Universiteit Leiden
The Netherlands

Opleiding Informatica

Open-source Large Language Models
for Named Entity Recognition

Bryan Joseph Owee
S3335690

Supervisors:
Zhaochun Ren
Jujia Zhao

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

07/05/2024

Abstract

As companies transition to automate extracting information from texts, Named Entity Recognition (NER) has become progressively more important. With the rise of Large Language Models, NER is becoming more accessible than ever. However, only some companies have the resources to acquire the best models. This thesis explores the effectiveness of smaller, open-source Large Language Models (LLMs) in Named Entity Recognition (NER) tasks, particularly focusing on cross-domain and business-specific applications. This research evaluates the performance of several smaller, open-source LLMs, OpenLLaMA 3b, Dolly-v2-3b, Falcon-rw-1b, and the closed-source PaLM model. This is done by conducting experiments on cross-domain and business-specific NER datasets. We calculate the accuracy and F1 score to determine which model performs the best in the cross-domain NER experiment. We use humans to evaluate the models responses for the business specific NER experiment. Here we use the ICC score to determine how reliable the responses are. The study aims to identify which models offer the best performance in these domains and assess their practicality for organizations with limited resources.

Contents

1 Introduction

1.1 Background

At the present day, many technological fields are growing and expanding rapidly [1]. One of the technological fields that is also growing, is Natural Language Processing (NLP) [2]. One of the critical tasks in the field of NLP is called information extraction. This task is aimed at automatically retrieving structured information from unstructured plain text [3]. Information extraction is applied in many different sectors such as customer relations, healthcare and legal document analysis [4]. It enables organizations to efficiently manage and analyze large amount of data. This information can then be used to do various things, such as to make informed decisions and operate more efficiently.

A crucial task within the field of Information Extraction is Named Entity Recognition (NER). This task is aimed at identifying named entities such as a person, organization, locations, time, etc [5] and classifying them. NER is a crucial task for extracting valuable and actionable information from raw unstructured text. However, there are still many challenges that need to be solved in NER. One of such complexities is the need to accurately classify entities across different contexts and domains [6]. An example of such a problem is when we want to classify the word "Apple", this can either be referring to the fruit or the tech company. Traditional NER systems struggle with such ambiguity and they struggle even more when they are applied to new domains which they were never specifically trained for [6]. This makes traditional NER systems less diverse and not scalable.

This field has made advancements over the recent years, partially due to the rapid growth of Large Language Models (LLMs) [7]. Some LLMs have showcased how good and efficient they are at NLP specific tasks and how generalizable they are over many domains [7] [8].

Even with LLMs achieving promising results with many NLP tasks, LLMs are still challenged by NER tasks. LLMs can struggle with accurately labeling entities in complex and ambiguous sentences [9]. This problem is then extended when LLMs are tasked to perform cross-domain NER on such sentences. Instead of building and training completely new models, using pre-trained open-sourced LLMs which already have some sense of contextual understanding could save costs in terms of money and time.

This thesis explores how well open-sourced Large Language Models perform in cross-domain and business specific NER tasks.

1.2 Problem Formulation

This thesis aims to answer two key research questions:

RQ1: Which (smaller size) open-sourced LLM performs the best in cross-domain NER.

RQ2: Which (smaller size) open-sourced LLM performs the best in the business domain NER.

Cross-domain NER tasks are one of the most challenging tasks in the realm of NLP [10]. Many factors such as terminologies, abbreviations and context can vary across different domains, making it hard to perform NER. Answering the question of which (smaller size) open-source LLM performs the best in cross-domain NER could lead to understanding more about how adaptable, efficient, and scalable these (smaller size) open-source LLMs are for the NER task. Knowing which model performs the best in cross-domain NER has possible real-world applications. For example, companies that have customer support can apply this LLM to their cross-domain NER task. More specifically this (smaller size) open-source LLM can be used in companies that do not have enough resources to acquire larger and more powerful models.

Extending our research to the business specific domain is important because of the wide range of applications it might cover. This includes finance, customer relation management, marketing and certain legal documentation. Instead of finding a LLM that performs best in cross-domain NER, we can find a LLM that performs the best in business specific NER. The applications stay the same but instead of using the LLM on different domains, this LLM is specifically good at business topics.

Not all companies have the resources to use larger and more expensive LLMs for their NER tasks. These companies might also not have sufficient computing power and memory needed to run these larger models. That is why, in this thesis we also want to focus specifically on smaller LLMs, which require less computing power and memory, to help understand how well these (smaller size) open-source LLMs perform in NER.

To answer the first question, we will conduct a few cross-domain NER experiments on three open-source LLMs, OpenLLaMA 3b, Dolly-v2-3b and Falcon-rw-1b. We will also perform these experiments on one other model which is not open-sourced, the PaLM model. To see which model performs the best, we calculate the accuracy and F1 score. To answer the second question, we will perform the tests on business specific data. We will then evaluate these models with human ratings, using the Intraclass Correlation Coefficient to evaluate their reliability.

2 Definitions

To better understand some terminology of this bachelor thesis, an overview of definitions will be made. These definitions will explain the concepts of some terminology used and in some cases also examples of how such technology will work.

2.1 Natural Language Processing

”Natural Language Processing (NLP) is the computerized approach to analyzing text that is based on both a set of theories and a set of technologies” [11]. Essentially, NLP helps computers communicate with humans in their own language. NLP makes it possible for computers to read text, interpret it, measure sentiment and identify what part of the sentence is important. It is possible to perform NLP operations on texts to complete certain tasks, one of these being Named Entity Recognition (NER) [12].
It is possible to break NLP down into two main components, Natural Language Understand and Natural Language Generation, respectively NLU and NLG for short [13]. NLG is the process

of generating things like sentences and paragraphs that are meaningful from an internal representation. NLU, as the name suggests, is the component that allows machines to understand natural language and analyze it by extracting entities, emotions, and keywords from the given text. Furthermore, this research will be focusing slightly more on this specific component of NLP [13].

2.2 Information Extraction

Information Extraction is defined as "IE systems analyze unrestricted text in order to extract specific types of information" by [14]. The subfield of artificial intelligence that focuses on the process of automatically retrieving structured information and specific entities from text data is called Information Extraction. This transformative technology turns unstructured text into meaningful information by applying computational techniques. The primary goal of information extraction is to facilitate computers to process human (natural) language and convert it into an organized format to be easily visualized and analyzed. To properly extract the correct information from a document, the computer must be able to identify keywords. Therefore this task, labeled Named Entity Recognition (NER), plays a major role in Information Extraction.

2.3 Named Entity Recognition (NER)

Named Entity Recognition (NER) is the part of IE that focuses on identifying and classifying named entities within a text. NER is a NLP task that tries to identify important parts of a natural language text and these are called entities [5]. These entities are pre-defined categories and can include people, organizations, locations and other such values. In short, the task of NER is to identify and classify entities within the text for applications such as summarization and question answering.

As an example, we can use paragraph p and the task of NER is to identify and classify entities e_1, e_2, e_3... within the text. For a practical example, we can assume paragraph p "Obama walks into the White House", with "Obama" and "White House" as entities e_1 and e_2. These entities are then labeled as 'person' and 'location', respectively [5].

2.4 Large Language Models (LLMs)

A Large Language Model is an advanced Artificial Intelligence (AI) that is capable of processing and generating natural language [15]. Such models are particularly good in handling NLP tasks and bare promising advancements for the future of NLP. A LLM can be trained to be effective in a broad range of tasks or a specific task. Although there are mainly three main schools of AI Narrow Artificial Intelligence (ANI), General Artificial Intelligence (AGI), and Superintelligent Artificial Intelligence (ASI), the main AI that has been developed and that we come across in our daily lives, such as ChatGPT, is classified as ANI. These machines are set to specific tasks and have limited abilities to understand and learn any intellectual task that a human can perform [16].

The size of a LLM is often measured by how many parameters a LLM has. In [17] parameters are defined as ”Parameters are the numerical values that an LLM learns during training, that it adjusts to understand language.”. Size or amount of parameters is often a big factor in how well LLMs perform in certain tasks [18]. Having more parameters and thus being larger has benefits. ”It allows them to capture more complex language relationships and handle nuanced prompts.” as cited in [17]. Additionally, size is not the only thing that is important in a model’s performance. Factors such as architecture [19], pre-training data [20] and token count [21] play a role in the model’s performance.

3 Related Work

3.1 Information Extraction

In the field of LLMs and Information Extraction (IE), some research has been done. [3] has conducted a survey that provides a comprehensive exploration of LLMs for IE, categorizing different works based on various IE subtasks and learning paradigms. The writer breaks generative IE down into three sub tasks/types of IE and those are Named Entity Recognition (NER), Relation Extraction (RE) and Event Extraction (EE).

The paper suggests [3] that multiple studies have pointed researchers in the direction of LLMs for NER tasks. Several studies have proposed methods using LLMs as an approach to NER tasks, such as GPT-NER, Uni-NER and PromptNER. The research mentioned in this paper suggests that LLMs have shown potential to enhance zero-shot NER performance through fine-tuning and training-free self-improving frameworks.

3.1.1 Relation Extraction (RE)

RE is the task of identifying relationships between entities in natural language texts. This paper [3] mentions different approaches such as QA4RE and GPT-RE to address the poor performance of LLMs on RE tasks. QA4RE introduces a framework that enhances an LLM’s performance by aligning RE tasks with question-answering tasks. GPT-RE incorporates task-aware representations and enriching demonstrations with reasoning logic, to improve the low relevance between entity and relation.

3.1.2 Event Extraction (EE)

EE focuses on the task of extracting events from a natural language text. This paper [3] mentions other proposed methods such as, BART-Gen, Text2Event, and Code4Struct as LLM-based methods to improve EE. The paper [3] collected experimental results from recent studies concerning LLM based EE. The results show that the number of methods using LLMs for either zero-shot or few-shot learning is still small. The results also indicate that generative methods significantly surpass discriminative approaches, particularly in Argument Classification metrics, further showing the possible implications of generative LLMs for EE.

3.1.3 Universal Information Extraction (UIE)

This paper [3] also showcases Universal Information Extraction which aims to handle various and diverse IE tasks and domains using a unified framework, either based on natural language-based LLMs (NL-LLMs) or code-based LLMs (code-LLMs). This paper mentions previous works such as InstructUIE, and GoLLIE which explore the use of NL-LLMs and Code-LLMs to model multiple IE tasks effectively.

3.2 Generative Information Extraction

The paper of [3] categorizes methods based on their learning paradigms such as, supervised fine-tuning, few-shot, zero-shot and data augmentation to highlight the different commonly used approaches for adapting LLMs to IE.

3.2.1 Zero-shot Learning

Zero-shot learning is a method of training LLMs to generalize to unseen tasks without any explicit training examples for specific IE tasks. In certain scenarios, zero-shot learning can leverage similarities with common cases to generate informed predictions or analyze customer reviews for new products [22].
Additionally, it enhances recommender systems by suggesting items that users have not previously encountered, based on their preferences, even for objects that have not been seen before [22].
Zero-shot learning faces significant challenges in ensuring the capacity to generalize effectively across untrained tasks and domains. Due to the large amount of knowledge embedded within, LLMs show impressive abilities in zero-shot environments [3].

3.2.2 Few-shot learning

Few-shot learning is a type of system that is commonly used in areas where there is very limited labeled training data available [3]. Unlike, zero-shot learning, few-shot learning uses a small number of examples per class to generalize to new, similar tasks. This method tries to train LLMs to generalize for tasks and domains they have not been exposed to. This paper [3] states that because of the very limited labeled training data, leads to challenges like overfitting and difficulty in capturing complex relationships. This paper states that LLMs with increased parameters tend to show more impressive generalization capabilities in few-shot learning environments.

3.2.3 Supervised Fine-Tuning (SFT)

Supervised fine-tuning is a common method for training LLMs on IE tasks using labeled data. It works by inputting all training data to fine-tune the LLM, this allows the model to capture the patterns in the data and also allows it to generalize well to unseen IE tasks. The survey also mentions Deepstruct, UniNER and GIELLM as supervised fine-tuning methods [3].

The method of supervised fine-tuning offers benefits such as high accuracy and task-specific optimization [23]. The explanation given by [23] clearly explains why SFT yields such high accuracy, "By exposing the LLM to labeled data, the model learns to make more accurate predictions by aligning its outputs with the desired labels". This makes it easier to train models for specific tasks, making them specialist in certain tasks. By learning from specific data, the models can be tailored to achieve high accuracy in specific tasks.

However, because it learns so well from labeled data, there are certain drawbacks of SFT. SFT is very dependent on high-quality data because it runs the risk of learning from biased data, which will let the model suffer from bias [24]. Additionally, using SFT can make a model bad at generalizing. Learning from specific datasets makes the models good at specific tasks but this prevents the models from being generalized to unseen data outside the scope of the training data [24].

3.2.4 Data Augmentation

"Data augmentation involves generating meaningful and diverse data to effectively enhance the training examples or information, while avoiding the introduction of unrealistic, misleading, and offset patterns" [3]. This paper divides data augmentation into three strategies, Data Annotation, Knowledge Retrieval, Inverse Generation.

Data Annotation is the strategy that directly generates labeled data using LLMs [3]. Research that is referenced in this survey proposes a method called LLMaAA to improve accuracy and data efficiency. Another method referenced in this paper, called AugURE enhances the diversity of positive pairs for unsupervised RE, and introduces margin loss for sentence pairs.

Knowledge retrieval strategies enhance IE by utilizing LLMs for retrieving relevant information, similar to retrieval-augmented generation (RAG) [25]. PGIM [26] employs ChatGPT for efficient entity prediction, while [27] uses synthetic context retrieval datasets for NER on long documents.

Inverse Generation is the strategy that prompts LLMs to generate natural text or questions based on provided data. Research in this survey [28] demonstrates that with SynthIE, LLMs can produce high-quality synthetic data for complex tasks by inverting the task directions. Another research [29] suggests that STAR generates structures from valid triggers and arguments, then generates passages with LLMs [3].

The survey [3] states that each strategy has its pros and cons. Data Annotation can directly meet task requirements but the ability of LLMs for structured generation needs improvements. Knowledge Retrieval provides additional information but it may hallucinate and introduce noise problems. Inverse Generation is aligned with the question-answer paradigm of LLMs but requires structural data.

3.3 Named Entity Recognition

3.3.1 Prompt Learning with Type Related features

There have been multiple researches done to look into the added value of LLMs on NER tasks. A research [30] conducted has explored a framework named PLTR that aims to address some challenges in few-shot NER. PLTR is short for Prompt Learning with Type-Related features (PLTR) and it can automatically extract entity type-related features (TRFs) to identify useful knowledge in the source domain and enhance knowledge transfer. This proposed framework aims to bridge the gap between training data and out-of-domain (OOD) data. Furthermore, to enhance the model's ability to capture important knowledge from the source domain, PLTR identifies tokens that are closely associated with specific entity types. This significantly improves the model's generalization to unseen domains. This unique approach and advancements allow these researchers to obtain an improved NER performance, allowing models to generalize effectively to new unseen domains with limited labeled examples.

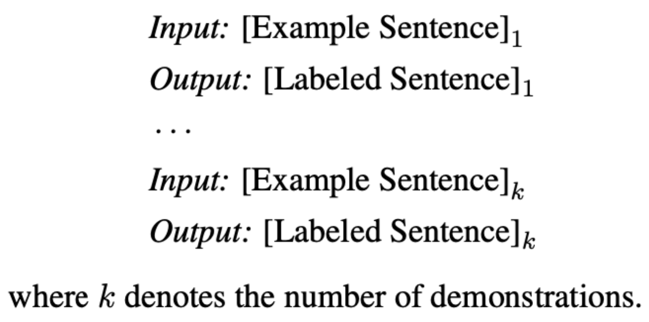
This method [30] was experimented with in both in-domain and cross-domain settings. The experiments demonstrate the effectiveness of the PLTR framework by surpassing SOTA methods, showing significant advancements in few-shot NER performance. The model's capability to accurately recognize named entities across various domains was severely improved. The key contributor to enhancing the model's generalization and NER performance is the incorporation of TRFs and the generation of prompts based on these features.

This paper [30] shows how the PLTR framework offers a promising solution to the challenges faced in few-shot cross-domain NER. It showcases a practical approach to improving model adaptability and performance across various domains. It showcases how NER performance can be enhanced even in an environment with limited labeled examples.

3.3.2 GPT-NER approach

Earlier, there was a method mentioned in [12] called GPT-NER, which was used in few-shot and zero-shot NER tasks. This method introduces a new perspective to the topic of using LLMs for NER tasks. This paper addresses how it is possible to harness the huge potential of LLMs for NER tasks. The researchers of this method pointed out that there was a large difference in how the two systems functioned [9]. Since LLMs widely focus on text generation and not sequence labeling tasks such as NER, LLMs still sometimes find it challenging to successfully perform NER tasks. This paper addresses how it is possible to harness the huge potential of LLMs for NER tasks.

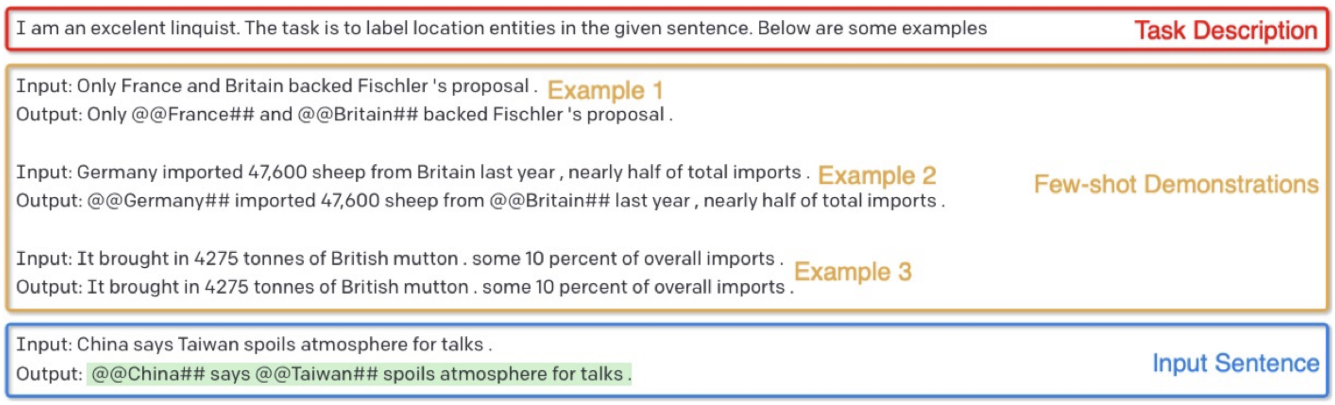
The researchers propose a method called GPT-NER to bridge the gap between the two different tasks. They created a prompt that would transform the sequence labeling task of a NER task to a generation task that an LLM could easily adapt [9]. They first start out with a task description to formally introduce the LLM to the task at hand. After that, in the case of few-shot demonstrations, they exposed the LLM to some examples of how they wanted their output to be. In the end, they asked the LLM to perform a NER task of a sentence. In figure 1 we can see an example of how they introduced the LLM to the examples in the case of few-shot demonstrations.


Input: [Example Sentence]$_1$
Output: [Labeled Sentence]$_1$
. . .
Input: [Example Sentence]$_k$
Output: [Labeled Sentence]$_k$

where k denotes the number of demonstrations.

Figure 1: Example Few-Shot Demonstrations GPT-NER

In figure 2 we can see how they constructed the entire prompt that they fed into the LLM to perform the NER task using the GPT-NER method. As seen in the examples they demonstrated to the LLM, GPT-NER uses the method of marking the entity with special tokens. They used @@ and ## to mark specific entities mentioned in the task description, like the location entities showcased in figure 2, @@entity##.


I am an excelent linquist. The task is to label location entities in the given sentence. Below are some examples

Input: Only France and Britain backed Fischler 's proposal .
Output: Only @@France## and @@Britain## backed Fischler 's proposal .

Input: Germany imported 47,600 sheep from Britain last year , nearly half of total imports .
Output: @@Germany## imported 47,600 sheep from @@Britain## last year , nearly half of total imports .

Input: It brought in 4275 tonnes of British mutton . some 10 percent of overall imports .
Output: It brought in 4275 tonnes of British mutton . some 10 percent of overall imports .

Input: China says Taiwan spoils atmosphere for talks .
Output: @@China## says @@Taiwan## spoils atmosphere for talks .

Figure 2: Example Input Sentence GPT-NER

Additionally, this paper also proposes a new strategy to combat the overconfidence and the common hallucination problem of LLMs. They propose a self-verification strategy in the form of a self-verifying prompt. They apply this strategy in the same form as the few-shot demonstrations showcased in figure 2, task description, few-shot demonstrations and input sentence . As shown in figure 3, they use few-shot demonstrations to boost the accuracy of the self-verifier but in this few-shot demonstration, each demonstration consists of three sentences [9].

They conducted experiments on both flat and nested NER datasets, and achieved comparable performances to fully supervised baselines. They concluded the paper by also stating that GPT-NER has a ”remarkable ability in the low-resource scenario” [9]. The results of GPT-NER are better than that of a supervised model when when there is a very limited amount of training data available.

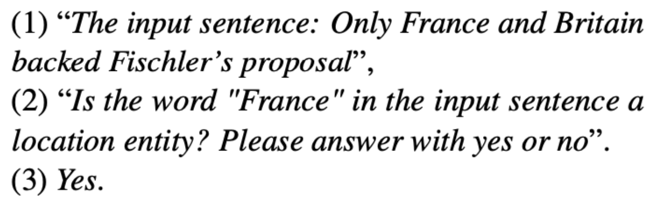

(1) "***The input sentence: Only France and Britain backed Fischler's proposal***",
(2) "***Is the word "France" in the input sentence a location entity? Please answer with yes or no***".
(3) ***Yes.***

Figure 3: Example Few-Shot Demonstration Self-Verifier GPT-NER

3.4 Human Evaluation

There is a statistical measure that is used in the human evaluation of the models. The Intraclass Correlation Coefficient (ICC), is used to assess the reliability of measurements made by different raters for continuous data, similar to the Cohen's Kappa Score. Reliability is defined as "consistency or stability of a measurement"[31]. There are 10 different forms of ICCs [32] and each form considers different situations. According to the McGraw and Wong (1996) Convention, there are certain criteria for determining which form of ICC is needed in an experiment.

The three factors are model, type of relationship and unit [33]. The factor "model", consists of determining how the group is chosen and what kind of group it is. The factor "type of relationship", focuses on the ratings of the raters. Lastly, the factor "unit", focuses on how many raters' ratings we are interested in.

4 Methods

To answer the research questions, we will perform cross-domain and business specific NER on 4 LLMs. These models are three open-source models, OpenLLaMA 3b[1], Falcon-rw-1b[2] and Dolly-v2-3b[3]. The other model which is closed-source, is the PaLM model. The aim of using these models is to evaluate how well open-sourced models perform in cross-domain and business specific NER compared to closed-sourced models, such as PaLM.

4.1 Model Selection and Justification

Since we needed to test four LLMs in this research, the Kaggle environment was used, to utilize the Python language. Kaggle allows its users to make use of an accelerator called GPU T4 x 2. This allows us to run the models and perform the experiments with increased speeds [34]. To conduct the experiments, the spaCy Large Language Model package[4] was used, due to the integration of LLMs into spaCy, turning unstructured responses into robust outputs for various NLP tasks, with no training data required. Using this package with the supported LLMs made the experiments a leveled playing field, making it easier to spot which LLM performs better than the other in a zero-shot learning environment. The three open-sourced models, OpenLLaMA 3b Falcon-rw-1b

[1]$https://huggingface.co/openlm-research/open_llama_3b$
[2]$https://huggingface.co/tiiuae/falcon-rw-1b$
[3]$https://huggingface.co/databricks/dolly-v2-3b$
[4]Documentation can be found at this web address: https://spacy.io/usage/large-language-models

and Dolly-v2-3b, were loaded from the website huggingface, where there are multiple open-sourced models to experiment on.

These models were chosen from the list of available open-sourced models in the spaCy documentation[5]. Due to problems with the 7 billion parameter models being too big to run in the Kaggle environment, the focus shifted to smaller models that could be experimented with in the Kaggle environment. Out of the list of models to choose from, from the spaCy documentation, there were only models with 3 billion parameters that were smaller than the 7 billion parameter models. The Dolly and OpenLLaMA models are both 3 billion parameter models. The Falcon model with 1 billion parameters was intentionally chosen to experiment and see how much impact the size of the model had on NER. The chosen closed-sourced model is from Google Cloud's services, the PaLM model, specifically the text-bison-001 variant of the model. The reason behind this choice, was because Google Cloud Services offered free credits when signing up that could be used to run some of its closed-sourced models. Running this model on Google Cloud's services allows us to use a larger model. With this model, we can compare PaLM to how well the open-sourced LLMs perform.

In the table below, various specifications are listed for each model. The architecture of each model is a transformer-based architecture with a decoder-only structure. One notable thing in the table is that the PaLM model has the highest amount of parameters and layers by far. Also, no information could be found on the amount of layers the Dolly model has.

Models	**Size (Parameters)**	**Layers**	**Pre-trained Knowledge**
		Open Sourced	
Falcon-rw-1b	1 Billion	24	Pre-trained on a mixture of English and multilingual text from the RefinedWeb dataset.
OpenLLaMA 3b	3 Billion	26	Trained on the RedPajama dataset, a large-scale collection of text data comprising various sources, including web pages, books, and academic articles.
Dolly-v2-3b	3 Billion	-	Fine-tuned from an existing open-source language model (pythia-12b), trained on a curated dataset called "databricks-dolly-15k".
		Closed Sourced	
PaLM	8 Billion-62 Billion	118	Pre-trained on a large-scale dataset comprising multilingual text from various sources, including web pages, books, Wikipedia, and code repositories.

Table 1: Table that shows the models and their specifications.

[5]Documentation can be found at this web address: https://spacy.io/usage/large-language-models

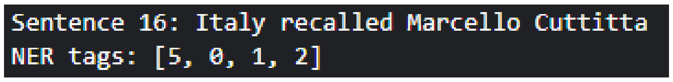
To load in the models, the following pseudo code is used. We import some necessary libraries to allow the code to work. To combat some GPU memory issues, we used certain techniques from the CUDA memory documentation. A tokenizer is also used, which converts raw text into tokens, which are numerical representations that the model can process. We also use a tokenizer specifically designed for the LLaMA model.

```
1. Import necessary libraries.
2. Set environment variable for CUDA memory allocation
to optimize memory usage.
3. Define model options:
   – Option 1: Load and configure the Dolly–v2–3b model.
   – Option 2: Load and configure the Falcon model.
   – Option 3: Load and configure the Open LLaMA model.
   – Option 4: Load and configure the PaLM model.
6. Clear GPU memory cache to free up space.
```

4.2 Data Collection

4.2.1 CoNLL

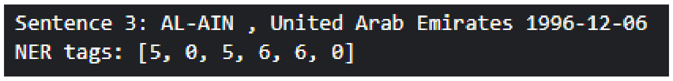
There are many datasets available for cross-domain NER evaluation. Taking the availability, comparability and costs into consideration, the CoNLL 2003 was chosen for the cross-domain task. The CoNLL dataset is an English dataset that focuses on four types of entities: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups [35]. The dataset consists of multiple features, there is id, tokens, pos_tags, chunk_tags and ner_tags. For this research, we will be focusing on the id, tokens and ner_tags. The dataset contains multiple English sentences which are marked by their ID number. These sentences are broken down into tokens, giving each word its token and each token could be an entity. There are also ner_tags which mark a token as being a certain entity or not. The ner_tags are noted in numbers in the following order: 'O': 0, 'B-PER': 1, 'I-PER': 2, 'B-ORG': 3, 'I-ORG': 4, 'B-LOC': 5, 'I-LOC': 6, 'B-MISC': 7, 'I-MISC': 8. "Whenever two entities of type XXX are immediately next to each other, the first word of the second entity will be tagged B-XXX to show that it starts another entity." [35]. The models will be examined on the test split of this specific dataset. It is also important to reduce the amount of ambiguity in the results and to improve the model's accuracy. It is the goal to see how well LLMs perform in NER, that is why it was chosen to leave out examples where the miscellaneous tag was given. Seeing as how this reduced the ambiguity for the models to understand what miscellaneous is defined as this could lead to more accurate results. To get a better understanding of the CoNLL dataset, an example is presented in figure 4.



```
Sentence 16: Italy recalled Marcello Cuttitta
NER tags: [5, 0, 1, 2]
```

Figure 4: Example CoNLL Dataset

Here we can see that Italy is a location that consists of one token, so the assigned entity is B-LOC, which denotes to 5 in the ner_tags. The next entity is Marcello, which is the first part of the name so we assign B-PER to Marcello which then denotes to 1. Cuttitta is the second part of the entity, so we assign I-PER to Cuttitta which denotes to 2.



```
Sentence 3: AL-AIN , United Arab Emirates 1996-12-06
NER tags: [5, 0, 5, 6, 6, 0]
```

Figure 5: Example CoNLL Dataset 2

In this next example, we can see that AL-AIN is a location that consists of one part, so the assigned entity is B-LOC, which denotes to 5 in the ner_tags. The next entity is United Arab Emirates, with the first part of this entity being United. This is assigned as B-LOC and denotes to 5. The next two parts of this entity is Arab Emirates, which are assigned I-LOC, which then denotes to 6 in the ner_tags.

To load the CoNLL dataset into the Kaggle environment we use the following pseudo-code and logic:

```
Input: CoNLL 2003 dataset, maximum number of instances
to collect max_instances = 100.

1. Load the CoNLL 2003 dataset.
2. Initialize an empty list filtered_selected_instances
to store selected instances.
3. Set the maximum number of instances to collect to max_instances.
4. For each instance in the test set of the CoNLL 2003 dataset,
do the following:
    1. Retrieve the ner_tags of the instance.
    2. Check if neither 7 nor 8 is present in ner_tags.
    3. If the condition is met, append the instance
    to filtered_selected_instances.
    4. If the number of collected instances reaches max_instances,
    stop the iteration.
5. Initialize two empty lists: all_sentences and all_ner_tags.
6. For each instance in filtered_selected_instances, do the following:
    1. Convert the tokens of the instance into a sentence.
    2. Retrieve the ner_tags of the instance.
    3. Append the sentence to all_sentences.
    4. Append the ner_tags to all_ner_tags.
7. Retrieve the mapping of NER tags from the dataset's features.
8. Initialize an empty list list_ner_tags.
9. For each NER tag number and name, append a string
representation to list_ner_tags.
0. Initialize an empty dictionary tag_dict.
1. For each item in list_ner_tags, do the following:
    1. Split the item by the ': ' separator to extract the key and value.
    2. Convert the key to an integer.
    3. Add the key-value pair to tag_dict.
2. Output all_sentences and all_ner_tags (if needed).
3. Output the tag_dict mapping NER tag numbers to their entity types.

Output: List of sentences (all_sentences), list of NER tags (all_ner_tags),
dictionary mapping NER tag numbers to their entity types (tag_dict).
```

In this pseudo-code, the datasets library is used to load in the CoNLL dataset. We want to collect the number of sentences we want to use and filter out the instances with miscellaneous in the sentence. In this code, we also add each token together to form sentences, to input into the model for it to perform NER. We also want to store the sentences and NER tags to evaluate the model responses later.

4.2.2 Business Specific Dataset

A possible application of NER is in articles. Specifically, the focus of this thesis is on business related articles. The goal is to analyze how well these models perform in business specific NER. Two articles were chosen in an attempt to reduce any outliers in the performance of these models. These articles were chosen in the business category of the BBC website. ”Instagram and Facebook ads drive profit surge” by the BBC[6], ”Intel axes 15,000 jobs after sales tumble” by Nick Edser & Natalie Sherman[7], are the articles that were chosen for the experiment.

4.3 Evaluation

4.3.1 CoNLL

Following the spaCy documentation[8], it was possible to set up the config file for the NER task. The possible labels are given to the config file under the variable ”labels”. The @llm_models and name variables can be changed to any of these options to fit the loaded model respectively: (spacy.OpenLLaMA.v1, open_llama_3b), (spacy.Dolly.v1, dolly-v2-3b), (spacy.PaLM.v1, text-bison-001) and (spacy.Falcon.v1, falcon-rw-1b). In the code provided above, the PaLM model was being used at the time.

```
config = """
[nlp]
lang = "en"
pipeline = ["llm"]

[components]

[components.llm]
factory = "llm"

[components.llm.task]
```

[6]https://www.bbc.com/news/articles/c0dmel29mk4o
[7]https://www.bbc.com/news/articles/c16j3318n08o
[8]https://spacy.io/usage/large-language-models#example-2

```
@llm_tasks = "spacy.NER.v3"
labels = ["PERSON", "ORGANISATION", "LOCATION"]

[components.llm.model]
@llm_models = "spacy.PaLM.v1"
name = "text-bison-001"
"""

with open("config.cfg", "w") as file:
    file.write(config)
```

To evaluate the model responses, we have to store the model responses somewhere first. With the pseudo-code provided below, we use the spaCy NLP pipeline from the previously mentioned config file and it iterates through a list of all the filtered sentences to identify the named entities in each sentence. The identified entities are stored in the list "model_answers", and the results are printed for convenience. If no entities are found in a sentence, a message is printed stating that there are no entities found.

```
Input: NLP pipeline configuration file config.cfg,
list of sentences all_sentences.

1. Import the necessary libraries for NLP pipeline and memory management.
2. Invoke garbage collection to clear unused objects from memory.
3. Clear the GPU memory cache.
4. Attempt to load the NLP pipeline using the configuration file config.cfg.
5. if the NLP pipeline is loaded successfully
then Print "Model loaded successfully".
6: else Print "Error loading model" with the exception details.
7: Initialize an empty list model_answers to store identified entities.
8: for each sentence in all_sentences do the follwing;
    1. Process the sentence using the NLP pipeline to get doc.
    2. Extract entities and their labels from doc and store them in entities.
    3. Append entities to model_answers.
    4. if no entities are found in entities then Print "No entities found".
    5. else Print the entities found.
15. Print the model_answers list containing all identified entities.

Output: List of identified entities (model_answers) for
each processed sentence.processing.
```

The following piece of code evaluates the model responses and calculates the accuracy and F1-scores. This piece of code imports the libraries such as NumPy and f1_score from sklearn.metrics to calculate certain operations and the F1 score. Two variables are initialized to keep track of the correct predictions and the total predictions made by the model excluding 'O' tags. The other two variables y_true and y_pred are initialized to calculate the F1 score later with the f1_score library.

A loop is created that loops over the interlocked variables of three lists: all_sentences, all_ner_tags and model_answers. In this loop, two lists are created to store token-level NER tags which can be compared later. For the label encoding, a certain tag mapping is used [36] to later translate certain tags to numerical values and vice versa.

To map the model output to token-level tags, we follow the following process. For each entity detected by the model, the corresponding tokens are marked with 'B-', for the beginning of an entity or 'I-', for being inside an entity. This is done depending on their position within an entity. This piece of code also utilizes a try-except function to handle cases where the entity tokens might not be recognizable within the tokenized sentences.

Similarly, the true NER tags are converted to token-level tags using. When these values are translated to token-level tags, these tags are then filtered to exclude 'O' tags. This is done to later calculate an accuracy that focuses on the true values of entities instead of focusing on when a word is predicted as no entity is correct.

In the following step, the code calculates the number of correct predictions by comparing the two variables, adding one increment to the variable of the correct prediction for each match. The total number of predictions, which is stored in the variable filtered_total_pred is updated with the length of filtered_expected_tags.

After the processing of all sentences is done, the code then calculates the filtered accuracy. This is done by dividing two variables, filtered_correct_pred and filtered_total_pred.
Following the sklearn.metrics.f1_score documentation[9], we process the two variables, y_true and y_pred. The F1 score is calculated using the previously mentioned method with the macro averaging method which weighs each class equally.

```
1. Import necessary libraries for numerical operations and F1 score calculation
2. Initialize counters for correct and total predictions.
3. Initialize lists for true and predicted labels.
4. For each sentence, corresponding NER tags, and model answers:
   1. Extract model output and ground truth labels.
   2. Tokenize the sentence.
   3. Initialize lists for token-level tags with 'O' (outside).
   4. Define mappings for label encoding.
   5. Map model output to token-level tags based on entity type and position.
   6. Convert numerical truth labels to token-level tags.
   7. Filter out 'O' tags from both model and expected tags.
   8. Calculate accuracy excluding 'O' tags.
   9. Create a reverse dictionary for tag mapping.
   10. Translate tags to numbers using the reverse dictionary.
   11. Append translated tags to the lists for true and predicted labels.
5. Calculate and print filtered accuracy:
   1. Compute the accuracy as the ratio of correct to
```

[9]Documentation can be found at this web address:https://scikit-learn.org/0.15/modules/generated/sklearn.metrics.f1_score.html

```
        total predictions, excluding 'O' tags.
    2. Print correct predictions, total predictions, and filtered accuracy.
6. Calculate F1 score:
    1. Flatten the lists of true and predicted labels into single lists.
    2. Compute and print the F1 score using the flattened lists.
```

4.3.2 Business Specific NER

An important difference between a dataset and an article is that there are no existing ground truths. Therefore, the evaluation of said responses will be done slightly differently. Three different people with backgrounds in computer science and economics will rate the responses of the models on a scale from one to five, with one being a completely wrong response and five being a perfect identification of an entity.

The pseudo-code that is provided below, is to define the two articles so that we can process these articles later in the notebook. What was discovered through testing, is that the ideal way to get the models to generate entities was to loop over each sentence of the article. To split the article into separate sentences, a small English model by spaCy was used.

```
1. Import the SpaCy library for natural language processing.
2. Load the SpaCy English model for sentence splitting.
3. Define the text of two articles.
4. Define a function to split an article into sentences:
    1. Process the article using the SpaCy model to obtain a document object.
    2. Extract sentences from the document object and
    return them as a list of stripped text.
5. Use the function to split both articles into sentences and
store the results in separate lists.
```

To get the model's answers, we have to alter the code that was created for the CoNLL dataset.

To actually evaluate the model's responses, we first create separate Excel sheets to allow the raters to rate each model's output without the influence of other rater's ratings. The raters will receive an Excel document containing the model's responses from every article and a separate sheet where they can note their rating. The raters will base their score on whether the model has identified all entities in the article and if the model has mislabelled the entity or not. We will use all this information to discuss if these models perform well on such business related articles. After collecting the result, we enter all ratings into a single Excel sheet for an overview of all ratings for each model. This is done to determine the reliability of measurements made by different raters in a continuous data setting. To determine which form of ICC is needed for this experiment, the three factors of ICC must be concluded. Looking at the factor of "model", we select the "Two-way mixed effects model". We are interested in these three people, who specifically have expertise in computer science and economics, to rate the outcomes of the four models [33]. Furthermore, looking at the factor "type of relationship", we select the "Absolute agreement model". This is because we are interested in whether the judges rate similar subjects low and high [33]. Lastly, for the factor

”unit”, we select the ”Mean of raters model” because we do not want to use the ratings one a singular rater as the basis for measurement [33]. In the paper of [32], figure 1 depicts ”A flowchart showing the selection process of the ICC form based on the experimental design of a reliability study”. Following this chart, which is depicted below in figure 6, the outcome of the form that is necessary for this experiment is a ”Two-way mixed effects, consistency, mean of raters” ICC. This form of ICC is commonly denoted as ICC(3,k) following the Shrout and Fleiss (1979) Convention [32].

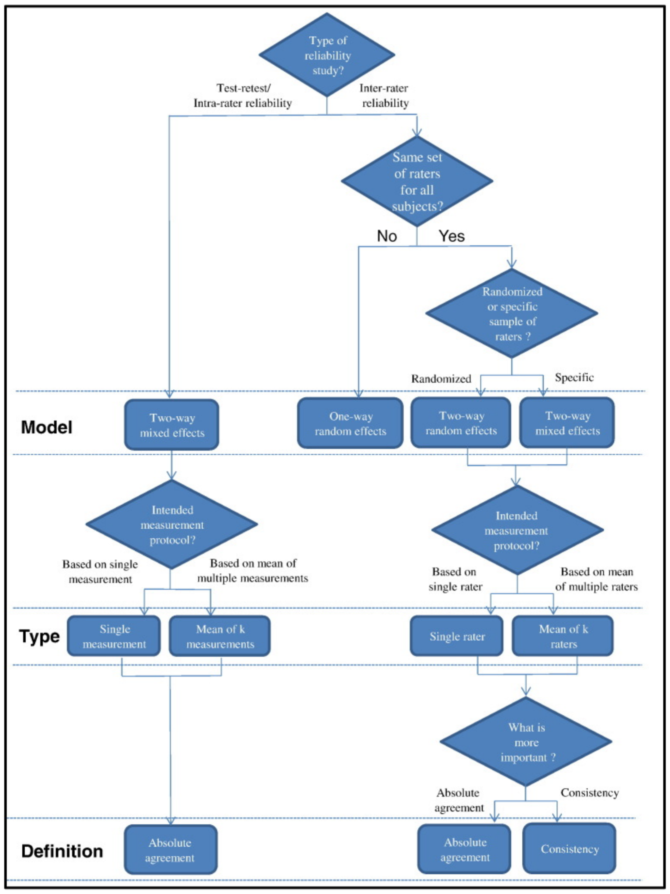


Figure 6: Figure 1 of the paper [32], depicts a flowchart to select the appropriate form of ICC.

To calculate the ICC, we follow the code that is presented in this article [37]. After inputting the correct data and formatting it to our data, we look at the results of the ICC(3,k), seeing as this is the form of ICC that was chosen for this experiment.

5 Experiments

5.1 CoNLL

5.1.1 CoNLL (Sampled 100) Results

English CoNLL2003 (Sampled 100)		
Model	**Accuracy**	**F1**
Falcon-rw-1b	2.13	1.57
OpenLLaMA 3b	13.07	8.46
Dolly-v2-3b	6.38	11.65
PaLM	**73.25**	**43.97**

Table 2: Results of sampled 100 pieces of data for CoNLL2003 dataset.

Table 2 shows the results for the partially sampled CoNLL test set. 100 samples were used to save computation power and time. The Falcon-rw-1b model scores an accuracy of 2.13 and a F1 score of 1.57. The OpenLLaMA 3b scores an accuracy of 13.07 and a F1 score of 8.46. The Dolly-v2-3b. model scores an accuracy of 6.38 and a F1 score of 11.65. The PaLM model scored an accuracy of 73.25 and a F1 score of 43.97.

5.1.2 CoNLL (Sampled 1000) Results

English CoNLL2003 (Sampled 1000)		
Model	**Accuracy**	**F1**
Falcon-rw-1b	2.78	4.15
OpenLLaMA 3b	8.35	10.70
Dolly-v2-3b	6.62	9.70
PaLM[10]	**58.86**	**57.15**

Table 3: Results of sampled 1000 pieces of data for CoNLL2003 dataset.

Table 3 shows the results for the partially sampled CoNLL test set. The Falcon-rw-1b model scores an accuracy of 2.78 and a F1 score of 1.57. The OpenLLaMA 3b scores an accuracy of 8.35 and a F1 score of 10.70. The Dolly-v2-3b model scores an accuracy of 6.62 and a F1 score of 9.70. The PaLM model scores a mean accuracy score of 58.86 and a mean F1 score of 57.15.

5.2 Business Specific NER Results

In tables 4 to 6, the model's responses are placed into tables showing the model's responses, to identifying entities in two articles. As can be noticed, there is no table present for the Falcon-rw-1b model. The reason for the absence of this table is that the Falcon model was unable to identify a single entity in both articles.

PaLM	
Article 1	**Article 2**
(’Meta’, ’ORGANISATION’)	
(’people’, ’PERSON’)	
(’the US state of Texas’, ’LOCATION’)	
(’Mark Zuckerberg’, ’PERSON’)	(’Intel’, ’ORGANISATION’)
(’Meta’, ’ORGANISATION’)	(’Nikkei’, ’ORGANISATION’)
(’Reality Labs’, ’ORGANISATION’)	(’Amazon’, ’ORGANISATION’)
(’Google’, ’ORGANISATION’)	(’Pat Gelsinger’, ’PERSON’)
(’Microsoft’, ’ORGANISATION’)	(’Intel’, ’ORGANISATION’)
(’OpenAI’, ’ORGANISATION’)	(’Apple’, ’ORGANISATION’)
(’Max Willens’, ’PERSON’)	(’Apple’, ’ORGANISATION’)
(’TikTok’, ’ORGANISATION’)	(’Apple’, ’ORGANISATION’)
(’the US’, ’LOCATION’)	(’Tim Cook’, ’PERSON’)
(’Mike Prolux’, ’PERSON’)	(’Apple’, ’ORGANISATION’)
(’Facebook’, ’ORGANISATION’)	(’Apple’, ’ORGANISATION’)
(’Instagram’, ’ORGANISATION’)	
(’Messenger’, ’ORGANISATION’)	
(’WhatsApp’, ’ORGANISATION’)	
(’Susan Li’, ’PERSON’)	

Table 4: PaLM model responses of both articles.

OpenLLaMA 3b	
Article 1	**Article 2**
(’The group’, ’ORGANISATION’)	
(’Mark Zuckerberg’, ’PERSON’)	(’company’, ’ORGANISATION’)
(’Mike Prolux’, ’PERSON’)	

Table 5: OpenLLaMA model responses of both articles.

Dolly-v2-3b	
Article 1	**Article 2**
(’”In a way it’s a smart move by the company to, effectively, introduce Meta AI to its users by forcing them to use it,”’, ’LOCATION’)	(’Intel’, ’ORGANISATION’) (’said’, ’PERSON’) (’catch up with competitors’, ’PERSON’) (’Apple’, ’PERSON’) (’from its services division’, ’ORGANISATION’)

Table 6: Dolly-v2-3b model responses of both articles.

Table 7, shows which value each rater gave to the model’s responses. There are no scores given to the Falcon model because the model did not identify a single entity in both articles. The average score of each model is also displayed in table 7. After calculating the ICC(3,k) value, we get a score of 0.971. Looking at the guidelines that are presented in [32], a score of 0.97 would be classified as "Excellent reliability" [32].

Model	Article	Person 1	Person 2	Person 3	Average
OpenLLaMA 3b	Article 1	2	2	2	1.5
	Article 2	1	1	1	
Dolly-v2-3b	Article 1	1	1	1	1.5
	Article 2	2	2	2	
PaLM	Article 1	4	3	4	**3.67**
	Article 2	3	4	4	
Falcon-rw-1b	Article 1	-	-	-	-
	Article 2	-	-	-	

Table 7: Table that shows the ratings that raters gave to each models response

6 Discussion

6.1 Findings

Looking at the results for the 100 samples cross cross-domain NER. The PaLM model scores significantly higher compared to the other open-sourced models. Out of the results of this smaller sample size experiment, we can conclude that the open-sourced models all have a very low accuracy and F1 score. With OpenLLaMA scoring the highest amongst the open-sourced models, we can also see that the Falcon-rw-1b model achieves by far the lowest scores. This suggests that the open-sourced models are underperforming and need improvements in predicting and handling class imbalances [38].

In a low sample environment, amongst the open-sourced models, we can conclude that the OpenLLaMA model achieves the highest accuracy score. The highest F1 score was achieved by the Dolly model.

In the larger sample size experiment, an internal error occurred on the Google Cloud's services. This issue made it so that the model would give an internal error in the middle of running the experiment. Our solution was to repeatedly run the experiment to achieve sample sizes of 250 samples. We performed this experiment on four different chunks of the data, allowing us to compute the mean accuracy and F1 score. Looking at the results of the larger, 1000 sample size experiment, although the mean accuracy and F1 score were used, the PaLM model shows us a lower accuracy score over a bigger sample size. However, the mean F1 score is slightly higher than in the 100 sample size experiment. This suggests that the model is performing well and outperforming every open-sourced model by a large margin.

In a larger sample environment, the Falcon model achieved a slightly higher accuracy and F1 score. Even with a slight increase in performance, the Falcon model still performs the worst. Amongst the open-sourced models, we can conclude the OpenLLaMA model achieves the highest accuracy and F1 score.

The results of our research which extended to the business specific NER show how detailed and extended the PaLM model was in NER. Comparing these responses to one another, we can see that many entities were left out or misidentified by the other open-sourced models. The rating results also suggest that human raters rated the PaLM model better than the open-sourced models. While not having identified every entity in both articles, the PaLM model does come very close to that ideal result. We see that the open source models struggle in identifying entities, missing out on entities such as ”Microsoft”, ”Meta” and many more. We can also see that the open-source models often misidentify certain entities. In the case of OpenLLAMA, it identified ”The group” to be an organization. Dolly-v2-3b identified ”In a way, it’s a smart move by the company to, effectively, introduce Meta AI to its users by forcing them to use it” as a location. The model also identified ”from its service division” as an organization. As previously mentioned, the Falcon-rw-1b model severely underperformed, not identifying a single entity in both articles. The open-sourced models tend to still make many mistakes compared to the PaLM model.

6.1.1 Possible Explanations

An explanation of this difference in performance could be the size of the models. As previously mentioned models with more parameters, it allows LLMs to capture more complex language relationships [17], which is a big part of NER. In table 6, we can see the size of each model used in the experiments. We see that Falcon-rw-1b is the smallest model, which performed by far the worst in this experiment. Both Dolly-v2-3b and OpenLLaMA 3b have 3 billion parameters, we can see that their results are similar to each other, with OpenLLaMA slightly outperforming Dolly’s model. OpenLLaMA technically has slightly more parameters than the Dolly model, which may explain why OpenLLaMMA performs slightly better than the Dolly model. The biggest model by far is the PaLM model which has at a minimum 8 billion parameters. This increase in size can explain why the PaLM model outclasses every other open-sourced model by far.

Another possible explanation for PaLM’s superior performance may be the quality of its pre-training data. The token count is crucial in the development of these models. Because these models are not able to process whole sentences and paragraphs as humans can, they break the text into smaller blocks. For example, the sentence “Obama walks in the White House” could be broken down into tokens like “Obama”, “walks”, “in”, “the” and “White House”. The model must then process each of these tokens separately to understand the whole sentence. If these models are able to process a higher token count, the model will be able to retain more information, generate longer and more detailed responses, and understand the context better.

The PaLM model was pre-trained using 780 billion tokens. OpenLLaMa, the model that scored second best, was trained for one trillion tokens. In contrast, Falcon-rw-1b was trained only on 350 billion tokens, which could explain its low ranking in the experimental results. The amount of tokens used during training for the Dolly models was not publicly disclosed.

Dolly-3b-v2’s training data involved around 15k instruction/response fine-tuning records generated by employees during a period of two months in 2023. This included passages from Wikipedia as references for instruction classifications like close QA and summarization. The use of Wikipedia as a main source means that this AI model could contain factual errors. Besides linguistic flaws, the dataset being trained manually by the company's employees opens up the possibility of biases and interests of the workers reflecting in the model's answers and ability to identify certain entities in texts.

Possible reasons why Falcon-rw-1b received the lowest score during the experiment could be because this model is trained on English data only. Training the model exclusively in English could bring many limitations [39] such as language barriers, cultural bias, loss of information and an inability to generalize appropriately to other languages. In addition to this, Falcon-rw-1b was trained on a large-scale corpora representative of the web, meaning it would carry the stereotypes and biases commonly encountered online.

In contrast, the PaLM pre-training dataset is comprised of a mixture of filtered webpages, Wikipedia articles, news articles, source code, books, and even social media conversations [40]. So not only is PaLM developed using natural language data, but code obtained from open-source repositories is also adopted in the pre-training dataset. Various amounts of the data sources were also multilingual, such as social media conversations and Wikipedia.

6.2 Future works

A suggestion for future works would be to perform this research on larger open-sourced models such as Mistral-7b and LLaMA 2. These models all have around 7 billion parameters which would increase the performance significantly. Furthermore, another suggestion for future works would be to perform the same experiments on the slightly larger models of this paper. These models would be falcon-7b-instruct[11], OpenLLaMA_7b(_v2)[12] and Dolly-v2-7b[13].

Another suggestion would be to explore the benefits of fine-tuning the models for NER tasks. Exploring how well the models perform after being fine-tuned for the NER tasks can yield valuable insights into whether fine-tuning can enhance performance. This could steer the course of further efforts into the correct direction.

6.3 Limitations

Firstly, due to the lack of resources concerning computing power, we were only able to experiment on the smaller open-sourced LLMs. Although smaller LLMs may not perform as well as closed-sourced LLMs, it is still possible for larger open-sourced LLMs to perform more evenly in NER. Furthermore, there is another limitation concerning the methods for evaluating human experiments. The results of these experiments are quite subjective, thus having such a small sample size, may affect the outcome of said experiment. Although the ICC scores classify the score to be "Excellent reliability" [32], this does not take into account the small sample size of people we had to work with. Lastly, due to Google Cloud's services having internal server issues while testing the PaLM model, the

[11] $https://huggingface.co/tiiuae/falcon-7b-instruct$
[12] $https://huggingface.co/openlm-research/open\_llama\_7b$
[13] $https://huggingface.co/databricks/dolly-v2-7b$

results may also be affected by this. Instead of using the accuracy and F1 scores, the mean accuracy and F1 scores were used for this. Also, due to the internal server errors that Google Cloud was having, the responses of the model may have been affected without us knowing.

6.4 Conclusion

This paper wanted to look at and test how well open-sourced models performed in cross-domain NER compared to propriety closed-sourced like PaLM. Specifically, we wanted to see which (smaller size) open-sourced LLM performed the best in cross-domain NER. The best performing (smaller size) open-sourced model was the OpenLLaMA 3b model. However, even with it outperforming the rest of the open-sourced models, it is still far behind the larger, closed-sourced model PaLM. We also wanted to see which (smaller size) open-sourced LLM performed the best in business specific NER. In this instance, both the OpenLLaMA and Dolly models scored the same mean score. We conclude that although most open-sourced models are somewhat capable of NER, smaller open-sourced LLMs are outclassed by bigger proprietary, closed-sourced models. There might be further explanations for this performance gap, other than only the size of the models. One of the possible explanations for this performance gap could be the quality and variety of the pre-training data. The amount of tokens used during the pre-training could also explain why certain models perform better than others.

References

[1] M. Roser, "This timeline charts the fast pace of tech transformation across centuries," *World Economic Forum*, February 27 2023. Programme Director, Oxford Martin School, University of Oxford.

[2] J. Singh, *Natural Language Processing in the Real World: Text Processing, Analytics, and Classification.* Chapman and Hall/CRC, 2023.

[3] D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng, and E. Chen, "Large language models for generative information extraction: A survey," 2023.

[4] A. A. Awan, "What is named entity recognition (ner)? methods, use cases, and challenges," *DataCamp Blog*, 2021.

[5] A. Roy, "Recent trends in named entity recognition (ner)," 2021.

[6] S. Kamath and R. Wagh, "Named entity recognition approaches and challenges," *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, vol. 6, no. 2, pp. 259–262, 2017.

[7] C. Sun, Z. Yang, L. Wang, Y. Zhang, H. Lin, and J. Wang, "Deep learning with language models improves named entity recognition for pharmaconer," *BMC bioinformatics*, vol. 22, no. Suppl 1, p. 602, 2021.

[8] C.-E. González-Gallardo, T. T. H. Hanh, A. Hamdi, and A. Doucet, "Leveraging open large language models for historical named entity recognition," in *The 28th International Conference on Theory and Practice of Digital Libraries*, 2024.

[9] S. Wang, X. Sun, X. Li, R. Ouyang, F. Wu, T. Zhang, J. Li, and G. Wang, "Gpt-ner: Named entity recognition via large language models," 2023.

[10] C. Jia, X. Liang, and Y. Zhang, "Cross-domain NER using cross-domain language modeling," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (A. Korhonen, D. Traum, and L. Màrquez, eds.), (Florence, Italy), pp. 2464–2474, Association for Computational Linguistics, July 2019.

[11] E. D. Liddy, "Natural language processing," 2001.

[12] B. Jehangir, S. Radhakrishnan, and R. Agarwal, "A survey on named entity recognition—datasets, tools, and methodologies," *Natural Language Processing Journal*, vol. 3, p. 100017, 2023.

[13] D. Khurana, A. Koli, K. Khatter, and S. Singh, "Natural language processing: State of the art, current trends and challenges," *Multimedia tools and applications*, vol. 82, no. 3, pp. 3713–3744, 2023.

[14] A. De Sitter, T. Calders, and W. Daelemans, "A formal framework for evaluation of information extraction," *Online http://www. cnts. ua. ac. be/Publications/2004/DCD04*, 2004.

[15] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A comprehensive overview of large language models," 2024.

[16] R. Colomer, "The three types of artificial intelligence: Ani, agi, and asi — discovering the world of ai," *Medium*, 2023.

[17] M. Nalpas, "Understand llm sizes," *AI Explore*, 2023. bookmark_border.

[18] J. Mahapatra and U. Garain, "Impact of model size on fine-tuned llm performance in data-to-text generation: A state-of-the-art investigation," *arXiv preprint arXiv:2407.14088*, 2024.

[19] M. Chinta, "Large language models (llms) — top view," *CodeNx*, February 2024.

[20] S. Sayehban, M. Ehghaghi, and S. Siri, "How do i prep my data to train an llm?," July 2024.

[21] F. Xue, Y. Fu, W. Zhou, Z. Zheng, and Y. You, "To repeat or not to repeat: Insights from scaling llm under token-crisis," in *Advances in Neural Information Processing Systems* (A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, eds.), vol. 36, pp. 59304–59322, Curran Associates, Inc., 2023.

[22] H. Kashyap, "Here's everything you need to know about zero-shot learning," February 19 2024.

[23] "Supervised fine-tuning: How to choose the right llm," https://www.sama.com/blog/supervised-fine-tuning-how-to-choose-the-right-llm: :text=While

[24] J. Kwon, "Explanation: Supervised fine-tuning & reinforcement learning from human feedback," 2023. Retrieved from [https://medium.com/@joonbeomkwon/understanding-ai-training-supervised-fine-tuning-vs-reinforcement-learning-from-human-feedback-55d1b65317e0].

[25] R. Guan, K. L. Man, F. Chen, S. Yao, R. Hu, X. Zhu, J. Smith, E. G. Lim, and Y. Yue, "Findvehicle and vehiclefinder: a ner dataset for natural language-based vehicle retrieval and a keyword-based cross-modal vehicle retrieval system," *Multimedia Tools and Applications*, vol. 83, no. 8, pp. 24841–24874, 2024.

[26] J. Li, H. Li, D. Sun, J. Wang, W. Zhang, Z. Wang, and G. Pan, "Llms as bridges: Reformulating grounded multimodal named entity recognition," *arXiv preprint arXiv:2402.09989*, 2024.

[27] E. Mukans and G. Barzdins, "Riga at smm4h-2024 task 1: Enhancing ade discovery with gpt-4," in *Proceedings of The 9th Social Media Mining for Health Research and Applications (SMM4H 2024) Workshop and Shared Tasks*, pp. 23–27, 2024.

[28] H. Bouamor, J. Pino, and K. Bali, "Proceedings of the 2023 conference on empirical methods in natural language processing," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

[29] M. D. Ma, X. Wang, P.-N. Kung, P. J. Brantingham, N. Peng, and W. Wang, "Star: Boosting low-resource information extraction by structure-to-text data generation with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 18751–18759, 2024.

[30] Z. Wang, Z. Zhao, Z. Chen, P. Ren, M. de Rijke, and Z. Ren, "Generalizing few-shot named entity recognizers to unseen domains with type-related features," 2023.

[31] D. Segal and F. Coolidge, *Reliability*, pp. 1835–1836. 02 2018.

[32] T. K. Koo and M. Y. Li, "A guideline of selecting and reporting intraclass correlation coefficients for reliability research," *Journal of chiropractic medicine*, vol. 15, no. 2, pp. 155–163, 2016.

[33] Z. Bobbitt, "Intraclass correlation coefficient: Definition + example," March 2021. Posted on March 19, 2021.

[34] Y. Sheng, L. Zheng, B. Yuan, Z. Li, M. Ryabinin, B. Chen, P. Liang, C. Re, I. Stoica, and C. Zhang, "FlexGen: High-throughput generative inference of large language models with a single GPU," in *Proceedings of the 40th International Conference on Machine Learning* (A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, eds.), vol. 202 of *Proceedings of Machine Learning Research*, pp. 31094–31116, PMLR, 23–29 Jul 2023.

[35] E. F. Tjong Kim Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147, 2003.

[36] D. Wintaka, M. Bijaksana, and I. Asror, "Named-entity recognition on indonesian tweets using bidirectional lstm-crf," *Procedia Computer Science*, vol. 157, pp. 221–228, 01 2019.

[37] Z. Bobbitt, "How to calculate intraclass correlation coefficient in python," March 19 2021. Posted on March 19, 2021.

[38] R. Kundu, "F1 score in machine learning: Intro & calculation," December 16 2022. 8 min read.

[39] S. Team, "Limitations of language models in other languages," April 25 2024.

[40] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.