data sequences in both directions. Later on, BERT would be proposed. BERT [9] (Bidirectional Encoder Representations from Transformers) is a pre-trained model based on the transformer [37] architecture. It was trained on an extensive, diverse corpus that offers the flexibility of fine-tuning it for NLP tasks. In fact, the model itself has great semantic capabilities for general tasks. Unlike Word2Vec, BERT leverages contextual information rather than relying on context-independent embeddings. Examples of PLMs are GPT-2 [30] and BART [21].

Finally, a fourth research category is Large Language Models (LLMs). An LLM is a large-sized PLM with billions of parameters that were trained on big unlabelled datasets using self-supervised learning [32]. Research shows that increasing the model size of these PLMs results in outstanding semantic abilities that allow them to solve complex tasks that require high reasonability [43]. Thus, rather than being specialized for a single NLP assignment (e.g. sentiment analysis, named entity recognition, text classification), LLMs are crafted to deliver competence across a diverse spectrum [38]. An LLM can, for example, answer factual questions and get similar benchmarks as a supervised approach [6]. An outstanding application of an LLM is ChatGPT, which

adapted GPT-based models to dialogue so they were easily accessible to humans in a conversation.

2.1.2 Models

For our task, we will be leveraging the semantic capabilities of these LLMs. Specifically, we used the subset of models LLaMA and OPT. The first one, LLaMA (Large Language Model Meta AI) [35], refers to a set of different LLMs released by Meta in February 2023. Model sizes vary from 7B parameters to 65B. They focused on releasing smaller models trained on more tokens (between 1 and 1.4 trillion), given that previous models required a high amount of computation resources for fine-tuning or pre-training. In July 2023, Meta AI released a second version of LLaMA, called LLaMA 2 [36]. This model was trained in up to 2 trillion tokens and performed similarly to OpenAI's GPT3.5. Furthermore, it aligns more with human evaluation and feedback, and at the same time, Meta AI claims the model is safer to use without compromising performance and usefulness. OpenLlama [13] is a replication of LLaMA original model. Authors report similar results to the original implementation while fully releasing the weights.

The second family of models is Facebook's OPT [42]. OPT (Open Pre-trained Transformer) is a suite of models that ranges in number of parameters from 125M to 175B. They were first released in June 2022. OPT models adopt a decoder-only architecture, omitting the encoder layer. This choice simplifies training and fine-tuning for specific tasks. We will be using for this project OPT-IML-based models [17], an instruction-tuned version of OPT that was fine-tuned on more than 1,500 NLP tasks. This allows this model to generalize and better perform on new, unseen tasks, something that will be definitely useful for this project.

2.1.3 Emergent abilities

LLMs have shown a new set of skills called emergent abilities. These didn't occur in previous PLMs and are useful for more complex NLP applications. There are three main emergent abilities. The first one is In-context learning (ICL) [6]. It is a technique that allows LLMs to learn new tasks from a few examples. ICL works by providing the LLM with a prompt, which is a natural language description of the task to be performed. The prompt typically includes a few examples of the desired output. The LLM then uses its pre-trained knowledge to infer the underlying concept of the task and to generate the correct output. This is all done without additional training or gradient updates. We will

```
Your goal is to extract structured information from the user's input that matches the
form described below.

This is a sample of the schema you will be extracting information for:
Array{
 property_name: string // The name of the variable or property
 property_value: number // The value of the variable
 property_unit: string // The unit of measurement of the variable
}

Example input:

The tensile strength and Young’s modulus of the gelatin/OI-BM/metronidazole fibres were 2.0
MPa and 25 MPa, while those of the gelatin fibres were 1.0 MPa and 17 MPa.

Example output:

property_name|property_value|property_unit
tensile strength|2.0|MPa
Young’s modulus|25|MPa
tensile strength|1.0|MPa
Young’s modulus|17|MPa

Input: The fabricated lead-free film, Ba 0.85Ca 0.15Zr 0.1Ti 0.9O 3 (BCZT),
shows a high piezoelectric coefficient d 33 = 209 ± 10 pm V-1 and outstanding flexibility of
maximum strain 2%

Output:
```

Figure 2.1: Prompt passed to an LLM using the ICL technique. Input sentences were taken from [23, 29].

use ICL in ChatGPT and compare the results vs. our fine-tuned models. Please refer to the results section for more details. We may see an example of ICL in Figure 2.1.

Another emergent ability is step-by-step reasoning. This was introduced through the chain of thought concept by [39]. Chain of thought is a technique used to improve the ability of LLMs to perform complex reasoning. It works by providing the LLM with a prompt that includes a few examples of the desired output, along with a chain of reasoning that leads to the output. The LLM then uses this chain of reasoning to generate its own output. This ability, however, seems to be stronger in models with more than 100B parameters [39]. We may see an example of a prompt using a chain of thought in Figure 2.2.

Instruction tuning is another emergent ability reported. It consists of fine-tuning a model with a dataset that contains multiple different instructions. This dataset doesn’t provide any examples for each task. Thus, models are expected to obtain a better generalization ability. Research has shown how models perform well in tasks described as an instruction [31]. We may see an example of an instruction dataset in Figure 2.3.

```
The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1. True or False?

Here is a chain of reasoning that leads to the answer:

1. The odd numbers in the group are 9, 15, and 1.
2. 9 + 15 = 24.
3. 24 is an even number.
4. Therefore, the odd numbers in the group add up to an even number.
```

Figure 2.2: Example of a prompt with a chain of thought we could pass to an LLM so it learns how to solve math problems.

```
[
  {
      "instruction": "What are the three primary colors?",
      "input": "",
      "output": "The three primary colors are red, blue, and yellow."
  },
  {
      "instruction": "Identify the odd one out.",
      "input": "Twitter, Instagram, Telegram",
      "output": "Telegram"
  }
]
```

Figure 2.3: Two samples from a dataset that follows the instruction tuning approach.

For our project, we will be using instruction tuning. However, we will maintain the same instruction, which will be to extract the numeric entities from the text. You may refer to the methodology section to explore how we built our dataset.

2.1.4 Fine-tuning

Given the huge amount of parameters LLMs have, they pose a problem when researchers want to fine-tune these models on downstream tasks. Therefore, literature has proposed different ways to tackle this issue. One of them consists in reducing the number of trainable parameters. Techniques using this approach are called Parameter-Efficient Fine-Tuning Methods. Some of them are adapter tuning [15], prefix tuning [22], prompt tuning [20], and LoRA [16], which is the one we will focus on.

LoRA (Low-Rank Adaptation) immobilises the LLM's existing weights and in-

troduces trainable rank-decomposition matrices. Specifically, two matrices, A and B, are injected. Matrix A has dimensions (r x d), where 'r' is the rank and 'd' is the dimensionality of the matrix [16]. The rank measures the effective number of linearly independent rows or columns in the matrix. By having a rank smaller than 'd', the matrix is represented using fewer basis vectors. This is akin to capturing the most important features of the original matrix while discarding less important information. Matrix B complements matrix A and has dimensions (d x r). It is a counterpart to A and contributes to forming the final weight matrix 'W'. This new weight matrix, thus, is defined as W = BA. By leveraging rank-decomposition matrices, LoRA can capture essential information from the original model and optimize its performance with a reduced parameter count while accomplishing performance levels similar to full-finetuning. Ultimately, this also represents savings in terms of storage and memory usage.

QLoRA [8], a further parameter-efficient tuning method, proposes a new quantization technique to reduce the memory usage of the LLM without degrading performance. This is done by quantizing the weights of the pre-trained model to 4 bits. Authors introduced, for example, the 4-bit NormalFloat (NF4) data type, which optimally preserves information for normally distributed weights. Also, they employed Double Quantization to significantly shrink the memory usage average by quantizing the quantization constants. The authors state that using a small high-quality dataset leads to state-of-the-art benchmarks while only using one consumer GPU.

2.2 Related Work

We previously discussed what Large Language Models are and their capabilities, as well as how to fine-tune them. Now, in this subsection, we will mention and discuss some works that have used deep learning and Large Language Models for tasks similar to ours; that is, for extracting information or entities from scientific or academic texts.

There have been several approaches in literature to extract information in a structured way from scientific papers. Most of them model the problem as a Named Entity Recognition task, where entities are extracted and tagged according to the chosen model. A first approach is using rules and dictionaries. Here, authors try to find tokens that are stored in a predefined, manually-built dictionary, that contains rules or terms that refer to a specific entity. This approach has several disadvantages. First, entities or examples that are not present in these artefacts won't be matched, causing the models to have low

recall [14]. Furthermore, you may need a high amount of memory to store and access these lists of terms [5]. Also, manually building the dictionaries is not sustainable given that new terms may appear in time and that they are built for one specific domain, meaning they couldn't be used in a different area [14].

A second approach uses deep learning models and BERT to extract and tag the entities from text. In the area of materials science, for example, several authors have pre-trained BERT models on specific domain datasets, such as inorganic and organic materials [40], ceramics and alloys [33]. SciBERT is a recent example of this approach. It is a pre-trained LLM based on BERT, that was trained on 1.14 million full-text papers. The corpora have texts from the biomedical and computer science domain, mainly [3]. In [], authors trained 2 SciBERT models. The first one extracts the quantity value, and a second model extracts the measure or variable represented by that value using a question-answering framework. For example, given the sentence "the accuracy of the model is 89%", the first model would extract "89%" the second model received the question: "What is the measured property of 89%", answering thus with the word "precision". They report results from a self-built dataset. F1 scores are 0.944 and 0.811 for quantity extraction and measured property extraction, respectively. This work is an innovative approach to the task of extracting data from scientific papers. However, it required training two different models, as well as annotating a dataset that has 7316 quantities and 4606 measured properties. This is a difficulty recognized in previous works [19], given that some tasks or applications are unlikely to collect a high number of samples; annotating requires investing time and also because domain experts may be needed. Our approach, on the other hand, uses one single model for the task, and our dataset is smaller, containing only 1695 variables and still obtaining high F1 scores. You may find more details in the methodology and results sections.

Although LLMs are a very new topic in research, some recent works have proposed using LLMs instead for extracting information from scientific corpora. In a first work [28], authors proposed a method that, using different LLMs, generates a database of a desired/specific property. Their approach was the following. First, sentences were extracted from abstracts, and those that contained numerical values were kept. Then, they passed each sentence to an LLM to classify -using a zero-shot approach- whether a sentence actually contained a numerical value for the desired property. After this, an optional step is proposed. It consists of having a human double checking the binary classifications made in the step before and then creating a dataset with the reviewed data to fine-tune another model. The last step involved extracting the data point itself.

An LLM initially did this and then checked by a human. Results reported show a 90% recall and a precision close to 90% as well when using a fine-tuned GPT-3.5, for the task of classifying if a sentence contains the property. At the same recall, the precision decreases to the 80% range when using DeBERTaV3 or Bart. (Given results are only reported in the figures and not in tables, we cannot get an exact metric for precision and recall). This work does not report results for the step of extracting the information itself since it focuses mainly on the binary classification task. Additionally, its objective is to extract one specific, predefined variable only. Our proposal is to create a model that, given an abstract, extracts all the numeric variables reported in it. Furthermore, they used GPT3.5, which has 175B parameters [12]. This requires high computation costs. The biggest model we tested has 7B parameters, given we want the deployment of this model to be feasible in terms of computation and costs.

In another research [11], authors propose a sequence-to-sequence approach for extracting formulas, applications and materials/component names from scientific papers. They fine-tuned GPT-3 with 650 samples. Each sample contained a prompt and a completion. The prompt would contain a passage of text and the output or completion of a JSON schema, with all the possible desired entities and their respective properties. Fine-tuning was done using the OpenAI API. Test scores were performed on 65 random samples. Results show an average extraction recall across all extracted entities of 0.74; an extraction precision of 0.84, and a F1 score of 0.77.

The authors discuss how specific entities, such as chemical compound names, get low scores when extracting them, given the ambiguity of naming an entity. For example, suppose the ground truth for an entity name is “Fe particles”, and the language model predicted “Iron particles” instead. They both refer, at the end, to the same concept. However, if one uses a word-matching approach when evaluating the input and the output, scores will be inaccurately low. Authors evaluated the predictions using both a word-matching approach and manually, through field experts. They report increased F1-scores of up to 0.4 when considering the experts’ opinion. This ambiguity is something we will discuss in our work. We propose calculating the Levenshtein radio between the truth and the prediction to detect whether they both refer to the same concept, thus allowing us to fasten the evaluation process without requiring the need for manual, expert evaluation. Finally, they report a parsable percentage of 95% success rate. Results suggest an error in parsing happens when the completion token limit is reached. When this occurs, the model stops at the middle of the sequence and stops generating, for example, a closing bracket of the JSON schema. Thus, we propose a simpler labelling

approach, very similar to Comma Separated Values, but instead, we will use a vertical bar as the separation token.

Chapter 3

Methodology

3.1 Dataset creation

First, to create the dataset, we downloaded abstracts from 26 different scientific fields using the Science Direct API. Only abstracts with at least one numeric character were saved. Originally, we aimed to download 400 abstracts per topic, but after applying the filter, the number of abstracts per field decreased. However, we ended up selecting 185 abstracts per field so we could keep a balance between the different areas of research.

After having our selected abstracts, we split them into sentences. The separator string was: ". ", representing the actual separation between sentences in a paragraph. Then, each sentence was split into words using the space separator (" "), and if at least one of those words was a number, then the sentence was kept. Two further considerations were taken into account: all parenthesis were removed since usually there's no separation between them and a number. This could lead to actual quantities not being detected. For example, consider the following sentence: "tensile strength was higher in the second experiment (241 MPa)". Since we split by the space character, our term that includes the quantity would be "(241 " which wouldn't be successfully detected and parsed as an instance of a number. The next consideration was that if the sentence contained at least one percentage symbol "%", then the sentence was kept. The reason is very similar to the first consideration. Let's consider the sentence "increase in precision was of 18%". When splitting the 18, we would actually get "18%", which -again- wouldn't be detected as a number. All these splitting and preprocessing steps led to having 6439 sentences.

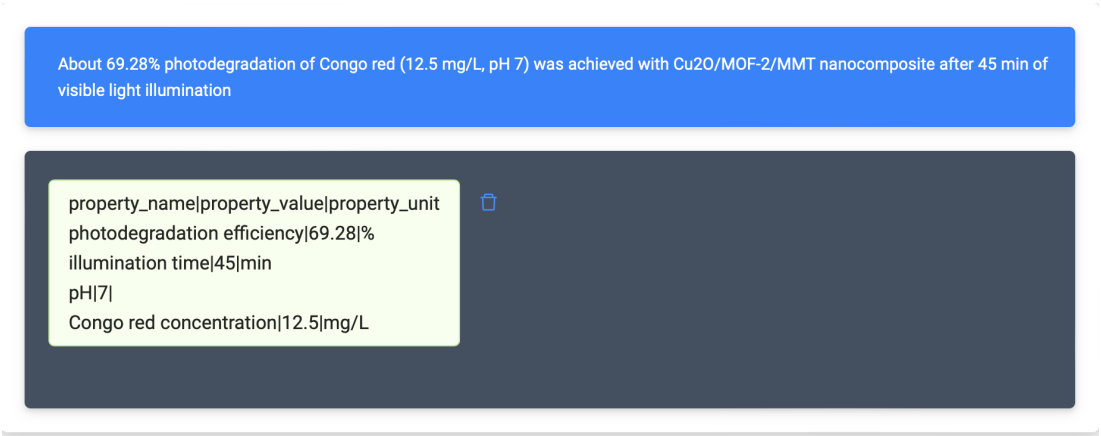

Figure 3.1: Example of how a label looks like for a given sentence in Label Studio. The text in the blue box is the sentence (sample) and the text in the green box is the label.

3.2 Dataset labelling

Now, having ready our bank of sentences, we started labelling. We used the tool Label Studio for this task. Specifically, the "Supervised Language Model Fine-tuning" interface. Here, one sentence represents one sample. For each sentence, there could be 0 or more than 0 variables present. All variables were written down manually. For each variable we wrote three properties: its name, its numeric value and its unit of measurement (in that order, from left to right). The vertical bar character separated each property, and each variable was written in a new line. We may see what the labelling looks like for a sample in Figure 3.1. The sample sentence is the text inside the blue box, and the labelling is the text inside the green box. All labels had a header: `property_name|property_value|property_unit`.

Using the vertical bar as a separator has several advantages over other options. First, this character rarely appears in scientific papers; thus, the model can understand that its function separates the three properties. Also, it uses the least possible amount of characters to structure the text. Other formats, such as JSON, would need to add more characters, such as brackets and double quotations. Another big benefit of this notation is that labelling time is reduced significantly compared to classical deep learning approaches such as relation extraction or named entity recognition. For our case, we used ChatGPT to generate a draft version of the label, and then all labels and quantities were double-checked by us. This support for the labelling process wouldn't have been possible if relation extraction or named entity recognition were used (at least not in Label Studio).

Out of all the 6439 sentences, we labelled 1000 random sentences. Given that there can be more than one variable per sentence, we had 1695 variables labelled. However, as we have mentioned, for our task, one sample is one sentence (and not one variable).

These were the standards and rules followed when labelling:

- The variable name had a minimum of 2 words and a maximum of 5 words. This is to keep a good representation of the variable. However, as we will discuss later on, the naming of a variable is highly subjective.
- In the case there was no unit of measurement, which is likely, we just didn't add anything to the end, hoping the model would catch that.
- Variables that represent objects or counts were rewritten as the unit of measurement. For example, for the sentence "800 research articles were reviewed", the corresponding label would be: `research articles reviewed|800|articles`
- Removed commas from thousands (2,000) or dots that are thousands separator (139.500).
- If a range was reported, create one registry per lower and upper range. For example, for the sentence "the frequency range for the experiment was of 120 Hz –100 kHz", we would create two different labels: `experiment frequency|120|Hz` and `experiment frequency|100|kHz`. A reason for this is that, as we noticed, the lower amount has a different unit than the upper amount.
- Ideally, the value -which is the property in the middle- should always contain only a numeric representation. This rule supports the statement above.
- The p-value was always written in the following form: `p-value|0.5|`.
- Error ranges in numeric values were not included. So, for example, for the sentence "the tensile strength was of 20.33 ± 6.35 MPa", its corresponding label is `tensile strength|20.33|MPa`.
- Abbreviations were included if available. For instance, for the sentence "the area under the curve (AUC) was 0.56", the corresponding label would be: `area under the curve (AUC)|0.56|`.
- Superscript was included even if not present in the original sentence. So if the sentence were "the velocity was 25 km h-1", the label would be `velocity|25|km h^-1`.

- Years were omitted. Thus, they are expected not to be extracted.

3.3 Dataset preparation for training

Once all sentences were labelled, the next step was to create the prompt structure. Prompts are key in making the models return the desired response. A prompt is a string; it's the text we will use to communicate with the model. Thus, we need to transform the labels properly so a large language model can understand what our task is about. For this, we used the format created by Stanford University in the Alpaca dataset [34]. This is an instruction-following dataset where the model is trained to accomplish specific instructions, which is what we need. In the Alpaca format, each sample that will be passed to the model has four fields:

- Header: an introductory text where we start giving context to the model of what we want it to accomplish. For our case, all samples have the same header: "Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request."

- Instruction: a string outlining the desired task for the model. For our case, all samples have the same instruction, which is: "Extract the property names, quantities and units."

- Input: a string providing context or input for the task. In our case, an input is a sentence containing numeric variables.

- Output: a string representing the expected response to the instruction, given the input. In our case, this represents the actual labels we created with Label Studio.

We may see an example of a prompt in figure 3.2.

As we mentioned previously, our dataset has 1000 samples (sentences). We split this dataset into training and test sets. For the training set, we created two variants: the first one had 800 samples, and the second one had 400 samples. This was done to test the influence of training data on the model's performance. For both cases, the test set was the same, and it had 200 samples. Samples were randomly selected. We can see the number of samples per topic in the table A.1. It's worth noting, though, that we won't perform evaluation metrics per topic because we consider that the structure for a sentence that reports a variable is quite similar across scientific fields. Furthermore, in

```
Below is an instruction that describes a task, paired with an input that provides further
context. Write a response that appropriately completes the request.

### Instruction:
Extract the property names, quantities and units.

### Input:
The optimum reaction condition for the synthesis of NCC was 50 °C with 45 wt% acid
concentration for 60 minutes due to its high cellulose content (85.6 %)

### Response:
property_name|property_value|property_unit
optimum reaction condition|50|°C
optimum acid concentration|45|wt%
reaction time|60|minutes
cellulose content|85.6|%
```

Figure 3.2: Example of how a prompt looks like for a given sentence and label.

table A.1, you may see how some fields ended up having more sentences than others, even though for each field, we initially selected the same amount of abstracts (185). This occurs because, in reality, not all fields have the same amount of sentences reporting a quantitative variable or number. Some might be more qualitative-oriented. Since this is the real representation of the data, we decided to maintain it when training and testing the model. The dataset can be found in the attached materials.

3.4 Advantages of our labelling approach

Before moving on, we would like to show some advantages of using our labelling approach for this task. As was shown in the background section, our task (extracting numeric entities from scientific abstracts) could be solved using relation extraction. However, if we had used this approach, several problems would have been raised. In figure 3.3, we can see an example of a label using the relation extraction approach. There are two main steps in this process. First, we need to do named entity recognition, that is, select those words or a set of words that belong to a specific entity (variable, quantity and unit of measurement). Then, we need to assign the value for each variable and the unit of measurement for each value. This involves several clicks and interactions within the Label Studio tool, which in the end, represents an increase in the time that it takes the user to label one sample. Besides this, the following are some scenarios where

labelling with the relation extraction approach isn't feasible for our task:

- Variable is not specifically written: let's consider the following sentence: "the samples were stored at 23°C for 12h". For that case, the variables for 23°C and 12h are absent in the sentence. However, we might easily infer them as "storage temperature" and "storage duration", respectively. If we were using the relation extraction labelling approach, it wouldn't be possible to insert these two variable names since they must be present in the text in the labelling tool. On the other hand, by using our approach, we just write it down in the labels, just as we showed in 3.1. The labels would be: `storage temperature|23|°C` and `storage duration|12|h`.

- Ranges in the quantities: please refer to the example shown in figure 3.3. You may see how the "cell density" variable has two values assigned to it, a lower and upper bound. We mentioned that in those cases, we would create one sample per each bound. This means "2.5 x 105" is one value, and "20 x 105" is another. It is not possible to showcase this in the labelling process if relation extraction was used.

- Adding fixes: we previously stated that we removed the comma from 2,000; or the dot when it was used as a thousand separator; or that we would always add the superscript if it didn't exist (just like in figure 3.3). Adding these fixes is not straightforward in the labelling tool if relation extraction is used.

- Values or units are represented as text: consider the sentence "twenty-eight people were divided into seven groups". We previously stated that the values we extract should always be a number. Here, however, they are represented as text. Thus, the correct labels for this sentence are: `number of people|28|people` and `number of groups|7|groups`. Another example occurs in the sentence "63 percent of the people answered the survey", where the label should be `percentage of people that participated|63|%`. It's not straightforward to do this in the labelling tool.

- Variable naming could be improved: in some sentences, the values refer to deltas, such as increases, reductions, and improvements, so they don't represent the value but a delta of the value. Thus, these words should be included so the extracted variable is not misleading. So, in the sentence "the recognition accuracy has been improved by 0.30%", the label should be `recognition`

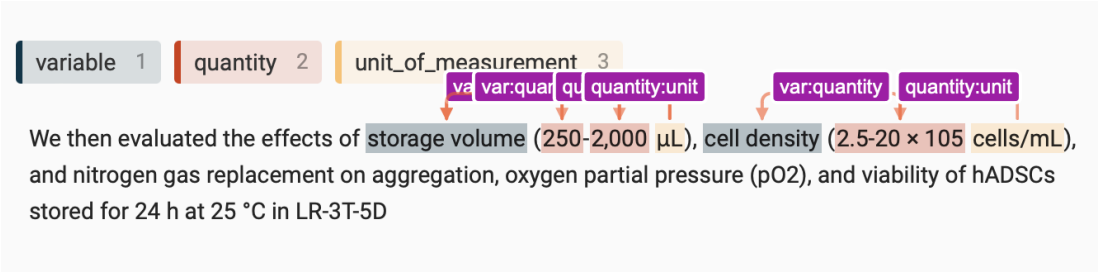

Figure 3.3: Example of how labelling should be done if relation extraction would have been used. (Sentence taken from [18]).

`accuracy improvement|0.30|%`. Another example occurs in ”temperatures in the experiment reached up to 98°C”, where a correct label should be `maximum temperature|98|°C`. Again, adding these clarifications in Label Studio using relation extraction is not immediately possible, given the words ”improvement” or ”maximum” are not explicitly written in the sentence.

3.5 Model training

A total of 9 different models were fine-tuned for this task. The training scheme used was Causal Language Modeling (CLM), where the model is trained to predict the next token in a sequence given the previous tokens. It is suitable for text generation, which is what we require the model to do for our task. We used the Trainer class from HuggingFace’s Transformers library.

A list and specifications of each model can be found in table 3.1, where we describe the characteristics of each model. First, we find a unique ID for each model, that we will use in some parts of the text. We also see the model name; the number of parameters (in billions) of the model; the training technique and the precision used for loading the weights during training; the number of samples used during training and finally the precision of the weights when exporting and using the model for inference. For example, for model with ID 4: it uses OpenLlama as the architecture, more exactly, the version with 3B parameters. The model was trained with 800 samples using LoRA. For training, the weights were loaded with 8-bit precision. When fine-tuning was done, the LoRA adapters were saved separately. Later on, for inference, the model was reloaded again in float16 or float32 precision, and then it was exported as it is stated in the `Exported` column of table 3.1.

ID	Name	Parameters (B)	Training	Samples	Exported
1	OpenLlama	7	QLoRA - 4bit	800	float16
2	Llama2	7	QLoRA - 4bit	800	float16
3	OpenLlama	7	QLoRA - 4bit	400	float16
4	OpenLlama	3	LoRA - 8bit	800	float16
5	OpenLlama	3	LoRA - 8bit	400	float16
6	OPT	1.3	LoRA - 32bit	800	float32
7	OPT	1.3	LoRA - 32bit	400	float32
8	OpenLlama	7	QLoRA - 4bit	800	4bit
9	OpenLlama	3	LoRA - 8bit	800	4bit

Table 3.1: Characteristics of the fine-tuned models

Regarding the architectures, we are using three different models: OpenLlama [13], which is an open-source replication of Llama [35]. This model was selected because it was, at the moment, one of the few high-performing LLM models whose weights were fully available. We requested access to the original LLaMA weights since they are only provided for researchers on case-by-case study. However, permission wasn't answered. We also used OPT models from Meta AI [42], since they offered a small model (1.3B) in terms of parameters, useful for evaluation purposes of this work. Finally, we used LLaMA 2, given that access to its weights was easily granted. Furthermore, we used two Parameter Efficient Tuning techniques to fine-tune the models: QLoRA [8] and LoRA [16]. For models using the QLoRA technique, the quantization type is nf4, and the compute type is bfloat16. Double quantization was applied. Finally, you may find a detailed list of the chosen hyperparameters for each model in A.2 and A.3.

3.6 Model inference

Inference was done on all models on the test set, which contained 200 samples. We used two frameworks. First, the serving library vLLM, which uses efficient memory management using paged attention, offering state-of-the-art serving throughput. This was useful for the 7B models, which would not normally fit if we were to use another approach. Second, we use the Transformers library. The generation parameters were the same for all models and are as follows: temperature=0.2, top_p=0.90, top_k=50, max_tokens=720, presence_penalty = 0.0 (no penalty) and frequency_penalty = 0.0

(no penalty). Temperature, which is meant to express the randomness and diversity of outputs, was set to 0.2 given we do not expect the model to generate new words but only to extract them from the text. However, there might be occasions where we do want the LLM to come up with words that are not explicitly written in the text, just as we explained in some sections above when exposing the benefits of our labelling approach. Furthermore, top k and top p values are similar to some standard libraries implementations. Finally, it is very likely that during inference, words are repeated, given there might several reports of a same variable. Thus, we don not want to penalize the model for repeating terms.

3.7 Search engine

After training the models, the next task was to create a search engine where users could explore the results of applying this model to real data. Thus, we built a search engine where we indexed more than 270,000 scientific abstracts. These abstracts come from two main sources: a scientific papers dataset from Arvix [1] and from the Science Direct API. Once we downloaded these papers, we generated the sentences for each abstract like we did when generating the training dataset. Then, these sentences were passed to the OpenLlama 7B model, and the results were indexed in the search engine. Typesense package was used to create the index, and Streamlit was used to create the front end of the web application.

3.8 Evaluation procedure

3.8.1 Intrinsic model evaluation

We propose two evaluation approaches for our models. First, we evaluate how well they retrieved the correct variables. For this purpose, we used precision, recall and F1-score. The idea here is to understand the retrieval capabilities of the LLMs. The following are our specific definitions:

$$precision = \frac{correctly\ predicted\ variables}{total\ number\ of\ predicted\ variables}$$

$$recall = \frac{correctly\ predicted\ variables}{total\ number\ of\ actual\ variables}$$

$$f1 = \frac{2 * precision * recall}{precision + recall}$$

A 'correctly predicted variable' is a prediction that is an actual variable according to the ground truth. For example, for the sentence "84 people were interviewed in the case study", let us suppose the LLM made the following prediction:

```
property_name|property_value|property_value
number of people|84|people
study type|study|case
study place|Edinburgh|place
```

For the following ground truth:

```
property_name|property_value|property_value
number of participants|84|people
```

Then, the correctly predicted variables are 1; the total number of predicted variables is 3, and the total number of actual variables is 1. We used the Levenshtein distance ratio to detect if a prediction and a label referred to the same variable. We loop through each pair of prediction samples and ground truth samples. For each pair, we did the following: first, we extracted the value and the unit of measurement part of both the prediction and the label. Then, we calculated the Levenshtein ratio for each pair of prediction-label samples and kept the maximum score. If this maximum ratio was equal to or greater than 0.5, we assumed that that specific label was the corresponding ground truth for that prediction sample. Then, after this, we removed the prediction from the list of predictions. Predictions that were not removed at the end of the iteration represented predictions that didn't obtain a match, thus, they were hallucinations or mistakes from the LLM. Furthermore, we kept track of the ground truth labels that obtained a match, so we could calculate at the end of the iteration how many variables should have been detected but were not.

To sum up, for each sentence (sample), we calculated the precision, recall and F1 score. The final scores for each model are the average across all sentences.

After obtaining these metrics, we evaluated the models by calculating how accurately they represented the extracted variable names, values and units of measurement. For those predictions that were correctly detected as a variable, we compared each of these 3 parts with the ground truth by calculating the Levenshtein distance ratio. Here, a score close to 1 would be a perfect match, whereas a score near to 0 would imply there are important differences between the prediction and the ground truth. Following the example above, we would calculate the scores only for the first prediction out of all three, since that first is the only one correct. Thus, three similarity scores would be obtained: the first one, `number of people` vs. `number of participants`, obtaining a score of 0.63. The second calculation, `84` vs. `84`, would obtain a score of 1. And finally, the calculation for the unit of measurement would be between `participants` and `participants`, obtaining again a score of 1. At the end, all scores were averaged across all variables correctly detected. This represents the score for the sentence. And, again, the average across sentences represents the model performance.

3.8.2 In-context learning comparison

In the subsection above, we propose an intrinsic evaluation approach, where the predictions are compared against the ground truth, which is our labelled test set. Here, on the contrary, we perform a new evaluation, using in-context learning in different types (zero-shot, one-shot, two-shot, five-shot and ten-shot) to generate predictions of the test set. The aim is to test whether it is necessary to perform fine-tuning to obtain the best results or if few-shot learning is sufficient for this task. The model we used for this is GPT-3.5, specifically ChatGPT. The used prompt is shown in 3.4. In that figure, we see the example of one-shot learning. As we mentioned, we used up to ten-shot. In those cases, we just added the other examples using the same Input/Output pattern. The prompt design was highly influenced by Kor library [41]. Note: all samples used across the few-shot experiments can be found in the attached files.

3.8.3 Search engine evaluation

Finally, we performed an online survey of 18 academics. They were asked to test the search engine and answer some questions about the relevance, use cases and general feedback on the web application and on the retrieval. This was done using Google Forms after the ethics committee granted permission.

```
Your goal is to extract structured information from the user's input that matches the
form described below. When extracting information please make sure it matches the type
information exactly. Do not add any attributes that do not appear in the schema shown below.

```TypeScript

variable_extraction: Array<{ // extraction of relevant information from numeric variable
 property_name: string // The name of the variable or property
 property_value: number // The value of the variable
 property_unit: string // The unit of measurement of the variable
}>
```

Please output the extracted information in CSV format in Excel dialect.
Please use a | as the delimiter.
Do NOT add any clarifying information. Output MUST follow the schema above.
Do NOT add any additional columns that do not appear in the schema.

Input: In this study, hydrates formed under temperature gradient of 0.012, 0.020 and 0.026 °C/mm
Output: property_name|property_value|property_unit
temperature gradient|0.012|°C/mm
temperature gradient|0.020|°C/mm
temperature gradient|0.026|°C/mm
reactor volume|2|L
depressurization pressure|1.0|MPa
exhaust rate|0.11|ln/min
exhaust rate|1.07|ln/min

Input: {input}
Output:
```

Figure 3.4: Prompt for applying one-shot learning to ChatGPT.

Chapter 4

Evaluation

4.1 Models

4.1.1 Training-wise

Table 4.1 shows results regarding the training process of each model. There are several takeaways to remark from this table. First, the precision used when loading the weights has a bigger influence on training time than the number of parameters. We may see this with the 3B versions of the OpenLlama (OL), which was loaded in 8-bit and took almost twice the time to converge compared to the 7B version, loaded in 4-bit precision format. This is a big difference, even more so when you note that the 8-bit version was trained on a v100 GPU, which is faster than the T4 GPU, in Google Colab's platform. A note should be added here, too: training had to be done in a v100 for the 8-bit model, given that when using a T4, the environment would crash because of the unavailability of GPU memory.

We also remark how training time is very low for the 1.3B models, but at the same time, these models don't converge. We can see in figure 4.1 how, for both the 800 and 400 versions, the loss won't decrease, even after eight epochs. On the other hand, the other models do decrease across epochs. The most likely reason for this behaviour is the number of parameters the OPT model has. That is, 1.3B parameters might not be enough to capture the patterns across the dataset and the task itself. This was previously discussed in the emergent abilities section, where it is stated that relatively small models may not have the capabilities that LLMs have. The problem occurs even when increasing the number of epochs or changing the gradient accumulation steps: it always stops when the loss is close to 1.

	Training time (hours)	GPU used for training	Training loss
llama2_7B_4bit_800_f16	7.77	T4	0.0972
openllama_7B_4bit_800_f16	2.23	T4	0.0564
openllama_7B_4bit_400_f16	3.68	T4	0.0722
openllama_3B_8bit_800_f16	7.15	V100	0.0564
openllama_3B_8bit_400_f16	6.42	V100	0.0468
optiml_1-3B_32bit_800_f32	0.22	T4	0.9387
optiml_1-3B_32bit_400_f32	0.22	T4	1.034

Table 4.1: Results of the training process for each model.

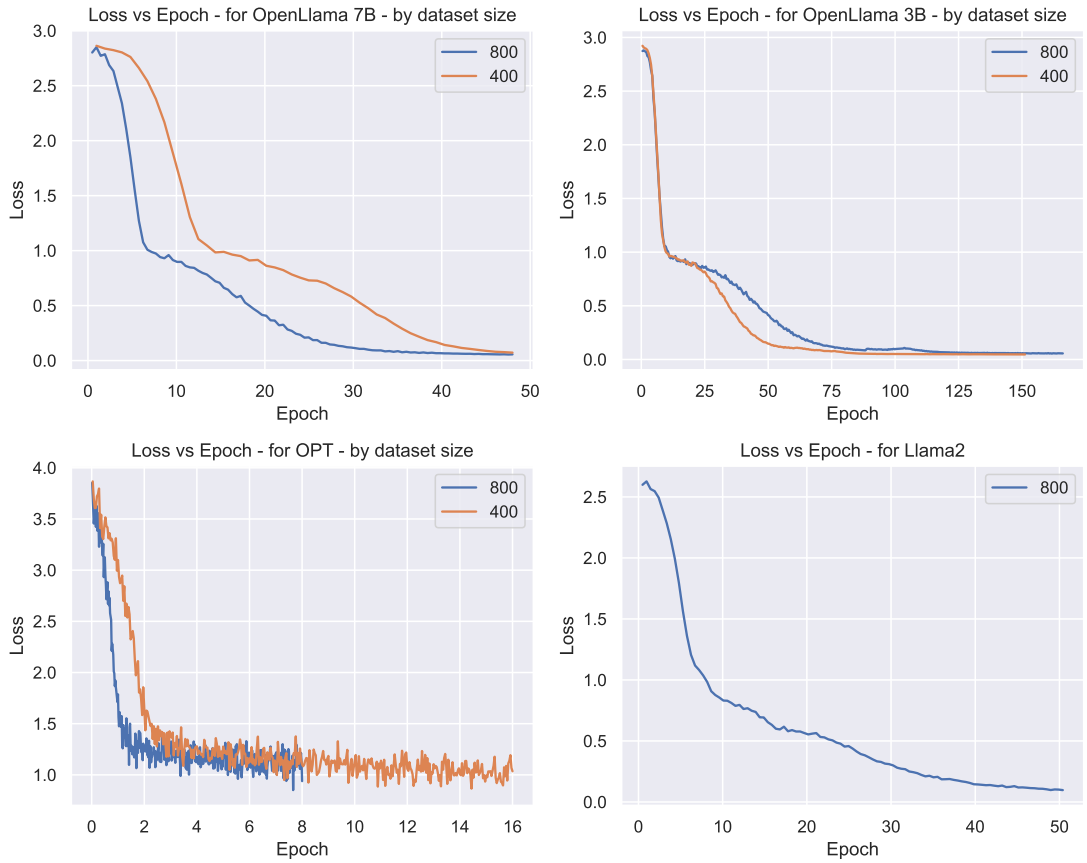

Figure 4.1: Cross entropy loss through epochs, for each model, for dataset size.

When comparing the 800 and the 400 versions, there doesn't seem to be a big difference in training: in the end, both versions, no matter the model size, seem to get the same or similar loss. However, OL 7B seems to benefit from having an 800 dataset, converging a bit faster. Nevertheless, this behaviour does not occur in the 3B version.

4.1.2 Intrinsic evaluation

As we described in the methodology, we calculated the F1 score, precision and recall for each model. For our task, it is preferable that the model obtains a higher recall over a higher precision. The reason is that if the LLM over-generates trash, further simple processing techniques might help filter them out before showing them to the user. However, missing actual variables is way worse since the LLM is actually working as a retriever. Note: the reader can find all predictions for each model in the attached files.

We may see some results of the intrinsic evaluation in the figure 4.2. We notice how there is a difference in the dataset size and in the number of parameters, with the number of parameters being the factor that affects the most precision, recall and F1. For example, the 7B version of OpenLlama obtains a recall score of 89.41, using 800 samples in the training set. The 3B version, also with 800 samples, obtains 16.91 points less, whereas the 7B version, but this time with 400 samples, obtains 7.33 points less.

Another useful insight we find is that increasing the number of samples for training seems to have a more impactful effect on models with a higher number of parameters: when increasing the samples from 400 to 800 in the 7B model, the recall score increased in 7.33 points, whereas it only increased 0.31 and 3.58 in the 3B and 1.3B model, respectively. Furthermore, regarding the best model above all, Llama 2 obtains the greatest precision (89.98) and F1 score (87.36); however, the highest recall score is seen in the OpenLlama 7B model (89.41). When looking at the results, this makes sense, given OL 7B only missed 44 actual variables. On the other hand, Llama 2 missed 61.

Regarding the similarity scores between the prediction and the labels, we may see the results in figure 4.3. We notice how, for the values and units of measurement, there is not a big gap in performance across the number of parameters or dataset size. A clear conclusion is the difference in performance for the variables' scores. This score represents the gap between how the expert named a specific value versus how the LLM named it. We discussed this ambiguity before, and previous works [11] have stated this is an issue for information extraction. We, ourselves, noted that difficulty while labelling, given the same variable can be referred to with multiple names. We stated

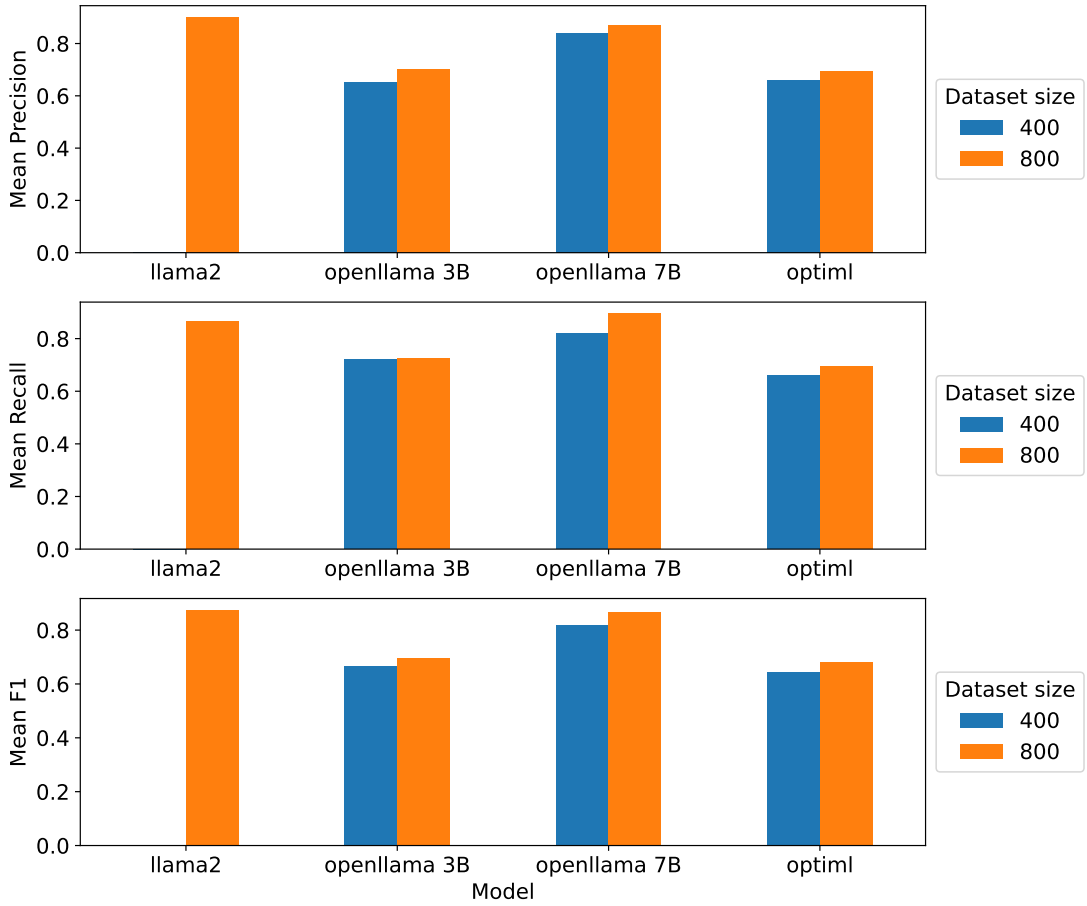

Figure 4.2: Precision, recall and F1 score for all models, grouped by dataset size.

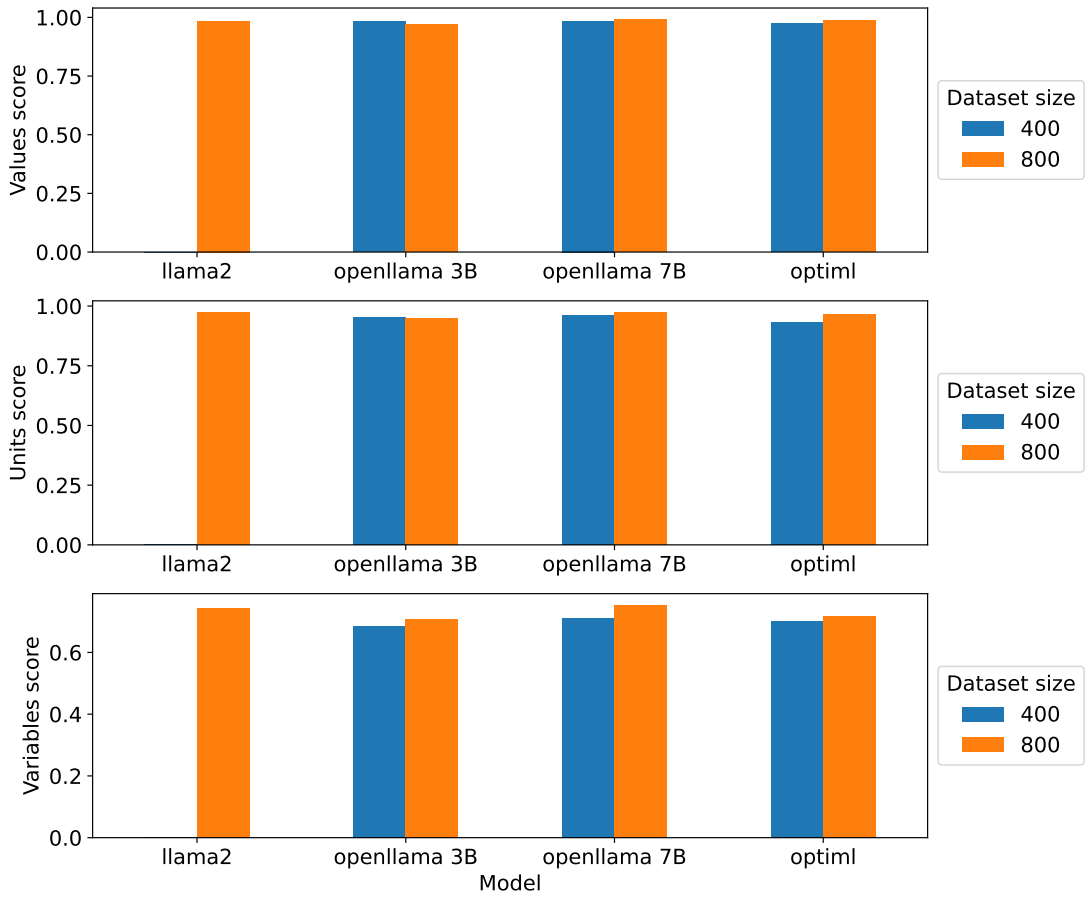

Figure 4.3: Similarity scores for values, units and variables for all models, for dataset size.

a previous example a few lines above. What is, for example, the difference between the number of people, survey participants, and the number of participants? In the end, they all refer to the same concept. However, in some cases, it would be more accurate to use one option or the other. We propose a possible solution for this in the future work section.

In summary, the best model in this evaluation approach is OpenLlama 7B for values (99.04), Llama2 for units of measurement (97.32) and OpenLlama 7B again for variable names (75.27). These numbers showcase how LLMs are extremely accurate for extracting the values and units of measurement. And, despite the vagueness of naming the variables, it still obtains high similarity scores.

We exported (after fine-tuning) the OpenLlama 7B and 3B models twice: once in f16 precision and a second time in 4-bit precision. The aim here is to understand the influence this may have on performance. When we say f16 precision, we mean that at inference time, the merged model contains the original weights in f16 precision plus the

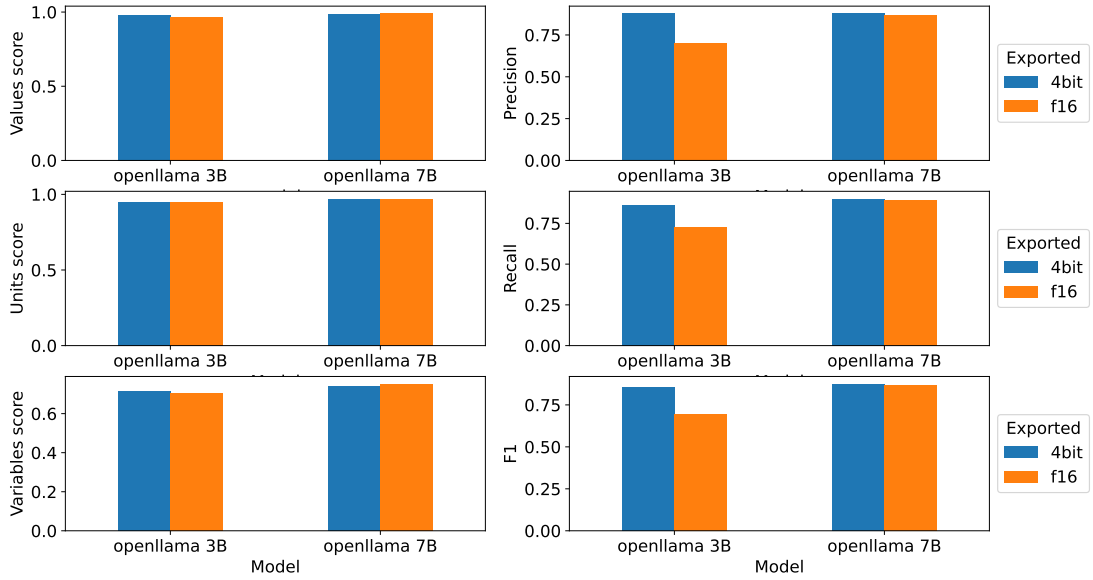

Figure 4.4: Difference in scores and metrics for exporting in f16 or 4bit precision.

adapters in f32 precision. On the other hand, when referring to 4-bit, the merged model weights have a 4-bit precision, and the adapters have an f32 precision, as usual. The results of this evaluation are shown in 4.4. We may see again how similarity scores stay the same across both types. However, we note two facts when assessing the precision, recall and F1 score. First, for the 3B version, the performance increases when using the 4-bit version instead of the f16. Furthermore, for the 7B version, results are also a bit better when using the quantized model. Actually, this OL-7B-4bit obtains the highest recall (90.17) and F1 score (87.56) across all fine-tuned models. The reason for this behaviour is the fact that during training time, the 7B model was trained using QLoRA. This implies that the fine-tuning of the adapters occurs while the original weights are frozen in 4-bit precision (and not in a f16 precision format).

4.1.3 In-context learning comparison

In this subsection, we will briefly discuss the in-context learning performance of our test set. We aim not to study ICL but to compare its results to our fine-tuned models. Results are shown in 4.5. We note there is a positive influence when increasing the number of examples given in the context, as expected. This occurs both for precision and F1 but not for recall. The highest scores for precision (67.20), recall (71.34) and F1 (66.49) are obtained by ten-shot, one-shot and ten-shot, again, respectively. When comparing these scores with the fine-tuned scores, we note there is a huge difference. Specifically, the

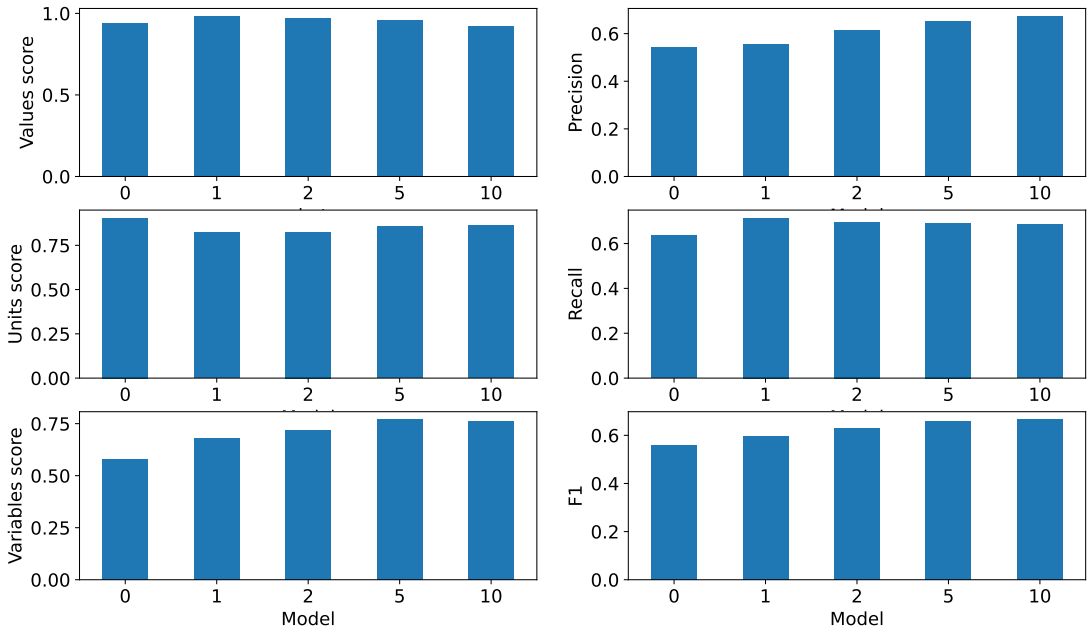

Figure 4.5: Results for different few-shot learning tests, using ChatGPT.

difference for the highest precision (89.98), highest recall (90.17) and highest F1 score (87.56) is 22.78, 18.83 and 21.07, respectively. They represent a huge gap between ICL and fine-tuning and demonstrate the need for parameter-efficient fine-tuning for the task of extracting information from scientific papers. The reason we think this occurs is because of the diversity of the data. We discussed in some sections above that there are many exceptions and special cases in this dataset. Thus, ten examples might not be enough for the model to grasp the patterns inside more than 20 scientific topics. Furthermore, we noted that ChatGPT struggles with hallucinations. The best ChatGPT model generated 173 hallucinations, whereas the best fine-tuned model produced only 37 hallucinations. Most of the time, the LLMs extract non-numeric variables, which is not our task.

4.2 Search engine

4.2.1 Web interface

The web application allows two types of searches. In the first one, the user can look for papers, just as in Google Scholar, and a list of papers will be returned. Each result will have several tags, representing the numeric entities extracted by the model. We may see an example of this view in image 4.6. Additionally, the user can expand on the ”See

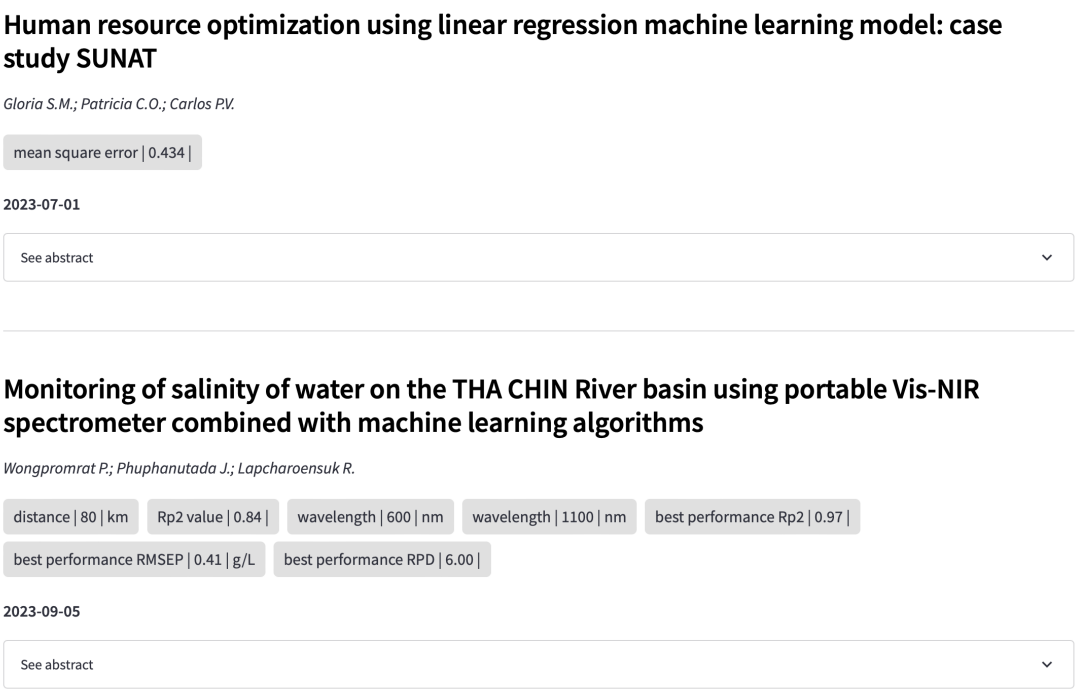

Figure 4.6: First view of the search engine: tags in grey show the numeric variables extracted by the model.

abstract" tab to open it and read the abstract if needed. Furthermore, we tried to redirect to the link of the original paper website by clicking on the title. However, this was only possible in the Science Direct papers, which are only around 6.000 (the Arvix dataset didn't include a URL to the repository of the paper).

There is a second view where a user can look for specific variables such as "tensile strength" or "F1 score", instead of looking by papers. There are several fields the user can search in. He or she can exclude words, or include them, or request for an exact match, for example. In the figure 4.7, we see the result of a query for this second view. It showcases the difference between common scientific databases, where you would probably have to read tons of papers to get to a table similar to that one. The user can filter that table by ordering based on columns, and also can see some statistics about the returned results. The search engine is available at this link for full exploration: https://www.open-science.live/.

4.2.2 Survey analysis

A survey was applied to 18 academics to evaluate the usefulness of the web application. In this subsection, we will discuss some of the results. You may find the raw questions

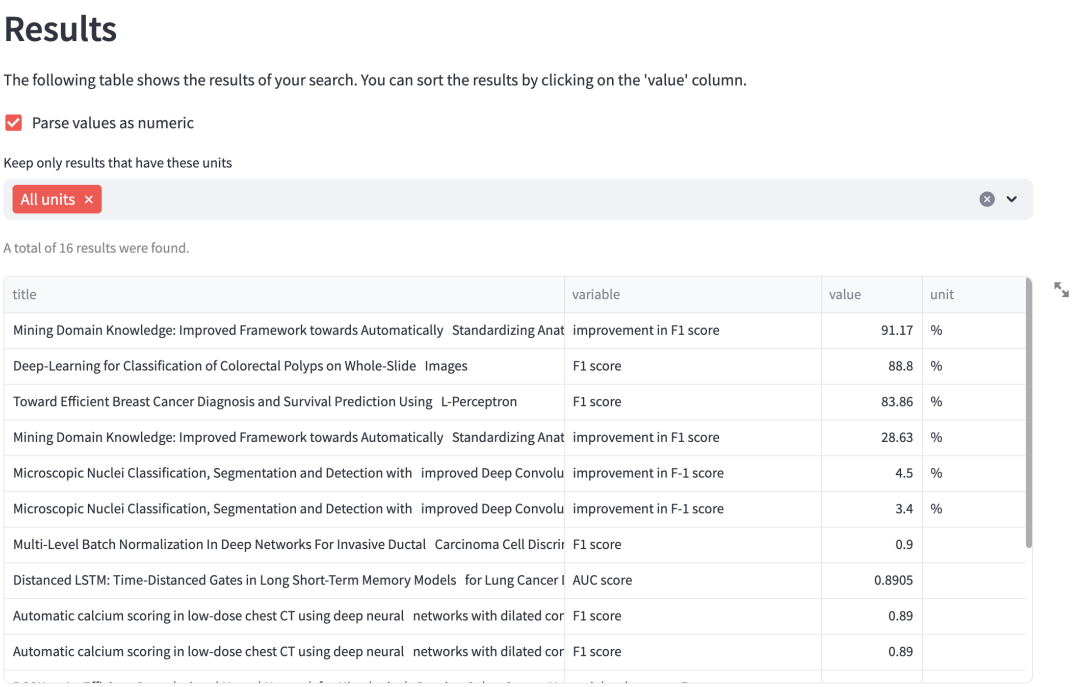

Figure 4.7: Second view of the search engine: the screenshot shows the results for a query where the user looked for the F1 score variable, as well as, filtering by cancer-related papers.

and responses in the attached file. We obtained responses from 13 different areas across STEM and social sciences. This is useful for having a broad amount of opinions. All participants, according to one of the first questions, spend at least 1 hour per week reading scientific papers. 72% of them use Google Scholar.

We asked them to review feature 1 (list of papers with tags) and feature 2 (table with results). For feature 1, 61 % of users found it very or extremely useful. The rest answered from neutral up to slightly useful. Regarding feature 2, 72% of participants considered it to be very or extremely useful. This implies users prefer a table that directly shows the variables instead of having to read through papers. Later on, we asked them if our proposed search engine could save time in their day-to-day research activities. 44% of them answered affirmatively, whereas 33% stated it really depended on the task. Also, 94% of the participants thought this tool is more useful for an academic than for an entrepreneur or for the public government. We do consider this type of tool to be a possibility for institutions outside of academia to participate and consume academic content more easily and straightforwardly.

The last two questions were open. The first one asked them to mention specific

variables they would like to be extracted from the scientific texts. There were plenty of responses. Some of the received answers are: specific performance criteria for different solution approaches (eg., makespan for the scheduling problem using heuristics and exact methods); first material used, methods used, a summary of the results; obtaining information about the variables and the methodological use between them; comparison about specific variables/conclusion of similar work. (More like a table in a review article about a specific domain/variable/conclusion; abstract, conclusions and relevant variables so I can easily find if a paper is related to my research topic; datasets used and tested against; evaluation measures and training times for models; carbon content, mass yield and water-to-biomass ratio) for cellulose, lignin, hemicellulose and different types of lignocellulosic biomass; ion concentrations, annealing temperatures, cost per unit (e.g., comparing different media or methods of energy storage for techno-economic analysis), among all.

In general, participants would like to filter by additional parameters that influence the numeric values. This would lead to include not only numeric but also non-numeric variables, a great opportunity for expanding and creating a more robust search engine. Furthermore, many of them request for a way to easily read a summary of the main sections of the paper, as well as how it connects to similar papers. There are definitely users willing to use this search engine, and there is room for improvement and development. It is a nice niche, given that 94% of respondents considered academic search engines should improve results returned.

Chapter 5

Conclusions

5.1 Final remarks

We discussed the different approaches in the literature to extract information from scientific papers and exposed some arguments on why Large Language Models are ideal for this task. With our own built dataset, we then fine-tuned 9 LLMs of different training samples and parameters. Furthermore, we used few-shot learning to test the performance of ChatGPT at extracting data from scientific corpora. Results show that fine-tuning LLMs leads to obtaining the best metrics compared to the in-context learning approach. Specifically, the OpenLlama 7B model trained and exported in 4-bit precision obtains the highest recall (90.17) and highest F1 score (87.56). This demonstrates the capabilities of LLMs to successfully retrieve the variables, values and units of measurement of scientific papers, even across more than 20 domains. Besides that, when it correctly detects the variables, similarity scores for values, units, and properties are 99.04, 97.32 and 75.27, meaning it correctly detects and extracts the possible variables.

Furthermore, the 7B model is a very good choice for deploying this task to a potential production framework for several reasons: first, it converges way faster than its 3B counterpart, meaning less computational resources will be spent. Second, it can be easily deployed with the vLLM library in a very accessible GPU (such as V100), with very fast response times. Another reason is that relying on ChatGPT, for example, for production-ready systems, is risky, given the model changes occasionally, even if the same prompt is being used.

The majority of the models struggle with hallucinations. One way to handle this is to represent the empty token with a special token (instead of not writing anything). In

that way, it might be easier for the LLM to capture this pattern. Nevertheless, the focus in this task should be on the recall, given we do not want to miss any actual variable in the text. Trash, on the other hand, can be easily filtered using programming.

Finally, there seems to be a niche of users with problems or needs in terms of a better search engine in the academic sector. Users tested our model and our search engine and gave some very interesting ideas on how to improve it and on what next features should be added. We think there is an opportunity to build a great service and product using state-of-the-art AI techniques.

5.2 Future work

As we discussed in the results section, almost all LLMs obtained similar losses during training. However, there are bigger differences when we pass each fine-tuned model through our own metrics. To solve this, a next future approach could be to develop a new loss function that replaces the cross entropy loss.

Furthermore, we mentioned the ambiguity that exists when setting the name of a variable or property. The domain expert may consider one word, but the LLM may choose another, which is not necessarily wrong. The best solution to this problem might be to co-create among different experts a list of possible, fixed entities to extract, given a scientific topic. With this list, the LLM would now try to match the entities it finds with the list of fixed variables it was given. In this way, we prevent this problem from occurring while still leveraging the power of LLM for information extraction.

Bibliography

[1] arXiv.org submitters. arxiv dataset, 2023.

[2] Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, (2):179–190, 1983.

[3] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

[4] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems*, 13, 2000.

[5] Nadezhda Biziukova, Olga Tarasova, Sergey Ivanov, and Vladimir Poroikov. Automated extraction of information from texts of scientific publications: insights into hiv treatment strategies. *Frontiers in genetics*, 11:618862, 2020.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Wim De Mulder, Steven Bethard, and Marie-Francine Moens. A survey on the application of recurrent neural networks to statistical language modeling. *Computer Speech Language*, 30(1):61–98, 2015.

[8] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019.

[11] Alexander Dunn, John Dagdelen, Nicholas Walker, Sanghoon Lee, Andrew S Rosen, Gerbrand Ceder, Kristin Persson, and Anubhav Jain. Structured information extraction from complex scientific text with fine-tuned large language models. *arXiv preprint arXiv:2212.05238*, 2022.

[12] Mehmet Firat. What chatgpt means for universities: Perceptions of scholars and students. *Journal of Applied Learning and Teaching*, 6(1), 2023.

[13] Xinyang Geng and Hao Liu. Openllama: An open reproduction of llama, May 2023.

[14] Satanu Ghosh and Kun Lu. Band gap information extraction from materials science literature–a pilot study. *Aslib Journal of Information Management*, 75(3):438–454, 2023.

[15] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[16] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.

[17] Srinivasan Iyer, Xi Victoria Lin, Ramakanth Pasunuru, Todor Mihaylov, Daniel Simig, Ping Yu, Kurt Shuster, Tianlu Wang, Qing Liu, Punit Singh Koura, Xian Li, Brian O'Horo, Gabriel Pereyra, Jeff Wang, Christopher Dewan, Asli Celikyilmaz, Luke Zettlemoyer, and Ves Stoyanov. Opt-iml: Scaling language model instruction meta learning through the lens of generalization, 2023.

[18] Takeshi Kikuchi, Masuhiro Nishimura, Chikage Shirakawa, Yasutaka Fujita, and Takeshige Otoi. Relationship between oxygen partial pressure and inhibition of cell aggregation of human adipose tissue-derived mesenchymal stem cells stored in cell preservation solutions. *Regenerative Therapy*, 24:25, 12 2023.

[19] Yanis Labrak, Mickael Rouvier, and Richard Dufour. A zero-shot and few-shot study of instruction-finetuned large language models applied to clinical and biomedical tasks. *arXiv preprint arXiv:2307.12114*, 2023.

[20] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[21] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[22] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[23] Shiyuan Liu, Junchen Liao, Xin Huang, Zhuomin Zhang, Weijun Wang, Xuyang Wang, Yao Shan, Pengyu Li, Ying Hong, Zehua Peng, et al. Green fabrication of freestanding piezoceramic films for energy harvesting and virus detection. *Nano-Micro Letters*, 15(1):131, 2023.

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[26] Vatsala Nundloll, Robert Smail, Carly Stevens, and Gordon Blair. Automating the extraction of information from a historical text and building a linked data model for the domain of ecology and conservation science. *Heliyon*, 2022.

[27] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[28] Maciej P Polak, Shrey Modi, Anna Latosinska, Jinming Zhang, Ching-Wen Wang, Shanonan Wang, Ayan Deep Hazra, and Dane Morgan. Flexible, model-agnostic method for materials data extraction from text using general purpose language models. *arXiv preprint arXiv:2302.04914*, 2023.

[29] Xiang Qin, Jingjing Zheng, Xiaojun Yang, Wensheng Gong, Liping Luo, and Lijun Ji. Bioactivity of a gelatin/organic-inorganic hybrid biomaterial fibre enhanced by metronidazole release. *Materials Letters*, 325:132803, 2022.

[30] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[31] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization, 2022.

[32] Yiqiu Shen, Laura Heacock, Jonathan Elias, Keith D Hentel, Beatriu Reig, George Shih, and Linda Moy. Chatgpt and other large language models are double-edged swords, 2023.

[33] Matthew C Swain and Jacqueline M Cole. Chemdataextractor: a toolkit for automated extraction of chemical information from the scientific literature. *Journal of chemical information and modeling*, 56(10):1894–1904, 2016.

[34] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[35] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal

Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[36] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[38] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[39] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[40] Michiko Yoshitake, Fumitaka Sato, Hiroyuki Kawano, and Hiroshi Teraoka. Materialbert for natural language processing of materials science texts. *Science and Technology of Advanced Materials: Methods*, 2(1):372–380, 2022.

[41] E. Yurtsev. Kor. `https://github.com/eyurtsev/kor`, 2023.

[42] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models, 2022.

[43] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models, 2023.

Appendix A

First appendix

A.1 First section

Category	Original (1000)	Train (800)	Train (400)	Test (200)
Nuclear Energy and Engineering	57	44	28	13
Biochemistry	56	46	20	10
Fuel Technology	56	50	27	6
Renewable Energy, Sustainability	56	44	16	12
Bioengineering	52	36	20	16
General Energy	49	42	23	7
Metals and Alloys	49	42	17	7
Biomaterials	48	38	16	10
Artificial Intelligence	47	43	23	4
Water Science and Technology	47	33	15	14
Organic Chemistry	43	35	13	8
Surfaces, Coatings and Films	43	34	18	9
General Chemical Engineering	42	28	13	14
Energy Engineering and Power Technology	40	36	23	4
General Engineering	40	30	17	10
Inorganic Chemistry	38	34	21	4
Mechanics of Materials	38	26	15	12
Pollution	34	29	13	5
Management Science and Operations Research	29	27	10	2
Ceramics and Composites	27	24	13	3
Control and Optimization	24	17	10	7
Materials Science (miscellaneous)	23	17	6	6
Computer Vision and Pattern Recognition	22	17	9	5
Materials Chemistry	21	12	7	9
Electronic, Optical and Magnetic Materials	14	11	3	3
Waste Management and Disposal	5	5	4	0

Table A.1: Number of sentences for each topic, in each one of the dataset splits.

	llama2_7B_ 4bit_800_f16	**openllama_7B_ 4bit_800_f16**	**openllama_7B_ 4bit_400_f16**
batch_size	128	128	128
per_device_train_batch_size	4	4	4
gradient_accumulation_steps	32	32	32
num_epochs	50	50	50
learning_rate	0.00024	0.0003	0.0003
cutoff_len	512	512	512
lora_r	16	16	16
lora_alpha	16	16	16
lora_dropout	0.05	0.05	0.05
lora_target_modules	['q_proj', 'v_proj']	['q_proj', 'v_proj']	['q_proj', 'v_proj']
warmup_steps	100	100	100
optimizer	paged_adamw_8bit	paged_adamw_8bit	paged_adamw_8bit

Table A.2: List of hyperparameters used for each fine-tuned model. (Table 1 of 2).

	openllama_3B_ 8bit_800_f16	**openllama_3B_ 8bit_400_f16**	**optiml_1-3B_ 32bit_800_f32**	**optiml_1-3B_ 32bit_400_f32**
batch_size	128	128	16	16
per_device_train_batch_size	4	4	4	4
gradient_accumulation_steps	32	32	4	4
num_epochs	275	315	400*	400*
learning_rate	0.0003	0.0003	0.0002	0.0002
cutoff_len	512	512	512	512
lora_r	16	16	16	16
lora_alpha	16	16	32	32
lora_dropout	0.05	0.05	0.05	0.05
lora_target_modules	['q_proj', 'v_proj']	['q_proj', 'v_proj']	['q_proj', 'v_proj']	['q_proj', 'v_proj']
warmup_steps	100	100	100	100
optimizer	paged_ adamw_8bit	paged_ adamw_8bit	paged_ adamw_8bit	paged_ adamw_8bit

Table A.3: List of hyperparameters used for each fine-tuned model. (Table 2 of 2). *For the OPT models, the values in `num_epochs` actually refers to the number of steps.

Appendix B

Participants' information sheet

Participants were given the official Participant Information Sheet. This contained the names of the researchers, the purpose of the study, the reason we asked them to participate in the study, that it was not mandatory to participate in the survey; what would happen if the person decided to take part as well as with the data collection and protection; risks or benefits associated with participating; their data protection rights; who they can contact; and a consent form. Full Participants' information sheet can be found here.

Appendix C

Participants' consent form

Consent was gathered through an online form, given the survey was done online. Participants, before starting, were asked to read, understand and agree on the consent. The following was the included text: I agree to participate in this survey and for the information I provide to be shared with the researcher. I understand my participation is voluntary, and I am free to choose not to participate and or withdraw at any time. I understand that my personal information will, if collected, be anonymised and will be secured for future research. I consent to my anonymised data being used in academic publications and presentations. I allow my data to be used in future ethically approved research. I have read and understood the Participant Information Sheet.