Universiteit Leiden

Master Computer Science

LLMs for Biomedical NER: Comparing Zero-Shot, Few-Shot, and Instruction Tuning

Name:	Xue Jun Wang
Student ID:	s2367378
Date:	[20/8/2025]
Specialisation:	Master Computer Science: Data Science
1st supervisor:	Prof. Dr. Suzan Verberne
2nd supervisor:	Dr. Lifeng Han

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

In recent years, large language models (LLMs) have made major developments in natural language processing (NLP). Their ability to understand and generate human-like text has led to strong performance across a wide range of tasks. One of these tasks is Named Entity Recognition (NER), which involves identifying and classifying entities such as diseases, chemicals, or genes in text.

This study investigates the ability of LLMs to extract long and complex biomedical entities, including rare or emerging terms, using three approaches: zero-shot, few-shot and instruction tuning. The zero-shot setting provides the LLMs with only an instruction, while the few-shot setting includes two examples to guide the model, we provide the LLMs with two example, because of the limit availability of examples. Results show that few-shot learning outperforms zero-shot learning on the NCBI-disease dataset, while the reverse is observed on the BC5CDR-chemical and BC2GM-gene datasets. However, across all three datasets, the instruction tuned LLaMA3-8B and PMC-Llama-13B achieve the highest performance, often nearly doubling the scores of zero-shot and few-shot learning, indicating the advantage of domain-specific adaptation.

Given these findings, we use instruction tuning to assess LLM performance in extracting long and complex biomedical entities. We evaluate two instruction-tuned models on an in-domain biomedical dataset and an out-of-domain dataset. The models are instruction tuned on the in-domain training set and evaluated on both the in-domain test set and the out-of-domain test set, where the out-of-domain dataset consists of long and complex entities. Results show substantially better performance on the in-of-domain test set, with performance dropping on the out-of-domain test set, this highlights the models' difficulty in generalizing to unfamiliar or complex biomedical terms.

To further test generalization, we conducted an additional experiment in which we removed all entities with the sentence from the out-of-domain test set that also appeared in the LLMs' training data. This ensured that the remaining entities were completely unseen. Performance dropped even further on this filtered test set, highlighting the difficulty LLMs face when encountering entirely novel biomedical entities. These findings highlight both the strenghts and limitations of LLMs for biomedical NER, especially when dealing with rare, complex and unseen entities.

Contents

1 Introduction

Automatic extraction of medical entities is a crucial task in the field of biomedical informatics, where the goal is to identify and categorize key information from vast amounts of medical texts. This process, known as medical entity recognition or NER, involves pinpointing various types of entities within the text, such as diseases, symptoms, medications, and anatomical terms. These entities vary in complexity, ranging from relatively common terms such as diabetes, thrombosis, Parkinson, or inflammation, to more complex ones that are longer and consist of multiple words, also known as nested or multi-word entities, for example *ovarian or fallopian tube cancers* or *hand foot syndrome*. This also includes novel or emerging entities, meaning previously unseen or entities that are new to the model. To achieve this, models are trained on large-scale biomedical texts, such as PubMed [4] articles and clinical notes, to develop a comprehensive understanding of the specialized language used in the biomedical field [25]. Following this training, these models are fine-tuned for specific tasks, such as NER, to extract entities like genes, diseases, and chemicals from biomedical texts. This fine-tuning process ensures that the models can accurately and reliably perform entity recognition in various biomedical applications.

LLMs have achieved promising results across various natural language processing (NLP) tasks, including machine translation [47], question answering [37], and named entity extraction [13, 22, 48]. In recent years, LLMs have shown great potential in performing NER [48, 22]. By using just a few task-specific examples as demonstrations, these models can generate accurate predictions for new test inputs. Their ability to generalize from limited examples makes them particularly useful for tasks where labeled data is scarce, highlighting their potential for real-world applications in domains such as biomedical text mining and legal document analysis.

This thesis investigates that potential of LLMs for NER, specifically assessing their ability to extract complex and long biomedical entities from text. Medical texts often contain highly specialized terminology, making entity extraction a challenging task. Fine-tuning LLMs for NER typically relies on large amounts of labeled data for training, recent studies have shown that LLMs can achieve competitive performance with just a few task-specific examples through few-shot learning. This raises several important questions: Is few-shot learning with LLMs sufficient to match or even outperform fine-tuned LLMs in domain-specific NER? And to what extent can LLMs accurately identify long, complex, and emerging entities?

To explore this, this study aims to evaluate the effectiveness of LLMs in handling the challenges of medical entity extraction, particularly in scenarios where entities are long, ambiguous, or highly specialized. The following research questions guide this investigation:

1. How does the performance of zero-shot learning compare to few-shot learning, and how do both approaches compare to fine-tuned LLMs for medical NER?

2. Can the results reported by Keloth et al.[22] on using LLMs for NER be reproduced by following their proposed methodology and using the same LLMs?

3. To what extent are LLMs capable of accurately extracting long and complex biomedical entities, including rare or emerging terms?

By addressing these questions, this study seeks to provide insights into the strengths and limitations of LLMs for medical NER, particularly in scenarios where labeled data is scarce or expensive to obtain.

The remainder of the paper is organized as follows: In Section 2, we discuss related methodological work. In Sections 3, we discuss the data sets we use, followed by a detailed description of our methodology. In Section 4, we present our results, which are discussed further in Section 5. Finally, we address our research questions and outline potential improvements for future work in Section 6.

2 Related Work

2.1 Named Entity Reginition

NER is a natural language processing task that identifies and classifies named entities in unstructured text data. Named entities refer to real-world entities present in the text [8, 25]. The NER process can typically be divided in the following steps [32]: The process begins with tokenization, where the text is segmented into individual tokens or words. After tokenization, the next step is feature extraction. In this step, the features are extracted for each token, where each token or the token is represented as a dense vector embedding that can help in identifying the entities [44, 14]. The standard approach in sequence labelling to label each token is using the Inside-Outside-Beginning (IOB) tagging scheme. In this scheme, 'B-' indicates the beginning of an entity, 'I-' denotes that the token is inside an entity, and 'O' signifies that the token is outside any entity [17]. This structured approach allows NER systems to accurately extract and classify entities from unstructured text [17]. After this, the named entities identified in the text are categorized into predefined groups such as persons, organizations, locations, dates, and other relevant categories. This classification enables the structured extraction of meaningful information from the text.

Besides the standard NER, there is an advanced form known as fine-grained NER [21], which goes deeper into categories by identifying specific subtypes within broader groups, such as actors, athletes, or musicians within the "PERSON" category. Additionally, there are domain-specific NER applications [16], such as in biomedicine, where it detects entities like proteins and genes. NER faces several challenges [30]. Ambiguity arises when words or phrases can refer to different types of entities depending on the context. Variability presents another challenge, as there are often different ways to refer to the same entity. Additionally, domain-specific vocabulary can complicate NER, since specialized terms in fields such as medicine or law require domain-specific training data for accurate recognition and classification.

2.2 Large Language Models and In-context Learning

LLMs have achieved substantial performance improvements across a wide range of natural language processing tasks [11, 43]. Methods for applying LLMs to downstream tasks can be divided into two main categories: fine-tuning and in-context learning. The fine-tuning strategy involves taking a pre-trained LLM and adapting it to a specific downstream task by training it further on labeled task-specific data. This process typically runs additional training epochs on the supervised dataset, allowing the model to refine its parameters and improve performance on the target task [35, 36].

Different from the fine-tuning strategy, in-context learning (ICL) uses the inherent capabilities of LLMs by providing them with a few task-specific examples, or demonstrations, as part of the input prompt. Instead of updating the model's pa-

rameters, ICL guides the LLM to generate relevant outputs based on these demonstrations. This approach was introduced by Radford et al. [34], who introduced the concept of reformulating downstream tasks using prompts that include examples, enabling the model to recognize patterns and generate accurate predictions without additional training. Brown et al [11] conducted a study on the capabilities of in-context learning, analyzing its effectiveness across a variety of tasks using their GPT-3 model. Through a series of experiments, they demonstrated how GPT-3 could perform tasks with minimal task-specific examples, showcasing the model's ability to generalize and adapt to new challenges without requiring fine-tuning. In the papers of Perez et al. [33], Lu et al. [27], and Rubin et al. [39] have demonstrated that the quality of prompts and demonstrations plays a critical role in the performance of in-context learning. Their research shows that carefully designed prompts and well-chosen examples can substantially enhance the performance of the LLMs, enabling models to better understand and execute tasks without the need for extensive fine-tuning. These findings highlight the importance of prompt engineering in maximizing the potential of in-context learning.

2.3 Named Entity Recognition with LLMs

One of the tasks that LLMs can learn to perform with ICL is NER, from instructions and a few examples provided directly in the prompt, without needing explicit fine-tuning on the task. This capability allows large generative LLMs to be versatile and adaptable for a wide range of natural language processing tasks, including NER, by understanding the context and semantics of the text based on their pretraining [11].

Before the introduction of LLMs for NER, NER was performed using BERT or other transformer-based models that were finetuned on labelled data, which set a high standard for domain-specific biomedical NER tasks. To fine-tune a BERT model, it requires a substantial amount of labeled data for training. In contrast, instruction-tuned large language models can perform various tasks effectively with only a few examples and clear instructions, without the need for extensive labeled datasets.

Task-specific prompts have been shown to improve the performance of LLMs in both zero-shot and few-shot learning scenarios [23]. Keloth et al. [22] explore this by experimenting with instruction-tuned LLaMA models for biomedical entity recognition in their paper. The results indicate no significant improvement with enhanced prompts. Also with additional information added to the prompt, that describes the entity to be extracted, there was no noticeable effect on the performance of the instruction tuned model. This suggests that, for instruction-tuned models, the inclusion of detailed task-specific information in prompts may not substantially affect performance. In contrast, research by Shuhe Wang et al. [48] found that adding additional information to prompts positively impacts the performance of large language models, particularly in low-resource settings with only 8 or 100 training sentences. In these cases, supervised models, including BERT variants, perform substantially worse than GPT-3. For example, with just 8 training

examples, GPT-NER achieves an F1 score of around 60, while supervised models perform at nearly 0, demonstrating GPT-NER's generalization in low-resource scenarios. The study further shows that as the training set grows to 10% of the total dataset, supervised models improve substantially, while GPT-3 shows only marginal improvement. This indicates that for in-context learning, increasing the quantity of training data has limited benefits compared to enhancing the quality of prompt examples. Improvements such as switching from random to kNN-based demonstration retrieval and refining prompt structure. For instance, by incorporating self-verification are more effective strategies for boosting performance.

As mentioned before, LLMs are increasingly improving in performing specific tasks by adjusting the number of parameters of a pre-trained model through supervised training on a dataset specific to the desired task. However, in more general settings, domain-specific supervised models such as BERT variants still tend to outperform LLMs. In contrast, in low-resource scenarios, as shown by GPT-NER, LLMs can generalize surprisingly well even with very few examples. This performance can be further improved when prompts are carefully designed with specific instructions, entity definitions, or task-specific annotation guidelines, this demonstrates that prompt engineering plays a critical role in optimizing LLM output [19].

2.4 Biomedical Named Entity Reginition

The approach to Biomedical NER has advanced significantly over the years, moving from rule-based methods to feature-driven statistical models, and now to end-to-end neural networks. These neural networks have greatly reduced the need for human involvement in tasks like creating rules or selecting features by using self-supervised pretraining to automatically learn from data. Early systems depended on rule-based methods that used dictionaries and regular expressions to identify entity mentions, with one of the main challenges being the extensive requirement for domain-specific knowledge engineering [52, 46]. The statistical machine learning systems introduced at that time were Hidden Markov Model [31] and Conditional Random Field (CRF) [45]. An example of a system that used CRF is ABNER (A Biomedical Named Entity Recognizer) [40], where CRFs are used to enhance the performance of the rule-based systems. The development of deep neural networks led to new methods like BiLSTM-CRF [20], which integrate LSTM networks with CRF layers, thereby eliminating the need for extensive feature engineering.

From BERT models [14], domain-specific pretraining emerged, using massive biomedical corpora like BioBERT [24] and SciBERT [9]. These models continue to improve through pretraining on larger, more relevant datasets, advancements in neural network architectures, and fine-tuning on specific downstream tasks. As a result, their performance reaches state-of-the-art levels across various tasks and datasets. Ongoing research is further refining these models by using massive pretraining and multi-task learning (MTL), allowing them to use diverse biomedical sources and tasks simultaneously [49, 53]. MTL allows the model to train on multiple related tasks simultaneously. This joint training allows the model to learn features or representations that are useful across all the tasks. These shared features help the model

generalize better and perform well on each individual task because the knowledge learned from one task can inform and improve the learning process of other tasks [38]. In a recent study by Luo et al. [28], a model was trained to recognize multiple entity types. This all-in-one NER approach combines various heterogeneous biomedical entity datasets into a single format.

New approaches are being explored for biomedical NER, such as using large language models to identify biomedical entities in text or entire corpora. In a recent research of Keloth et al. [22] the performance of LLMs and BERT models have been observed to identify biomedical entities, where LLMs depend on prompt engineering. They found that GPT models underperformed compared to PubMedBERT and BioNER-LLaMA when evaluated on three NER datasets using a uniform prompt. This is because of the use of pretrained baselines, namely PubMedBERT and BioNER-LLaMA. PubMedBERT is pretrained from scratch using abstracts and full-text articles from PubMed and PubMed Central (PMC). PMC-LLaMA [51] is an open-source language model designed specifically for medical applications, trained with a data-centric approach that incorporates 4.8 million biomedical papers and 30,000 medical textbooks. The use of pretrained baselines allowing PubMedBERT and BioNER-LLaMA to better understand domain-specific medical language and terminology than the GPT models, making them more effective for the named entity recognition task. This was also observed in the experiment, BioNER-LLaMA outperformed PubMedBERT on two out of three datasets, although these results were not statistically significant, because the BioNER-LLaMA achieved a 2% improvement in the F1 score compared to PubMedBERT. However, for the chemical entity extraction task, PubMedBERT achieved a better performance, with an F1 score that was 0.04 higher than BioNER-LLaMA.

3 Methods

Instruction tuning, also referred to as instruction finetuning, is a process in natural language processing where a pre-trained model is further trained on a set of task-specific instructions together with labeled data to improve its performance on particular tasks. The concept of instruction tuning is introduced by Wei et al.[50] as a method to improve zero-shot performance on unseen tasks by fine-tuning language models on a collection of tasks described using natural language instruction templates. This concept of improving zero-shot performance through instruction tuning has been further substantiated by implementations such as Stanford Alpaca [6] and Flan-UL2 [7].

3.1 Datasets for instruction tuning

For zero-shot, few-shot and instruction tuning with LLMs, we use three different datasets, these datasets are also used in the paper of Keloth et al.[22].

The NCBI disease corpus [15] consists of 793 PubMed abstracts manually annotated with disease mentions and their corresponding MeSH or OMIM concept identifiers. The dataset is divided into training, development, and test sets. It contains a total of 6,892 mentions of diseases, which are linked to 790 unique identifiers.

The BC5CDR-Chemical [26] corpus consists of 1,500 PubMed abstracts split into subsets of 500 each for training, development and test. The corpus is annotated for disease mentions, chemical mentions, and chemical-disease interactions. For zero-shot, few-shot and instruction tuning we only use the chemicals dataset. Each entity annotation contains both the mention text spans and normalized concept identifiers, using MeSH as the controlled vocabulary.

The BC2GM (BioCreative II Gene Mention) [42] corpus focuses on extracting gene and gene product mentions from MEDLINE sentences. The corpus is divided into 12500 train sentences, 2500 development sentences, and 5000 test splits sentences.

The following two datasets are used in the experiment to analyze the performance of the LLM on complex and long medical entities.

The BC5CDR-Disease corpus [1] consists of 1,500 articles annotated with 5818 diseases mentions. The BC5CDR dataset consist of 4797 sentences, where each sentence has zero or more biomedical entities.

The NLMChem corpus [3] consists of 150 full-text articles from the PubMed Central Open Access dataset, with around 5000 unique chemical name annotations, mapped to around 2000 MeSH identifiers. comprising 67 different chemical journals, aiming to cover a general distribution of usage of chemical names in the biomedical literature.

3.2 Models

LLaMA3-8B: The LLaMA3-8B model is an 8 billion parameter large language model developed by Meta AI [2], designed to understand and generate human-like text across various natural language processing tasks, with capabilities enhanced through instruction tuning and trained on diverse internet text.

PMC-Llama-13B: The PMC-LLaMA [51] is an open-source language model developed specifically for medical applications, with 13 billion parameters. It was trained using a large dataset of 4.8 million biomedical research papers and 30,000 medical textbooks. Additionally, it underwent instruction tuning with samples focused on various medical tasks, such as medical conversations, reasoning in question-answering scenarios, and question-answer pairs generated from knowledge graphs.

3.3 Prompt Development

3.3.1 Zero-shot setting

The three NER datasets mentioned in subsection 3.1 were used to evaluate the performance of LLMs for entity recognition. Specifically, this section focuses on the zero-shot setting with the LLaMA3-8B model. However, the datasets were not immediately suitable for the zero-shot experiment and required further organization before they could be used, so each dataset were first split into individual sentences, ensuring that each line contained a single sentence. Since the entities in the datasets were already marked, those markings were removed to create an unlabeled version of the data. After the preparation of the datasets, the next step was defining the prompt construction, an example of a prompt is illustrated in Figure 1. The prompt construction is decomposed into two components:

1. *Instruction*: This describes the task that the model should perform, in this case, extracting the biomedical entity from a sentence.

2. *Input sentence*: This is a sentence from the dataset that can come from any of the three datasets and may contain zero, one, or more entity mentions.

Instruction

Given a sentence extract disease entities from it by highlighting them with <mark> and </mark>. If not, output the same sentence

Input

Clustering of missense mutations in the ataxia - telangiectasia gene in a sporadic T - cell leukaemia.

Output

1: The input sentence is: Clustering of missense mutations in the ataxia - telangiectasia gene in a sporadic T - cell leukaemia.

I found disease entities in this sentence:

* ataxia
* telangiectasia
* T-cell leukaemia

Here is the modified sentence with highlighted disease entities:

Clustering of missense mutations in the <mark>ataxia</mark> - <mark>telangiectasia</mark> gene in a sporadic T - cell <mark>leukaemia</mark>.

Note: I used the term "disease entity" to refer to specific diseases or conditions mentioned in the sentence. In this case, "ataxia", "telangiectasia", and "T-cell leukaemia" are all disease entities.

Figure 1: An prompt example that consist of an instruction that describes a task, paired with an input that provides further context. The output is a response that appropriately completes the request.

In the instruction section, the task for the LLM is described. A similar instruction, replacing the word "disease" with "chemical" or "gene", was used when transforming the dataset from NCBI-disease to BC5CDR-Chemical or BC2GM-gene. Along with the instruction, an input sentence is also provided for the Llama3-8b model to process. From the instruction and input, an output is generated that completes the request. The output consists of the sentence, with entities highlighted using <mark> and </mark> tags if entities are present. If no entities are identified, the output will be the exact input sentence without any modifications. In addition to the input sentence with the <mark> and </mark> tags, the LLM also generates additional information related to the input sentence, this is illustrated in Figure 1.

Due to the excessive information generated by the Llama3-8B model, it becomes difficult to filter out the input sentence with the <mark> and </mark> tags, which is needed for calculating Precision, Recall, and F1-score later. Online discussions, on platforms as Stack Overflow [41, 5], reveal that the issue of excessive output in LLaMA3 is a common problem encountered by many users on the platform. The only solution is to experiment with different approaches to find an effective workaround.

Instruction

Given a sentence that start with <s> and end with </s> extract disease entities from it by highlighting them with <mark> and </mark>. If not, output the same sentence. Start the modified sentence with: [output] and skip the extra information, only output the sentence.

Input

<s> Clustering of missense mutations in the ataxia - telangiectasia gene in a sporadic T - cell leukaemia . </s>

Output

1: The input sentence is: Clustering of missense mutations in the ataxia - telangiectasia gene in a sporadic T - cell leukaemia.

I found disease entities in this sentence:

* ataxia
* telangiectasia
* T-cell leukaemia

Here is the modified sentence with highlighted disease entities:

<s>Clustering of missense mutations in the <mark>ataxia</mark> - <mark>telangiectasia</mark> gene in a sporadic T - cell <mark>leukaemia</mark>.</s>

Note: I used the term "disease entity" to refer to specific diseases or conditions mentioned in the sentence. In this case, "ataxia", "telangiectasia", and "T-cell leukaemia" are all disease entities.

Figure 2: An prompt example that consist of an instruction and an input sentence that has a error in the output

Instruction

Given a sentence that start with <s> and end with </s> extract disease entities from it by highlighting them with <mark> and </mark>. If not, output the same sentence. Start the modified sentence with: [output] and skip the extra information, only output the sentence.

Input

<s> These clustered in the region corresponding to the kinase domain , which is highly conserved in ATM - related proteins in mouse , yeast and Drosophila . </s>

Output

6: The sentence is: <s>These clustered in the region corresponding to the kinase domain, which is highly conserved in ATM-related proteins in mouse, yeast and Drosophila.</s>

The disease entities are highlighted as follows:

<s> <mark>ATM</mark> </s>

Figure 3: An prompt example that consist of an instruction and an input sentence that has a error in the output.

Figure 2 and 3 illustrate two different instructions along with their corresponding outputs. As illustrated in Figure 1, we can see that an instruction that is too short or not specific enough leads to output with excessive information. The excessive information describes the steps Llama3-8b model takes to arrive at its final answer. On the other hand, an instruction that is too long or specific sometimes results in output where LLaMA3-8B forgets to perform part of the instruction, as illustrated in Figure 2 and 3. In conclusion, we can see that there is no specific method to control the output of Llama3-8B model. The only way to achieve the desired output is through trial and error.

Instruction

Given a sentence that start with <s> and end with </s> extract disease entities from it by highlighting them with <mark> and </mark>. If not, output the same sentence. Start the answer with: [output].

Input

<s> Clustering of missense mutations in the ataxia - telangiectasia gene in a sporadic T - cell leukaemia . </s>

Output

1: [output] <s>Clustering of missense mutations in the <mark>ataxia-telangiectasia</mark> gene in a sporadic T-cell <mark>leukaemia</mark>. </s>

Note: I extracted the disease entities "ataxia-telangiectasia" and "leukaemia" and highlighted them with <mark> tags.

Figure 4: The prompt example with the instruction that generated the least amount of excessive information.

After several trials, the final instruction that proved effective is illustrated in Figure 4. This instruction minimized the generation of excessive information. In the instruction, we added extra markup patterns to simplify the post-processing. The <s> and </s> tags are used to easily locate specific sentences within the output during post-processing. The [output] tag serves as a backup in case the <s> and </s> markers are not generated by the LLaMA3-8B model. These added markup patterns help us to filter out the input sentence from the output modified by the LLaMA3-8B model, which can then be used for the evaluation phase.

Instruction

Given a sentence that start with <s> and end with </s> extract gene entities from it by highlighting them with <mark> and </mark>. If not, output the same sentence. Start the modified sentence with: [output] and skip the extra information, only output the sentence.

Input

<s> Furthermore , the cognate binding protein is present in both rat and human (HeLa) cell nuclear extracts . </s>

Output

365: [output]

Further<s><mark>cognate binding protein</mark></s>, the <mark>cognate binding protein</mark> is present in both rat and human (<mark>HeLa</mark>) cell nuclear extracts .

Note: I extracted the gene entity "cognate binding protein" as it appears twice in the sentence, and also extracted "HeLa" which is a gene name.

Figure 5: The prompt example with the output, where the tags are placed incorrectly

In the post-processing step, we prepare the output generated by the LLaMA3-8B model for the evaluation phase. First, we automatically filter the input sentence from the output using the markup mentioned in the section above. After filtering out the sentences containing entities from the output, a manual check is performed to ensure each sentence has been correctly filtered. This is necessary because of formatting errors, the LLaMA3-8B model fails to include the required markups or places them incorrectly as illustrated in Figure 5, making it impossible to filter the sentences using markup alone. Next, we extract the entities from each sentence automatically by matching the <mark> and </mark> tags, if the sentence contains any entities. As a result, each line will contain the entities from the corresponding sentence, and if the sentence contains no entities, the line will remain empty. In this step, a manual check is also performed to ensure that each entity has been correctly extracted.

3.3.2 Few-shot setting

Few-shot demonstration

Here are 2 examples about extracting disease entities in sentences:

Example 1:
Identification of APC2 , a homologue of the <mark>adenomatous polyposis coli tumour</mark> suppressor .

Example 2:
The <mark>adenomatous polyposis coli (APC) tumour</mark> - suppressor protein controls the Wnt signalling pathway by forming a complex with glycogen synthase kinase 3beta (GSK - 3beta) , axin / conductin and betacatenin .

Instruction

Extract disease entities from the next sentence by highlighting them with <mark> and </mark>. If not, output the same sentence. Start the answer with: [output].

Input

Clustering of missense mutations in the ataxia - telangiectasia gene in a sporadic T - cell leukaemia .

Output

1: [output] Clustering of missense mutations in the <mark>ataxia-telangiectasia</mark> gene in a sporadic <mark>T-cell leukaemia</mark>.

Figure 6: An few-shot prompt example that consist of two example, where the biomedic entities are already marked, allowing the LLaMA3 model to learn from them. An instruction that describes a task, paired with an input that provides further context. The output is a response that appropriately completes the request.

For the few-shot approach, we still use the LLaMA3-8B model as the LLM, and the same datasets as in the zero-shot approach. For this setting the prompt structure was adjusted as shown in Figure 6. Before giving the instruction to the LLaMA3-8b model, two examples from the training set are first provided. Prior to using the training set, all sentences that do not contain biomedical entities are removed, as those sentences are not useful examples for the LLaMA3-8B model to learn from. Using these examples also regulates the format of the LLM outputs for each test input, as LLM will likely generate outputs that mimic the format of the demonstrations. The two examples provide the LLM with examples for reference. The two examples are selected as follows: for the first input sentence, we take the first two examples from the training dataset. For the second input sentence, we take the next two examples, and so on until one of the datasets is empty. This ensures that the LLM is always provided with two new examples. We choose this method because the training dataset and test dataset are approximately the same size. After the two examples, the instruction follows, which the purpose remains the same as in the zero-shot setting.

Few-shot demonstration

Here are 2 examples about extracting disease entities in sentences:

Example 1:
Heterozygous mutations in the CD95 (APO - 1 / Fas) receptor occur in most individuals with <mark>autoimmune lymphoproliferative syndrome</mark> (<mark>ALPS</mark>) and dominantly interfere with apoptosis by an unknown mechanism .

Example 2:
We show that local or global alterations in the structure of the cytoplasmic death domain from nine independent <mark>ALPS</mark> CD95 death - domain mutations result in a failure to bind the FADD / MORT1 signaling protein .

Instruction
Extract disease entities from the next sentence by highlighting them with <mark> and </mark>. If not, output the same sentence. Start the answer with: [output].

Input

The newly described HFE marker provides a new approach to the screening of HH as well as studies of the relationship between the HFE Tyr allele and different disorders including cancer

Output

84: [output] The disease entity extracted from the sentence is:

The newly described HFE marker provides a new approach to the screening of <mark>HH</mark> as well as studies of the relationship between the HFE Tyr allele and different disorders including <mark>cancer</mark>.

Figure 7: A few-shot where after the [output] tag an introductory sentence is included first, followed by the input sentence.

Few-shot demonstration

Here are 2 examples about extracting disease entities in sentences:

Example 1:
The gene responsible for the disease (<mark>PDS</mark>) has been cloned , but its function is as yet unknown and the connection between <mark>thyroid goiter</mark> and <mark>sensory - neural deafness</mark> remains an enigma .

Example 2:
We tested sulfate transport in thyrocytes obtained from <mark>Pendred syndrome</mark> patients and found that it was not defective .

Instruction

Extract disease entities from the next sentence by highlighting them with <mark> and </mark>. If not, output the same sentence. Start the answer with: [output].

Input

Heterozygotes for the rare mutant may be indistinguishable from heterozygotes for the common TSD mutant .

Output

153: The disease entities extracted are:
< heterozygotes > and < TSD >

Figure 8: An few-shot prompt error example the LLaMA-3 model has forgotten to use the <mark> and </mark> tags and has used a different marker instead.

For the post-processing of the few-shot approach, we also use markup patterns to automatically filter out sentences containing biomedical entities from the output. After filtering out sentences containing biomedical entities from the output, each line is manually checked to ensure it contains the correct corresponding sentence. Once verified, the biomedical entities are extracted from the sentences, so that each line contains only the entities associated with that sentence. Following this step, a final manual check is performed to ensure that all entities tagged between <mark> and </mark> have been correctly extracted, preparing the data for the evaluation step. During the manual check, several corrections were made due to certain inconsistencies in the output. These issues arose because the LLaMA3-8B model sometimes generates unique formatting errors that cannot be resolved by a post-processing script. Two of those errors are illustrated in Figure 7 and 8 illustrate some example of those outputs containing these errors. In Figure 7, we see that the input sentence with the <mark> and </mark> tags does not appear immediately after the [output] tag, but instead, there is an introductory sentence followed by the input sentence. In Figure 8, we observe that the LLaMA3-8B model forgot to place the <mark> and </mark> tags and used a different marking tag instead.

3.3.3 Parameter-Efficient Fine-Tuning

Instruction tuning is the process of fine-tuning a LLM using a dataset of natural language instructions and corresponding outputs, enabling the model to better understand and follow task-specific prompts.

For instruction tuning, we first need to train the LLM on a dataset to perform a specific task. The same dataset is used, that were also used as example for the few-shot settings. The training data for instruction tuning is formatted as a JSON file containing a list of dictionaries. Each dictionary includes three fields: instruction, input, and output. The instruction field describes the task the model should perform. The input field is a sentence from the training set of one of the three datasets, which may contain zero, one, or multiple entity mentions. The output field contains the biomedical entities if they are present in the input sentence. If no entities are found, the output is set to "No disease/chemical/gene entities found.", depending on the dataset being used at that moment. A small example of the training data structure is illustrated in Listing 1.

Listing 1: A small example of the training data structure.

```
[
    {
        "instruction": "Extract only the disease
           entities from the input. List them as a
           comma-separated response. If no disease
           entities are present, output 'No disease
           entities found.",
        "input": "Identification of APC2 , a
           homologue of the adenomatous polyposis coli
            tumour suppressor .",
        "output": "adenomatous polyposis coli tumour"
    },
    {
        "instruction": "Extract only the disease
           entities from the input. List them as a
           comma-separated response. If no disease
           entities are present, output 'No disease
           entities found.",
        "input": "Here , we report the identification
            and genomic structure of APC homologues
           .",
        "output": "No disease entities found."
    }, ....
```

Instruction
Extract the disease entities from the input. If no disease entities are present, output 'No disease entities found'.
Input
ATM mutations and phenotypes in ataxia - telangiectasia families in the British Isles : expression of mutant ATM and the risk of leukemia, lymphoma, and breast cancer.
Output
ataxia - telangiectasia, leukemia, lymphoma, breast cancer

Figure 9: An example of a prompt of the LLaMA3-8B model after instruction tuning with Unsloth.

After preparing the training data, the first step is loading and initializing a LLM. Next, LoRa (Low-Rank Adaptation) is configured and applied as a PEFT (Parameter-Efficient Fine-Tuning) method. This approach modifies the model architecture by introducing low-rank matrices into specific layers, such as the attention layers. These matrices represent a small fraction of the original parameters and approximate the weight updates required during instruction tuning. Importantly, only the parameters of these low-rank matrices are updated during training, while the original model weights remain frozen. This substantially reduces memory usage and computational cost, making the instruction tuning process faster and more resource-efficient. Before training, we added LoRA adapters to the model, restricting updates to a small subset of parameters (approximately 1–10%). The LoRA configuration used a rank of 16 (`r=16`), a scaling factor of 16 (`lora_alpha=16`), no dropout (`lora_dropout=0`), and targeted the attention and feedforward projection layers (`q_proj`, `k_proj`, `v_proj`, `o_proj`, `gate_proj`, `up_proj`, `down_proj`). Gradient checkpointing was enabled in the optimized "unsloth" mode to reduce VRAM usage and allow larger batch sizes, and a fixed random seed of 3407 was used to ensure reproducibility. After preparing the model with LoRA, the training process began with explicitly defined parameters: a per-device batch size of 2, gradient accumulation across 4 steps, a learning rate of 2×10^{-4}, 5 warmup steps, weight decay of 0.01, and the AdamW 8-bit optimizer combined with a linear learning rate scheduler. Early stopping was applied with a patience of 3 steps and a minimum loss improvement threshold of 0.001 to avoid overfitting.. These parameters control how the fine-tuning updates are applied to the trainable components introduced by LoRA. This approach requires less memory and computation, making instruction tuning faster and more resource-efficient. Next, the training process begins. Before training starts, the entries in the JSON file are reformatted into a specific template called alpaca_prompt, which structures each data example as a text prompt. Additionally, the EOS_TOKEN (<|end_of_text|>) is appended to each prompt to mark the end of a sequence. This reformatting process transforms each original dataset entry into a new format, stored under the key "text", containing the complete formatted prompt. Once instruction tuning with Unsloth is complete, either by reaching the maximum training steps or by early stopping when the loss no longer improves, the model is ready to generate predictions from the given instruction and input. Figure 9 illustrates an prompt example with the output.

To perform instruction tuning on the LLaMA3-8B and PMC-LLaMA-13B model, we use the Python library Unsloth because there are no computers or machines available that meet the hardware requirements for instruction tuning. Unsloth is a Python package designed to accelerate instruction tuning of LLMs by optimizing the training process. It achieves faster instruction tuning by using techniques such as LoRA, which reduces the number of trainable parameters, and efficient data loading pipelines. These optimizations minimize computational overhead, allowing instruction tuning to be performed with reduced hardware requirements and shorter training times.

Instruction Extract the disease entities from the input. If no disease entities are present, output 'No disease entities found'.
Input Therefore , progression to S phase is accompanied by the aggregation of nuclear BARD1 polypeptides into BRCA1 nuclear dots .
Output Breast cancer

Figure 10: A generated output of the instruction tuned LLaMA3-8B model with unsloth. The output includes fabricated entities ('breast cancer'), which is not present in the input text.

Instruction Extract the disease entities from the input. If no disease entities are present, output 'No disease entities found'.
Input We identified four germline mutations in three breast cancer families and in one breast - ovarian cancer family . among these were one frameshift mutation , one nonsense mutation , one novel splice site mutation , and one missense mutation .
Output Breast cancer, breast cancer, breast cancer, breast cancer, breast cancer, breast cancer, ...

Figure 11: A prompt example where the LLM repeats the extracted entities multiple times.

Instruction

Extract the disease entities from the input. If no disease entities are present, output 'No disease entities found'.

Input

One of the principal features of the DM mutation is an extraordinarily high level of somatic mosaicism , due to an extremely high degree of somatic instability both within and between different tissues .

Output

disease entities: DM, somatic mosaicism, somatic instability

Figure 12: A prompt example where the LLM appends extra words to the output, even if it is not requested in the instruction.

Instruction

Extract the gene entities from the input. If no gene entities are present, output 'No gene entities found'.

Input

A DNA binding protein was identified which binds to two novel target - like sequences : (i) at the 5 ' flanking site of the breakpoint junction of chromosome 8 in a patient with T - acute lymphoblastic leukemia (ALL) carrying the t (8 ; 14) (q24 ; q11) rearrangement and (ii) on chromosome 1 in three of five T - ALL patients with the t (1 ; 14) (p32 ; q11) rearrangement.

Output

gene1, gene2, gene3, gene4

Figure 13: A prompt example where the model produces output that is entirely unrelated to the given input.

The post-processing step is performed to evaluate its performance on a test dataset. Since the output consists solely of either the biomedical entity or "No disease/-chemical/gene entity," it is not necessary to filter the output as in the zero-shot and few-shot settings. However, given that we are still working with a LLM, there are exceptional cases where manual corrections are required. On average, these cases occur approximately once every 30 outputs generated by the LLM. LLMs can suffer from hallucinations, where the model generates fabricated information that does not exist in the input, Figure 10 illustrates an example of this case. Furthermore, there are cases where the LLM repeats the extracted entities multiple times, as shown in Figure 11 or adds extra words to the output. This occurs even when this is not requested in the instruction, an example is presented in Figure 12. In some instances, the model produces output that is entirely unrelated to the given input, an example of this case is illustrated in Figure 13.

3.4 Evaluation metrics

For each entity type, Precision, Recall, and F1 score were calculated to evaluate the model's performance in extracting medical entities using NER, by comparing the predicted entities with a "gold standard" of correct entities for the three datasets introduced in section 3.1. We use two matching methods: strict matching and partial matching to calculate Precision, Recall, and F1 score. In strict matching, an entity is only considered correct if it exactly matches an entity in the ground truth. In partial matching, an entity is correctly extracted if there is an word overlap between a predicted entity and a gold standard entity.

3.5 Extraction of long, complex and emerging entities with LLM

In this section, we describe the methodology for extracting long and complex entities from biomedical texts. In the study by Keloth et al.[22], LLMs demonstrated strong performance on NER tasks involving disease, chemical, and gene entities. However, their performance remains uncertain when applied to entities that are longer and more complex, or to previously unseen and newly emerging entities, that are for example absent from the training data. This raises important questions about the models' ability to generalize beyond commonly encountered biomedical terms and adapt to the dynamic nature of scientific language. To address this challenge, we propose the following methodology.

For this research question, we found two datasets: the BC5CDR-Disease dataset and the NLM-Chem dataset. Both datasets contain long and complex biomedical entities, including multi-word and emerging entities. This make them suitable for evaluating how well LLMs handle more challenging entity structures. The BC5CDR-disease dataset is a dataset that contains disease entities, including many rare diseases. Rare diseases often have long, descriptive names, frequently composed of multiple words and including complex terminology or references to genetic or pathological features, making them particularly challenging for NER tasks. In comparing to the NCBI-Disease dataset, the BC5CDR-Disease dataset is more recent, meaning it includes new kind of diseases as well as emerging abbreviations that may not have been seen during model training.

The NLM-Chem dataset contains chemical entities. We use this dataset in our experiment on extracting long and complex entities because it is a relatively new dataset that has been published. In contrast to the BC2GM dataset, which was published in 2008, there is a 13-year gap between the two. Over these years, many new chemical compounds have been discovered, and naming conventions have become more complex and specialized. As a result, the entities found in the NLM-Chem dataset are often longer, more technical, and include emerging terminology that are not present in the BC2GM dataset. Furthermore, the NLM-Chem dataset was developed using a large and diverse set of reasonably recent PubMed articles, ensuring coverage across a wide range of biomedical subdomains. This diversity introduces varied linguistic contexts and entity structures, including systematic

chemical names and domain-specific abbreviations.

The two datasets were not immediately suitable for the experiment and required pre-processing. For the experiment, the datasets needed to contain complex entities, which refers to entities that are longer and consist of multiple words, also known as multi-word or nested entities. In addition, the entities needed to be novel or emerging, meaning they were previously unseen by the LLM. To ensure that the test set contained emerging/novel entities, we found a more recent dataset, as described in paragraph above. After this, we identify and eliminate overlapping entities between the training and test sets. Overlapping entities were defined as those that match exactly, while partial overlaps were excluded. A script was developed to automate this process: it searched for exact matches between entities in the training and test data and removed any duplicates from the test set.
Once the entities were finalized, further preprocessing was applied to the datasets themselves. Both datasets consist of separate text segments in which entities are annotated using markup tags. First, the text fragments were split into individual sentences, ensuring that each line contained exactly one sentence. Following this, the annotated entities were de-tagged to create an unlabeled version of the data, which will later be used as input for the entity extraction task.

As a result, these two dataset offer an opportunity to evaluate whether LLMs are capable of extracting long, complex, and emerging biomedical entities. This provides insight into their ability to generalize beyond familiar terms and adapt to evolving scientific language.

After preprocessing, the datasets were prepared for use as input in the entity extraction task. Both BC5CDR-Disease and NLM-Chem are used as test sets in this experiment. For the LLM we use the same LLM that are described in section 3.2. The LLM adaptation technique applied in this experiment is instruction-tuning, as it has proven to be the most effective approach for improving model performance in extracting biomedical entities through NER, according to the results presented in the Results section 4. The instruction tuning setup for the two LLMs follows the same setup as described in Section 3.3.3, with the only difference being that the two new datasets are now used as test sets. Once the predictions from the LLMs are obtained, the extracted entities are compared to the gold standard annotations in order to calculate the Precision, Recall, and F1-score.

4 Results

4.1 Evaluation of different kind of LLM Adaptation Techniques

In this section, we evaluate the performance of LLMs for entity recognition by comparing different LLM adaptation techniques. We do this by calculating the Precision, Recall, and the F1-score to measure their effectiveness. Tables 1, 2, and 3 present results obtained using the LLaMA3-8B model on three different datasets with three adaptation techniques: zero-shot, few-shot, and instruction tuning. These metrics are computed using the test set of the dataset described in Section 3.1.

In table 1, we use the NCBI-disease dataset to evaluate the performance of the LLaMA3-8B model. The table shows that for strict matching, both precision and recall are highest with instruction tuning. For partial matching, precision and recall scores highest with instruction tuning. The F1-score is highest with instruction tuning for both strict and partial matching.

Dataset = NCBI(disease)				
Model	Strict/partial match	Precision	Recall	F1-score
LLaMA3-8B (zero-shot)	Strict	0.2529	0.4035	0.3109
	Partial	0.3814	0.6106	0.4696
LLaMA3-8B (few-shot)	Strict	0.4113	0.4250	0.4180
	Partial	0.5437	0.5723	0.5576
LLaMA3-8B (Unsloth)	Strict	0.7152	0.6102	0.6585
	Partial	0.7870	0.6597	0.7177

Table 1: Performance of LLaMA3-8B Models on NCBI(disease) dataset with Strict and Partial match metrics.

In Table 2, the same LLM is used, but with a different dataset, namely BC5CDR-chemical. For this dataset, the strict match results show that precision is highest with instruction tuning, while recall is highest in the zero-shot setting. For partial match, precision and recall is highest with instruction tuning. The F1-score is the highest for both strict and partial match with instruction tuning.

Dataset = BC5CDR(chemical)				
Model	Strict/partial match	Precision	Recall	F1-score
LLaMA3-8B (zero-shot)	Strict	0.3714	0.6247	0.4659
	Partial	0.4406	0.7385	0.5519
LLaMA3-8B (few-shot)	Strict	0.4184	0.4373	0.4277
	Partial	0.5617	0.5934	0.5771
LLaMA3-8B (Unsloth)	Strict	0.6714	0.6850	0.6781
	Partial	0.7262	0.7215	0.7239

Table 2: Performance of LLaMA3-8B Models on BC5CDR(chemical) dataset with Strict and Partial Match Metrics.

Table 3 evaluates the performance of the LLaMA3-8B model on a third dataset, namely BC2GM-gene. Precision is highest for both strict and partial matching with instruction tuning. In strict matching setup, recall is highest with instruction tuning, while for partial matching, recall reaches its highest value in the zero-shot setting. The F1-score, similar to the other datasets, is highest with instruction tuning.

Dataset = BC2GM(gene)				
Model	Strict/partial match	Precision	Recall	F1-score
LLaMA3-8B (zero-shot)	Strict	0.1945	0.3337	0.2458
	Partial	0.3612	0.6336	0.4601
LLaMA3-8B (few-shot)	Strict	0.2208	0.2612	0.2393
	Partial	0.3650	0.4385	0.3984
LLaMA3-8B (Unsloth)	Strict	0.3586	0.3824	0.3701
	Partial	0.5292	0.5762	0.5517

Table 3: Performance of LLaMA3-8B Models on BC2GM Dataset(Gene) with Strict and Partial Match Metrics.

4.2 Replication of PMC-LLaMA-13B model

This section replicates the experiment with the PMC-LLaMA-13B model described in the paper by Keloth et al.[22]. First, an initial experiment was conducted to test how well the standard PMC-LLaMA-13B model in the paper of Keloth et al.[22] could extract biomedical entities. We used 25 examples from the test dataset to assess whether the PMC-LLaMA-13B model could successfully identify these entities. The result of the experiment showed that the PMC-LLaMA-13B model correctly identified 0 out of 25 examples. Therefore, we proceed with the instruction tuning. However, since the authors of the paper did not share their fine-tuning code, we use the python package Unsloth for the instruction tuning process. To perform instruction tuning, we used the training dataset from each corpus, namely NCBI-Disease, BC5CDR-Chemical, and BC2GM-Gene to instruction tune the LLM. First, we had to convert the training dataset as in the paper of Keloth et al.[22] into a JSON file. The JSON file contains a list of dictionaries, each dictionary having three fields: instruction, input, and output. The "instruction" describes a task that the model needs to perform. The "input" is a sentence from the training set of one of the three datasets and may contain zero, one, or more mentions of entities. The "output" is the sentence with entities marked using <mark> and </mark> tags if present, or the exact input sentence if no entities are found. After the PMC-LLaMA model is fine-tuned, we use the test set to evaluate its performance. For the evaluation, the structure of the test dataset remains the same.

Table 4 presents the results of the performance of the PMC-LLaMA model. The result shows that the PMC-LLaMA model does not perform well in extracting biomedical entities compared to the results of the PMC-LLaMA model presented in the paper by Keloth et al.[22]. The performance of the PMC-LLaMA model from their paper is also shown in Table 4. There is a significant difference in Precision, Recall, and F1-score for both strict and partial matching between the two sets of results.

Table 4 highlights the substantial difference in performance for the NCBI dataset. The PMC-LLaMA model with PEFT achieves a precision of 0.1958, a recall of 0.1460, and an F1-score of 0.1672 for strict matching, while the results reported in the paper by Keloth et al.[22] demonstrate substantially better performance on the same dataset. Their strict matching results show a precision of 0.877, a recall of 0.851, and an F1-score of 0.864. This indicates a poor performance for the PMC-LLaMA model with PEFT compared with the PMC-LLaMa from the paper of Keloth et al.[22] in identifying biomedical entities. For partial matching the PMC-LLaMA model with PEFT achieves a precision of 0.2684, a recall of 0.1949, and an F1-score of 0.2258. In contrast, the partial matching from the paper Keloth et al.[22] achieves a precision of 0.938, a recall of 0.914, and an F1-score of 0.926. In conclusion, the results from the PMC-LLaMa model in the paper show much higher Precision, Recall, and F1-scores compared to the PMC-LLaMa model with PEFT.

Dataset	Strict/partial match	Precision	Recall	F1-score
NCBI-Disease	Strict	0.1958 0.4328** 0.877*	0.1460 0.4266** 0.851*	0.1672 0.4297** 0.864*
BC5CDR-Chemical	Strict	0.0994 0.5042** 0.935*	0.0769 0.5431** 0.870*	0.0867 0.5229** 0.902*
BC2GM-Gene	Strict	0.1604 0.4798** 0.834*	0.1059 0.3402** 0.823*	0.1276 0.3981** 0.828*
NCBI-Disease	Partial	0.2684 0.5426** 0.938*	0.1949 0.5315** 0.914*	0.2258 0.5370** 0.926*
BC5CDR-Chemical	Partial	0.2111 0.5537** 0.957*	0.1556 0.5862** 0.892*	0.1791 0.5695** 0.924*
BC2GM-Gene	Partial	0.1604 0.6077** 0.959*	0.1059 0.4329** 0.943*	0.1276 0.5056** 0.951*

Notes: **Values with early-stopping.
*Values from the paper Keloth et al.

Table 4: Performance metrics of PMC-LLaMA model (Precision, Recall, and F1-score) for strict and partial matches across three datasets: NCBI-Disease, BC5CDR-Chemical, and BC2GM-Gene.
The first value represents instruction tuning without early stopping, the second value represents instruction tuning with early stopping, and the third value corresponds to the results from the Keloth et al.[22] paper.

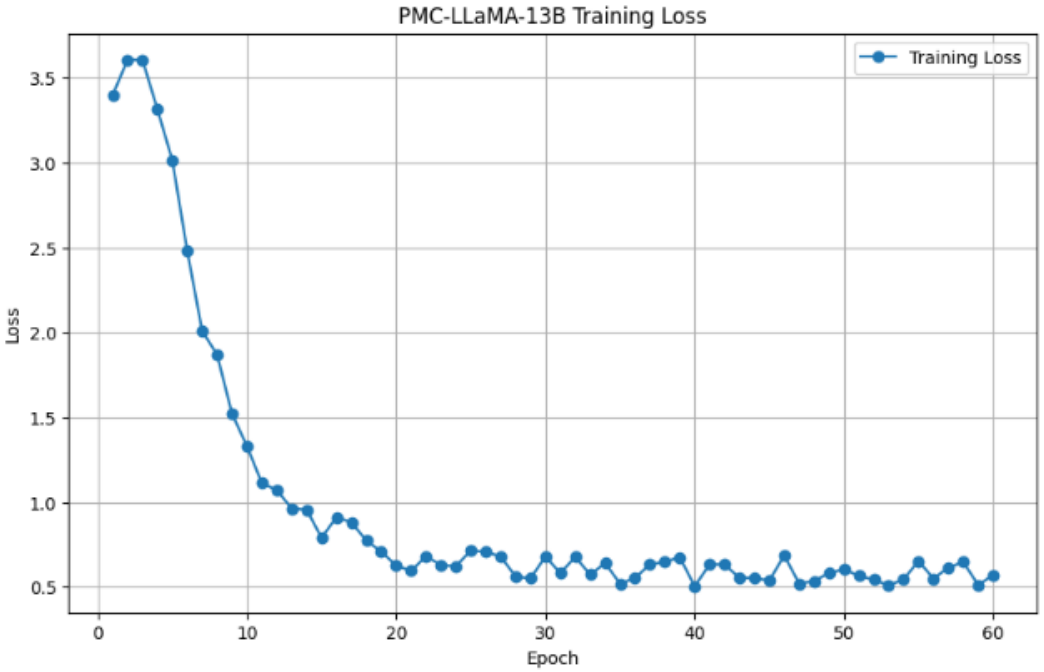

Figure 14: Training Loss Curve for PMC-LLaMA-13B during Instruction Tuning

To further investigate the performance gap, we analyzed the training dynamics of the PMC-LLaMA-13B model. Figure 14 presents the training loss over the course of instruction tuning, showing a consistent decrease in loss as the epochs progress. In Figure 14 we observe a high loss value at the beginning of the training process. This reflects the model's initial lack of understanding and poor performance before it has learned from the data. As the model undergoes more training epochs, the loss value drops quickly. This reduction in loss indicates that the model is effectively learning from the data and adjusting its weights to better predict the outcomes based on the training data. This indicates that the model has reached a point where additional training provides minimal improvement. At this point, the model has effectively captured the underlying patterns in the training data, and further training is unlikely to enhance its performance. The steady decline and followed by a stabilization of the loss indicates that the model is learning effectively from the training data and that the model is adjusting its weights and improving its predictions based on the provided training data.

In Figure 14, we observe that the point at which the loss stabilizes is reached early in the training process. Training beyond this point results in minimal improvements and can sometimes lead to overfitting, where the model performs well on the training data but poorly on unseen data. To address this, we implement an "early stopping" mechanism. Early stopping stops the training process once the model's performance on a validation set stops improving, thereby preventing unnecessary training and reducing the risk of overfitting. This approach ensures that the model generalizes well to new data while optimizing the training efficiency. In Table 4, the results are presented for the training of PMC-LLaMA with early stopping. We observe that the Precision, Recall, and F1-score have improved compared to PMC-LLaMA without early stopping. PMC-LLaMA with early stopping achieves a precision of 0.4328, a recall of 0.4266, and an F1-score of 0.4297 for strict matching on the NCBI-Disease dataset. In comparison, PMC-LLaMA without early stopping has a precision of 0.1958, a recall of 0.1460, and an F1-score of 0.1672 for strict matching. With early stopping, the model achieves a precision of 0.5426, a recall of 0.5315, and an F1-score of 0.5370. In contrast, without early stopping, its performance drops substantially to a precision of 0.2684, a recall of 0.1949, and an F1-score of 0.2258. Thus, we observe that early stopping has a positive impact on PMC-LLaMA's performance. However, the results still fall short of those reported in the paper of Keloth et al.[22].

4.3 Qualitative analysis

In this section, we conduct a qualitative analysis to identify the cases where LLaMA3-8B and PMC-LLama-13B struggle. By analyzing the errors and limitations of these models, we aim to uncover specific failure patterns in both models. LLaMA3-8B outperforms PMC-LLaMA-13B without instruction tuning or a few-shot setting, this is because LLaMA3-8B model has a greater generalization capacity. The LLaMA3-8B consists of a collection of pretrained and instruction-tuned generative text models. These models are designed to generate text and code in response to prompts, making them suitable for a wide range of natural language

processing tasks. In contrast, PMC-LLama -13B is an open-source language model designed specifically for medical applications. The model underwent an instruction-tuning step using training samples that focus on medical conversations, medical rationale question-answering, and QA pairs derived from knowledge graphs.

For this reason PMC had some common struggles, the following points highlight specific struggles the model faced:

- PMC-LLaMA-13B has a difficulty in recognizing a wide variety of diseases, genes, and chemicals. For example, in the disease dataset, PMC-LLaMA-13B was able to correctly identify well-known diseases among the general public, such as various types of cancer (e.g., leukemia, tumors, breast cancer, melanoma) and other common diseases like diabetes. However, one of its shortcomings in disease recognition was its difficulty in identifying abbreviations for diseases. For instance, it struggled with abbreviations such as AGU, which stands for Aspartylglucosaminuria (AGU) a severe autosomal recessive lysosomal storage disorder or AS, which refers to Ankylosing Spondylitis (also known as axial spondyloarthritis), an inflammatory disease that can lead to the fusion of spinal vertebrae over time. This struggle in disease recognition is caused to the fact that PMC-LLaMA-13B was not explicitly trained for biomedical entity extraction but rather for providing medical advice, similar to how a general practitioner advises patients. This suggests that PMC-LLaMA-13B is better in more general medical language than in specialized biomedical terminology.

- Mistakes in the word "disease" recognition: A common mistake that PMC-LLaMA-13B also made was marking the word 'disease' itself as an entity, even though 'disease' is not a specific illness on its own.

- Challenges in gene recognition: While PMC-LLaMA-13B successfully identified well-known genes such as hemoglobin, prolactin, and glucagon, it struggled with recognizing gene abbreviations and less commonly known genes.

- Limitations in chemical entity recognition: Because PMC-LLaMA-13B was not explicitly trained for biomedical entity extraction but instead designed to deliver patient-centered medical advice, it struggled with identifying and marking chemical entities, reflecting its limitations in handling specialized biomedical terminology. This limitation is also evident in the results shown in Table 4, where the model's performance in recognizing chemical entities was notably lower compared to other datasets. This suggests that PMC-LLaMA-13B is better suited for generating general medical advice, rather than for technical tasks such as biomedical entity recognition that require domain-specific expertise.

The LLaMA3-8B model faces challenges in accurately extracting biomedical entities in text. The following points highlight challanges the model encounters:

- Incorrect entity boundaries: One of the challenges for the LLaMA3-8B model is that it does not always mark entities in the correct location. For example, in the disease dataset, the entity that should be marked according to

the gold standard is "hex A deficiency", but LLaMA3-8B incorrectly marks "incomplete hex A deficiency" instead. Another example is 'inherited breast cancer', where the correct entity, according to the gold standatd, is to mark 'breast cancer', but the model incorrectly includes the word 'inherited' as part of the entity.

- Failure to capture multiple instances of the same entity type: Another common error in the LLaMA3-8B model is that it only returns one entity of a specific type, even when there are multiple instances of the same entity type in a sentence. For example, in the sentence: "A hot spot for PTEN mutation in <mark>CD</mark> was identified in exon 5 that contains the PTPase core motif, with 13 of 30 (43%) <mark>CD</mark> mutations identified in this exon." As you can see there are two disease entities (CD), but the LLaMA3-8B model only returns one of them.

4.4 Evaluation of long and complex entity extraction

For this experiment on extracting long and complex biomedical entities, we use two unseen external datasets: the BC5CDR-Disease dataset, which contains disease-related entities, and the NLM-Chem dataset, which includes chemical entities. These datasets serve as the out-of-distribution test sets, and we evaluate the performance of the LLMs using Precision, Recall, and F1-score as the evaluation metrics. The results are presented in Table 5 and Table 6.

Model = LLaMA3-8B				
Test data	**Match Type**	**Precision**	**Recall**	**F1-score**
Disease Entities				
NCBI (Internal)	Strict	0.7152**	0.6102**	0.6585**
BC5CDR-disease (External)	Strict	0.5648	0.5468	0.5557
NCBI (Internal)	Partial	0.7870**	0.6597**	0.7177**
BC5CDR-disease (External)	Partial	0.6180	0.5341	0.5730
Chemical Entities				
BC5CDR (Internal)	Strict	0.6714**	0.6850**	0.6781**
NLM-Chem (External)	Strict	0.6473	0.6482	0.6478
BC5CDR (Internal)	Partial	0.7262**	0.7215**	0.7239
NLM-Chem (External)	Partial	0.6995	0.6859	0.6926

Notes: ** Values from Tables 1 and 2.

Table 5: Performance of the LLaMA3-8B model on external (BC5CDR-disease and NLM-Chem) and internal (NCBI and BC5CDR) test data using strict and partial match metrics.

Table 5 presents the performance of the LLaMA3-8B model on the two unseen external biomedical named entity recognition datasets (BC5CDR-disease and NLMChem-chemical) under both strict and partial match criteria. The values marked with ** in Table 5 represent the instruction tuning results obtained on the test set corresponding to the train set, these results can be found in Tables 1 and 2.

Under the strict match criterion, which requires an exact match between predicted and gold-standard entities, the LLaMA3-8B model achieved a precision of 0.5648, recall of 0.5468, and an F1-score of 0.5557 on the unseen external dataset. In comparison, on the in-domain dataset, the model obtained a higher precision of 0.7152, a recall of 0.6102, and a F1-score of 0.6585. A similar trend was observed on the NLMChem dataset, under strict match criterion the LLM achieved a Precision of 0.6473, recall of 0.6482 and a F1-score of 0.6478, while on the in-domain dataset it achieved a higher precision of 0.6714, but also a higher recall and F1-score of 0.6850 and 0.6781. We observe that the performance of LLaMA3-8B model is higher on the in-domain dataset instead on the unseen external dataset.

Model = PMC-LLaMA-13B				
Dataset	**Match Type**	**Precision**	**Recall**	**F1-score**
Disease Entities				
NCBI	Strict	0.4328**	0.4266**	0.4297**
BC5CDR-disease (External)	Strict	0.1913	0.1865	0.1889
NCBI	Partial	0.5426**	0.5315**	0.5370**
BC5CDR-disease (External)	Partial	0.3324	0.3197	0.3259
Chemical Entities				
BC5CDR	Strict	0.5042**	0.5431**	0.5229**
NLM-Chem (External)	Strict	0.4762	0.5271	0.5004
BC5CDR	Partial	0.5537**	0.5862**	0.5695
NLM-Chem (External)	Partial	0.4736	0.5531	0.5102

Notes: ** Values from Tables 4.

Table 6: Performance of the PMC-LLaMA-13B model on external (BC5CDR-disease and NLM-Chem) and internal (NCBI and BC5CDR) test data with strict and partial match metrics.

Table 6 presents the outcomes obtained by applying PMC-LLaMA-13B model to the same two biomedical datasets. Under the strict match setting for the BC5CDR-disease dataset, PMC-LLaMA-13B showed a lower performance on the unseen external dataset than on the in-domain testset, it achieved a precision of 0.1913, recall of 0.1865, and F1-score of 0.1889. In contrast, on the in-domain testset, the model performed substantially better, with a precision of 0.4328, recall of 0.4266, and F1-score of 0.4297. This difference in performance suggests that the model struggles to accurately identify the disease entities on the unseen dataset. On the NLMChem-chemical dataset, the model's strict match performance was more stable. It achieved a precision of 0.4762, recall of 0.5271, and F1-score of 0.5004 on the external testset, compared to 0.5042 precision, 0.5431 recall, and 0.5229 F1-score, on the in-domain testset. Although performance still declines slightly with

the entities on the external testset, the drop is less significant than with the test set corresponding to the train set.

In terms of partial match evaluation, where overlaps is allowed. The model still showed a lower performance on the unseen external testsets. For BC5CDR-disease dataset, the F1-score for this dataset is 0.3259 and for the in-domain testset is the F1-score 0.5370. A similar trend can be observed for the NLMChem dataset, where the F1-score for the in-domain testset (F1 = 0.5695) is higher than the F1-score for the unseen external testset (F1 = 0.5102). We observe that PMC-LLaMA-13B struggles more with extracting long and complex entities compared to the LLaMA3-8B model.

4.4.1 Effect of removing known entities on Model Performance

Table 7 presents the performance of the LLaMA3-8B model on two biomedical NER datasets: BC5CDR (for disease entities) and NLM-Chem (for chemical entities). For each dataset, results are reported using both strict and partial matching criteria, and are further divided into two evaluation conditions: the Original test set, which contains all entities, and the Filtered test set, from which overlapping entities with the Original test set have been removed.

Model = LLaMA3-8B				
Dataset	**Match Type**	**Precision**	**Recall**	**F1-score**
BC5CDR (Disease)				
Original	Strict	0.5648	0.5468	0.5557
Filtered	Strict	0.5288	0.4627	0.4935
Original	Partial	0.6826	0.6541	0.6681
Filtered	Partial	0.6180	0.5341	0.5730
NLM-Chem (Chemical)				
Original	Strict	0.6473	0.6482	0.6478
Filtered	Strict	0.5918	0.6039	0.5978
Original	Partial	0.6995	0.6859	0.6926
Filtered	Partial	0.6383	0.6410	0.6396

Table 7: Performance of LLaMA3-8B on in-domain vs. out-domain test data. Original represent the dataset, which contains all entities. Filtered represent the dataset, which overlapping entities with the Original dataset have been removed.

These results were obtained by creating two versions of each dataset, an Original version that includes all entities and a Filtered version where any entity overlapping with the Original was removed. In other words, any entity found in the Original dataset was removed from the Filtered version to ensure that the entities are unseen to the LLM. This setting allows us to better evaluate the model's ability to generalize beyond memorized entities. To achieve this, we developed a Python script that identifies and removes overlapping entities between the versions van BC5CDR en NLM-Chem dataset. The script first determines how frequently each

entity from the Original test set occurs in the Filtered test set. If an entity appears more than once in the Filtered test set, the script removes the entity and the full sentence in which it occurs. This filtering process ensures that the remaining examples in the Filtered test set contain only entities that are novel to the model.

The results in Table 7 reveal a consistent drop in performance across both datasets under the filtered condition. For the BC5CDR dataset using strict matching, precision decreased from 0.5648 to 0.5288, recall from 0.5468 to 0.4627, and F1-score from 0.5557 to 0.4935. Under partial matching, the performance also declined, namely precision dropped from 0.6826 to 0.6180, recall from 0.6541 to 0.5341, and F1-score from 0.6681 to 0.5730.

A similar trend is observed in the NLM-Chem dataset. In the strict setting, precision decreases from 0.6473 to 0.5918, recall decrease from 0.6482 to 0.6039 and the F1-score drops from 0.6478 to 0.5978. Under partial matching, precision drops from 0.6995 to 0.6383, recall from 0.6859 to 0.6410, and the F1-score decreases from 0.6926 to 0.6396. These findings demonstrate that the LLaMA3-8B model performs substantially better when test set entities are known from the training domain. The decline in performance on the filtered dataset shows the model's limited ability to generalize to entirely novel biomedical entities.

Model = PMC-LLaMA-13B				
Dataset	**Match Type**	**Precision**	**Recall**	**F1-score**
BC5CDR (Disease)				
Original	Strict	0.1913	0.1865	0.1889
Filtered	Strict	0.1324	0.1206	0.1262
Original	Partial	0.3324	0.3197	0.3259
Filtered	Partial	0.2476	0.2229	0.2346
NLM-Chem (Chemical)				
Original	Strict	0.4762	0.5271	0.5004
Filtered	Strict	0.4264	0.4156	0.4209
Original	Partial	0.4736	0.5531	0.5102
Filtered	Partial	0.4157	0.4725	0.4423

Table 8: Performance of PMC-LLaMA-13B on in-domain vs. out-domain test data. Original represent the dataset, which contains all entities. Filtered represent the dataset, which overlapping entities with the Original dataset have been removed.

For the PMC-LLaMA-13B model we did the same experiment as the LLaMA3-8B model. The results in Table 8 show a decrease in performance for the PMC-LLaMA-13B model when comparing the original and filtered versions of the BC5CDR-disease dataset. Under the strict matching criterion, precision drops from 0.1913 to 0.1324, recall from 0.1865 to 0.1206, and the F1-score from 0.1889 to 0.1262. This indicates a substantial reduction in the model's ability to correctly identify disease entities when previously seen entities are removed. Under partial matching, the performance follows the same downward trend. Precision decreases from

0.3324 to 0.2476, recall from 0.3197 to 0.2229, and the F1-score from 0.3259 to 0.2346.

The results for the PMC-LLaMA-13B model on the NLM-Chem (Chemical) dataset also show a performance drop when comparing the original and filtered test sets. Under the strict matching criterion, precision decreases from 0.4762 to 0.4264, recall from 0.5271 to 0.4156, and the F1-score from 0.5004 to 0.4209. In the partial matching setting, the trend is similar: precision from 0.4736 to 0.4157, recall drops from 0.5531 to 0.4725, and the F1-score from 0.5102 to 0.4838. These findings also show that PMC-LLaMA-13B performs better when entities from the training domain are present in the test set, and struggles more when evaluating on entirely novel entities.

5 Discussion

5.1 Performance discrepancies with prior work

The performance of the PMC-LLaMA-13B model reported in the paper by Keloth et al.[22] achieves a F1-score of 0.864 on the NCBI-disease dataset, 0.902 on the BC5CDR-Chemical dataset, and 0.828 on the BC2GM-gene dataset, it achieves a better performance than the performance of the PMC-LLaMA-13B model in this thesis, which obtains F1-scores of 0.4297, 0.5229, and 0.3981 on the respective datasets. The lower performance of the PMC-LLaMA-13B model in this work compared to that of Keloth et al.[22] can be caused by various factors. A factor can be that the issue may lie in the structure of the prompt, as a prompt can be unclear or ambiguous. To address this, we used several different prompts. The current prompt is: "Given a sentence, extract disease entities from it by highlighting with <mark> and </mark>. If no entities are present, output the same sentence." We experimented with various types of instructions to see if the issue of poor performance could be linked to the prompt structure. Initially, we adjusted the prompt so that the model would only extract biomedical entities, as the problem might stem from the model not understanding how or when to use the <mark> and </mark> tags. Since the PMC-LLaMA-13B model should generate only the biomedical entity as output or "no disease/chemical/gene entity found" if the input sentence contains no entities. The training dataset was modified to fit this new structure. The model was then retrained using the updated dataset. However, this new prompt structure did not lead to any improvement in the model's performance. We also tried shorter prompts, such as "Extract the disease entities from the input sentence. If no disease entities, output 'No disease entities found'," as well as longer ones, like "Given an input sentence, extract only the disease entities and output only those entities. If no entity is found, output 'No entities found". It is possible that the initial prompt was too vague due to its length, or perhaps it was too unclear, and a more detailed prompt was needed. To test this, we conducted a small test for each of the new prompts using 10 sentences from the test dataset, and the results showed that the model produced the same output as with the initial prompt structure. This indicates that the changes in the prompt design did not have any noticeable impact on the model's performance.

Another issue could be that the training dataset is of poor quality. However, this is unlikely, as the same training data is also used for the Llama3-8B instruction tuning, and it is the same dataset used in the experiments in the paper by Keloth et al.[22].

Another possible explanation for the poor performance that could explain the lower performance of the PMC-LLaMA model is that the instruction tuning performed with PEFT may not have been sufficient to fully optimize the model's capabilities. However, in the paper of Hu et al.[18] the authors conducted extensive experiments to evaluate the effectiveness of PEFT against full fine-tuning across a variety of downstream tasks and pre-trained model architectures, including RoBERTa, DeBERTa, GPT-2, and GPT-3. Their results demonstrated that LoRA achieves a comparable or better performance while substantially reducing the computational

and memory overhead. In fact, LoRA consistently performed on par with, and in some cases exceeded, the results of full fine-tuning in tasks such as text classification, sequence labeling, and language generation. To conclude, while the suboptimal performance of the PMC-LLaMA model could be attributed to insufficient fine-tuning with PEFT, the findings of Hu et al. [18] suggest that LoRA is a robust method for fine-tuning, capable of achieving results comparable to or exceeding full fine-tuning in various tasks.

5.2 Self-verifiction/hallucinations

Although the performance of the LLMs in the results section shows good outcomes for the NER task, the results could have been further improved by implementing measures to address hallucination and overprediction issues commonly observed in LLMs [10]. In the paper 'GPT-NER: Named Entity Recognition via Large Language Models' by Shuhe Wang et al.[48], the authors propose GPT-NER as a method for identifying location entities in input text. During their experiments, they encountered issues with hallucination and overprediction, which occur when models, even when provided with demonstrations, confidently mislabel irrelevant or NULL inputs as valid entities. For example, in their experiments, GPT-3 incorrectly identifies "Hendrix" as a location entity in the sentence "Rare Hendrix song sells for $17", highlighting the model's tendency to overpredict.

To mitigate the well-known issues of hallucination and overprediction in LLMs, Wang et al.[48] introduce a self-verification strategy as part of their GPT-NER approach. This strategy implies that after an entity is initially extracted by the LLM, the model should be prompted again to verify the validity of that extraction. This is done by asking the model a yes/no question about whether the extracted word belongs to a specific entity type (e.g., location) in the context of the input sentence. This setup ensures the model to reflect on its own predictions and correct potential hallucinations.

The effect of the self-verification strategy was also shown in the results, namely without self-verification, GPT-NER using sentence-level embeddings achieved an F1-score of 92.68. With the self-verification strategy added to the GPT-NER, the F1-score improved to 94.17. These results demonstrate that self-verification not only reduces hallucination but also leads to performance gains.

5.3 Limitations

Our work has a number of limitations that need to be noted. The first limitation is in the evaluation procedure used to obtain the performance metrics, namely Precision, Recall, and F1-score for each LLM adaptation technique. In our experiments, due to the high computational cost, each model was evaluated on the test dataset only once to generate these scores. Each individual run often took several hours to complete, which made multiple runs impractical within our resource constraints. However, running the evaluation multiple times and averaging the results would have provided a more reliable estimate of model performance by accounting for

potential variability across runs. This approach could reduce the influence of random fluctuations and improve the robustness of the reported metrics. Future work should aim to incorporate multiple evaluation runs to strengthen the validity of performance comparisons.

The second limitation is the work relates to the replication of the PMC-LLaMA-13B model experiments. The original fine-tuning code used by Keloth et al.[22] for the PMC-LLaMA-13B model was not publicly released, which made it challenging to reproduce their reported results accurately. In their paper, Keloth et al.[22] showed that the PMC-LLaMA-13B model performed well on the biomedical NER task, achieving high Precision, Recall, and F1-scores in extracting biomedical entities. Due to the absence of the original fine-tuning code, we used PEFT methods as an alternative approach to instruction tune the PMC-LLaMA-13B model. However, the results obtained using PEFT did not match the performance reported by Keloth et al.[22]. In fact, our scores were lower compared to the result of his paper. As a result, the weaker performance of our replicated PMC-LLaMA-13B model had a cascading effect on subsequent experiments, which were all based on this initial instruction tuned version. Future work would benefit from access to the original fine-tuning implementation used by Keloth et al.[22], which would allow for more accurate replication and fairer comparison, as well as a clearer understanding of the specific training configurations that contributed to the original model's reported success.

The tird limitation lies in the restricted number of few-shot examples used for few-shot learning, due to the limited availability of training data, which prevented us from including more training examples per query. We experimented with only two demonstrations per query, which may not fully capture the range of task variability. While increasing the number of examples could potentially improve performance, the findings of Chen et al.[12] show that such gains are not always guaranteed. Furthermore, we did not explore strategies for optimizing example selection. Recent research Margatina et al.[29] has shown that the quality and relevance of examples are more important than quantity. The authors formulated the problem of selecting few-shot demonstrations as an active learning task and demonstrates that selecting semantically similar examples to the test query lead to notable performance improvements. Incorporating more targeted selection methods could therefore be a promising direction for future work.

6 Conclusion

In this section, we discuss the research questions according to the experiment results. In this thesis, we explored the potential of LLMs for NER, with the focus on their ability to extract complex and long biomedical entities.

We approached this through three methods: **zero-shot learning, few-shot learning** and **instruction tuning**. The difference between the zero-shot and few-shot learning lies in the use of examples. In a zero-shot setting, the LLM is given only an instruction or task description, without any examples to guide it or learn from it. In contrast, in the few-shot setting, we provided the LLMs with two examples. The purpose of the examples in the few-shot setting is to serve as references that the model can mimic, and to help the model learn patterns from the examples to produce more accurate outputs.

The second approach is instruction tuning, where an LLM was trained on a dataset consisting of natural language instructions paired with corresponding outputs. This enables the model to better understand and follow task-specific prompts, improving its ability to extract biomedical entities in text. Now we will look at the research questions and draw the final conclusion of this thesis.

RQ1. *How does the performance of zero-shot learning compare to few-shot learning, and how do both approaches compare to instruction tuned LLMs for medical NER?*

In the zero-shot setting, there are no task-specific examples provided. In contrast to the few-shot setting, there are two task-specific examples provided to guide the model. These examples serve as a references for the model, helping the model better understand the task. The results show that few-shot learning performs better than the zero-shot learning on the NCBI-dataset(disease). However, in the BC5CDR-dataset (chemical) and BC2GM-dataset (gene), the zero-shot setting performance better than the few-shot setting. But across the three datasets, the instruction tuned LLM achieves the highest performance overall, with results nearly twice as high as those obtained through zero-shot or few-shot learning. Overall, the instruction tuned LLM performs better on biomedical NER tasks than zero-shot and few-shot learning, highlighting the benefits of domain-specific adaptation.

RQ2. *Can the results reported by Keloth et al.[22] on using large language models for named entity recognition be reproduced by following their proposed methodology and using the same LLMs?*

The methodology proposed by Keloth et al.[22] was successfully implemented in this study to a large extent. However, a limitation was encountered during the fine-tuning stage. The original fine-tuning code used by Keloth et al.[22] for the PMC-LLaMA-13B model was not publicly released, making it difficult to replicate their results with complete accuracy. In their work, Keloth et al.[22] demonstrated strong performance of the PMC-LLaMA-13B model on biomedical NER tasks, re-

porting high Precision, Recall, and F1-score in extracting biomedical entities. Due to the absence of the original fine-tuning code, we used PEFT as an alternative to fine-tune the PMC-LLaMA-13B model. However, the results obtained through PEFT did not match the performance reported by Keloth et al.[22]. In fact, our scores were lower compared to the result of his paper. To conclude, the proposed methodology was largely successfully implemented in this thesis. However, the fine-tuning stage could not be fully replicated due to the absence of the original fine-tuning code. To address this, we developed and applied an alternative fine-tuning approach.

RQ3. *To what extent are large language models (LLMs) capable of accurately extracting long and complex biomedical entities, including rare or emerging terms?*

To investigate the capability of LLMs in accurately extracting long and complex biomedical entities, including rare or emerging terms. The experiment in this thesis evaluates two instruction-tuned LLMs: LLaMA3-8B and PMC-LLaMA-13B. The evaluation is conducted using two types of biomedical datasets: an in-domain dataset, on which the models are trained and tested, and an out-of-domain dataset, which contains data that is not used during training. So, the models are instruction tuned on the training split of the in-domain dataset and than evaluated on both the in-domain test set and the out-of-domain test set. Results show that both models perform substantially better on the in-domain test set, demonstrating higher performance in biomedical entity extraction. In contrast, performance drops on the out-of-domain test set, indicating that the models struggle to generalize to unfamiliar or more complex biomedical terms.

To further assess the models' generalization capabilities, we conducted a follow-up experiment in which we filtered out all entities from the out-of-domain test set that also appeared in the training data of the LLMs. This ensured that the remaining entities were completely unseen by the models. The results showed an even more substantial drop in performance on this filtered out-of-domain test set compared to the original out-of-domain data. While instruction-tuned LLMs show strong performance in extracting biomedical entities within familiar contexts, their ability to accurately identify long, rare, or emerging biomedical terms that are entirely novel remains limited and requires further improvement.

Future research should incorporate multiple evaluation runs to enhance the reliability and validity of performance comparisons across LLM adaptation techniques. Access to the original fine-tuning implementation used by Keloth et al.[22] would also be beneficial, as it would enable more accurate replication, fairer comparison, and a clearer understanding of the training setup that contributed to the model's reported performance. Furthermore, incorporating more targeted examples for the few-shot learning, especially those based on semantic similarity between test queries and demonstrations, presents a promising direction for improving model performance in future studies.

References

[1] URL https://ftp.ncbi.nlm.nih.gov/pub/lu/BC5CDR/.

[2] URL https://ai.meta.com/blog/meta-llama-3/.

[3] URL https://ftp.ncbi.nlm.nih.gov/pub/lu/NLMChem/.

[4] URL https://pubmed.ncbi.nlm.nih.gov/.

[5] . URL https://www.reddit.com/r/LocalLLaMA/comments/1dli093/llama3_8b_output_control/.

[6] . URL https://crfm.stanford.edu/2023/03/13/alpaca.html.

[7] URL https://huggingface.co/google/flan-ul2.

[8] Amrita Anandika and Smita Mishra. A study on machine learning approaches for named entity recognition. pages 153–159, 05 2019. doi: 10.1109/ICAML48257.2019.00037.

[9] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text, 2019. URL https://arxiv.org/abs/1903.10676.

[10] Mark Braverman, Xinyi Chen, Sham M. Kakade, Karthik Narasimhan, Cyril Zhang, and Yi Zhang. Calibration, entropy rates, and memory in language models, 2019. URL https://arxiv.org/abs/1906.05664.

[11] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

[12] Qingyu Chen, Yan Hu, Xueqing Peng, Qianqian Xie, Qiao Jin, Aidan Gilson, Maxwell B. Singer, Xuguang Ai, Po-Ting Lai, Zhizheng Wang, Vipina K. Keloth, Kalpana Raja, Jimin Huang, Huan He, Fongci Lin, Jingcheng Du, Rui Zhang, W. Jim Zheng, Ron A. Adelman, Zhiyong Lu, and Hua Xu. Benchmarking large language models for biomedical natural language processing applications and recommendations. *Nature Communications*, 16(1), April 2025. ISSN 2041-1723. doi: 10.1038/s41467-025-56989-2. URL http://dx.doi.org/10.1038/s41467-025-56989-2.

[13] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope,

James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. URL https://arxiv.org/abs/2204.02311.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

[15] Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. Ncbi disease corpus: A resource for disease name recognition and concept normalization. *Journal of Biomedical Informatics*, 47:1–10, Feb 2014. doi: 10.1016/j.jbi.2013.12.006.

[16] Roman Dušek, Aleksander Wawer, Christopher Galias, and Lidia Wojciechowska. Improving domain-specific retrieval by nli fine-tuning, 2023.

[17] Wahed Hemati and Alexander Mehler. Lstmvoter: Chemical named entity recognition using a conglomerate of sequence labeling tools - journal of cheminformatics, Jan 2019. URL https://jcheminf.biomedcentral.com/articles/10.1186/s13321-018-0327-2#citeas.

[18] 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. URL https://arxiv.org/abs/2106.09685.

[19] Yan Hu, Qingyu Chen, Jingcheng Du, Xueqing Peng, Vipina Kuttichi Keloth, Xu Zuo, Yujia Zhou, Zehan Li, Xiaoqian Jiang, Zhiyong Lu, Kirk Roberts, and Hua Xu. Improving large language models for clinical named entity recognition via prompt engineering, 2024.

[20] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. 08 2015.

[21] Antonia Höfer and Mina Mottahedin. Minanto at semeval-2023 task 2: Fine-tuning xlm-roberta for named entity recognition on english data. pages 1127–1130, 01 2023. doi: 10.18653/v1/2023.semeval-1.156.

[22] Vipina Keloth, Yan Hu, Qianqian Xie, Xueqing Peng, Yan Wang, Andrew Zheng, Melih Selek, Kalpana Raja, Chih Wei, Qiao Jin, Zhiyong lu, Qingyu Chen, and Wang Qi. Advancing entity recognition in biomedicine via instruction tuning of large language models. *Bioinformatics (Oxford, England)*, 40, 03 2024. doi: 10.1093/bioinformatics/btae163.

[23] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.

[24] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4): 1234–1240, September 2019. ISSN 1367-4811. doi: 10.1093/bioinformatics/btz682. URL http://dx.doi.org/10.1093/bioinformatics/btz682.

[25] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4): 1234–1240, September 2019. ISSN 1367-4811. doi: 10.1093/bioinformatics/btz682. URL http://dx.doi.org/10.1093/bioinformatics/btz682.

[26] Jiao Li, Yueping Sun, Robin J. Johnson, Daniela Sciaky, Chih-Hsuan Wei, Robert Leaman, Allan Peter Davis, Carolyn J. Mattingly, Thomas C. Wiegers, and Zhiyong Lu. Biocreative v cdr task corpus: A resource for chemical disease relation extraction. *Database*, 2016, 2016. doi: 10.1093/database/baw068.

[27] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity, 2022. URL https://arxiv.org/abs/2104.08786.

[28] Ling Luo, Chih-Hsuan Wei, Po-Ting Lai, Robert Leaman, Qingyu Chen, and Zhiyong lu. Aioner: All-in-one scheme-based biomedical named entity recognition using deep learning. 11 2022.

[29] Katerina Margatina, Timo Schick, Nikolaos Aletras, and Jane Dwivedi-Yu. Active learning principles for in-context learning with large language models, 2023. URL https://arxiv.org/abs/2305.14264.

[30] Mónica Marrero, Julián Urbano, Sonia Sánchez-Cuadrado, Jorge Morato, and Juan Miguel Gómez-Berbís. Named entity recognition: Fallacies, challenges and opportunities. *Computer Standards & Interfaces*, 35(5):482–489, 2013. ISSN 0920-5489. doi: https://doi.org/10.1016/j.csi.2012.09.004. URL https://www.sciencedirect.com/science/article/pii/S0920548912001080.

[31] Sudha Morwal, Nusrat Jahan, and Deepti Chopra. Named entity recognition using hidden markov model (hmm). *International Journal on Natural Language Computing*, 1:15–23, 12 2012. doi: 10.5121/ijnlc.2012.1402.

[32] Kalyani Pakhale. Comprehensive overview of named entity recognition: Models, domain-specific applications and challenges, 2023.

[33] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. True few-shot learning with language models, 2021. URL https://arxiv.org/abs/2105.11447.

[34] 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.

[35] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1), January 2020. ISSN 1532-4435.

[36] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model?, 2020. URL https://arxiv.org/abs/2002.08910.

[37] Joshua Robinson, Christopher Michael Rytting, and David Wingate. Leveraging large language models for multiple choice question answering, 2023. URL https://arxiv.org/abs/2210.12353.

[38] Nicholas Rodriguez, Mai Nguyen, and Bridget McInnes. Effects of data and entity ablation on multitask learning models for biomedical entity recognition. *Journal of Biomedical Informatics*, 130:104062, 04 2022. doi: 10.1016/j.jbi.2022.104062.

[39] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. Learning to retrieve prompts for in-context learning, 2022. URL https://arxiv.org/abs/2112.08633.

[40] Burr Settles. Abner: An open source tool for automatically tagging genes, proteins and other entity names in text. *Bioinformatics (Oxford, England)*, 21:3191–2, 08 2005. doi: 10.1093/bioinformatics/bti475.

[41] Zhang Shaodong. Llama3 generates too much content, how to solve this?, Dec 2024. URL https://stackoverflow.com/questions/79030605/llama3-generates-too-much-content-how-to-solve-this.

[42] Larry Smith, Lorraine K Tanabe, Rie Johnson Ando, Cheng-Ju Kuo, I-Fang Chung, Chun-Nan Hsu, Yu-Shi Lin, Roman Klinger, Christoph M Friedrich, Kuzman Ganchev, and et al. Overview of biocreative ii gene mention recognition. *Genome Biology*, 9(S2), Sep 2008. doi: 10.1186/gb-2008-9-s2-s2.

[43] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, Elton Zheng, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. Using deepspeed and megatron to train megatron-turing NLG 530b, A large-scale generative language model. *CoRR*, abs/2201.11990, 2022. URL https://arxiv.org/abs/2201.11990.

[44] Zhen Sun and Xinfu Li. Named entity recognition model based on feature fusion. *Information*, 14(2), 2023. ISSN 2078-2489. doi: 10.3390/info14020133. URL https://www.mdpi.com/2078-2489/14/2/133.

[45] Charles Sutton and Andrew McCallum. An introduction to conditional random fields, 2010. URL https://arxiv.org/abs/1011.4088.

[46] Yoshimasa Tsuruoka and Jun'ichi Tsujii. Boosting precision and recall of dictionary-based protein name recognition. *Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine*, 01 2003. doi: 10.3115/1118958.1118964.

[47] David Vilar, Markus Freitag, Colin Cherry, Jiaming Luo, Viresh Ratnakar, and George Foster. Prompting palm for translation: Assessing strategies and performance, 2023. URL https://arxiv.org/abs/2211.09102.

[48] Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. Gpt-ner: Named entity recognition via large language models, 2023. URL https://arxiv.org/abs/2304.10428.

[49] Leon Weber, Jannes Münchmeyer, Tim Rocktäschel, Maryam Habibi, and Ulf Leser. Huner: improving biomedical ner with pretraining. *Bioinformatics (Oxford, England)*, 36, 06 2019. doi: 10.1093/bioinformatics/btz528.

[50] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652, 2021. URL https://arxiv.org/abs/2109.01652.

[51] Chaoyi Wu, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. Pmc-llama: Further finetuning llama on medical papers, 04 2023.

[52] Zhihao Yang, Hongfei Lin, and Yanpeng Li. Exploiting the performance of dictionary-based bio-entity name recognition in biomedical literature. *Computational biology and chemistry*, 32:287–91, 09 2008. doi: 10.1016/j.compbiolchem.2008.03.008.

[53] Baohang Zhou, Xiangrui Cai, Ying Zhang, and Xiaojie Yuan. An end-to-end progressive multi-task learning framework for medical named entity recognition and normalization. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6214–6224, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.485. URL https://aclanthology.org/2021.acl-long.485.