# Master Project

Laboratory : Swiss Data Science Center

Company : Ketl Sàrl

---

# Information Extraction in Official Documents using Large Language Models

---

Mathieu Desponds,
*supervised by*
Nicolas Casademont *(ketl)* Clément Lefebvre *(SDSC)* and Olivier Verscheure *(SDSC)*

March 1, 2024

**Abstract**

Ketl is an AI-driven company specializing in the automatic extraction of information from official documents to facilitate efficient document retrieval using filters. However, the company encounters the challenge of maintaining client document privacy, prohibiting document sharing between clients while needing data to train their models.

Traditionally, clients classified manually documents that was then used for training a classification and a named entity classification models. With the emergence of Large Language Models (LLM), new possibilities arose. LLMs demonstrate impressive document understanding, enabling effective retrieval of critical information without the need for training. However, Ketl refrains from sending client documents to external APIs like OpenAI. Hence, one objective of this study is to assess the feasibility of deploying a smaller, on-premise model.

This thesis investigates LLM usage for named entity retrieval, exploring various prompt techniques, few-shot selection methods, output restriction strategies, and fine-tuning techniques. Due to the absence of labeled official document data, initial focus is directed towards traditional NER tasks, later extending these findings to evaluate LLM performance for Ketl. The models evaluated include deep-learning models like Flair, SpaCy and Wikineural-multinlingual-ner for comparison with the LLMs like ChatGPT and MistralAI.

Key findings reveal that, in the traditional NER task, while ChatGPT initially outperformed the raw Mistral-7B model, fine-tuning enhanced Mistral-7B's performance above those of ChatGPT. In comparison with deep-learning models, Flair and SpaCy consistently surpasses LLMs. However, for official document named entity extraction, ChatGPT yields superior results, followed by a fine-tuned MistralAI-7B-Instruct-v0.2 model, and deep-learning models exhibiting only limited capabilities.

Furthermore, a study on LLM confidence assessment proposes training logistic regression on model logits before token generation for the desired output. Given the rapid advancement in the domain of LLM and the associated tools, the potential for integrating LLM models into Ketl's operations and ease the life of many secretary that need to classify documents manually is real.

# Contents

## Acknowledgement

I would like to thank everyone that helped me during my Master Thesis. In particular the Ketl team, Nico, James, Carlos and Florian that have welcomed me in the team and made my first job a good experience.

I also would like to thanks Clément and Olivier from the Swiss Data Science Center that supervised my thesis and helped me with its structure and guided me.

Finally, I would like to thanks all my family, friends and Aless that were present all this time even in the more difficult moments.

# Part I
# Named Entity Recognition with Large Language Models

## 1 Introduction

The objective of this first study is to compare the performance of various models on the task of Named Entity Recognition (NER), focusing specifically on the differences in performance between deep-learning NER models and Large Language Models (LLMs). The results will then be used for specific task of extracting information in document for Ketl.

LLMs have significantly impacted the field of Natural Language Processing (NLP). Despite challenges related to speed and precision, LLMs have consistently outperformed other models in many benchmarks. However, this success has not fully translated to NER. The primary reason is that while LLMs are trained for text generation, NER is a tagging task. Nevertheless, the fact that LLMs are behind in the task of NER may change in the coming months, as new papers are published and more advanced models are introduced frequently.

The question is then how should we use LLM to extract named entity and get better results than deep-learning architecture. In this work, we go through multiple techniques that includes the way of prompting, how to chose the demonstration presented in the prompt, how to handle the hallucination problem of LLM and how to fine-tune models with billions of parameters.

In addition, we want to evaluate the performance of small LLMs like MistralAI-7B (Jiang, Sablayrolles, Mensch, et al. 2023) that we can load locally and then fine-tune for a particular task. Finally, we want to compare these results with the performance of large SOTA LLMs like ChatGPT (Sakib Shahriar 2023).

While not being able to reach the performance of deep-learning methods on NER, we evaluated the benefits of the different techniques and how it increased the scores. One important finding is that the performance of a Mistral-7b model fine-tuned on 2000 samples beat ChatGPT on NER. This is motivating for Ketl as it might be true that having our own model on premise can beat OpenAI models and remove our dependencies toward them. Finally, with the fast development in the field of LLM, new findings might change these results and LLM might beat deep-learning models also in NER.

# 2 Literature review

In this section, we first take a look at the development of deep-learning model and how NER was performed with these models. Then, we provide an overview of the capabilities and challenges of LLMs. Finally, we examine the current solutions proposed for NER using LLMs.

## 2.1 Deep-learning architecture

With advancements in hardware technologies, new architectures requiring significant amounts of data and computing power can now be implemented with favorable results. In an attempt to emulate neurons in the brain, **Feed forward Neural Networks (FNNs)** were developed. The basic architecture comprises an input layers of neurons, an output layer, and a variable number of hidden layers that are fully connected. Data flows forward between these layers via matrix computation, and a *non-linear* function is applied at each neuron. The model learns through back-propagation, where derivatives are computed using a backward algorithm. With this architecture, it becomes possible to approximate any function on the input. However, the input is of a constant size, and more importantly, this type of model lacks memory or foresight and can not handle dependencies between the inputs.

**Recurrent Neural Networks (RNNs)** attempt to address the memory limitations of FNNs. Each token of the input is processed sequentially, and a hidden layer is computed before producing the output. The hidden layer serves as memory in the network, retaining information about the past. Additionally, the input can vary in size. However, RNNs encounter challenges with long-term memory and learning long-term dependencies. This is attributed to the vanishing gradient problem, where the gradients propagated backward are multiplied and eventually diminish, preventing the model from effectively learning long-range dependencies.

**Long short term memory (LSTM)** tries to solve the challenges of long term memory. While having a similar architecture as RNN, LSTM add a cell state that contain three gate that may add, modify or remove information. The forget gate takes the previous hidden state and the current input and decide how much to forget from previous hidden state. The input gate is responsible for adding the new important information that is not redundant to the cell state. Finally the output gate, takes the information from the cell state that is important for the current output. This can typically contain information about the syntax as the need for a noun after an adjective. This permits long term dependencies. However, there is still some issues as the training requires a lot more data and computation power than RNN. Moreover, the memory is still represented by a fixed size vector that limits the amount of memory it can have. Finally, the input is processes sequentially from start to end, therefore, the model is only capable of taking into account the past. To overcome the last issue, **Bi-directional LSTM** (BiLSTM) can be implemented where a LSTM is implemented form left to right and another LSTM is implemented from right to left. On top of these two LSTM, a function or another model can be used to merge the output of the LSTMs. **Linear-chain Conditional Random Field** (CRF) is popular on top of BiLSTM. Compared to a Logistic Regression, CRF is discriminative probabilistic model that takes the context into account and therefore is a good choice to model sequence of data. Another popular choice is **Multilayer Perceptron** (MLP) that is a FNN with at least three layers of fully connected neurons.

Another type of architecture that has been used in NLP is **Convolutional Neural Net-**

**work** (CNN). Traditionally, CNN has been largely utilized for image processing where filters are applied on the image until we reach the last layer that is typically a classification layer. The same architecture was then used for text where the input is the embeddings of the tokens that can form a matrix of embeddings. Filters are then applied on this matrix to find dependencies inside the input sequence until the last layer that performs classification.

One of the main downsides of previous models is the difficulty in parallelizing their training, as each output depends on the preceding one. (Vaswani et al. 2017) addresses this issue with **Transformers**, which are based on self-attention, also known as scaled dot-product attention. The principle of self-attention is that each token in the input sequence computes its attention - its compatibility - with all the other tokens, including itself. These computations can be performed in parallel, accelerating the training process compared to RNNs and LSTMs. The architecture proposed in the paper consists of an encoder that uses self-attention and a decoder that computes a masked self-attention that compute attention only with the previous token, preventing it from accessing future information.

Transformers have significantly impacted the field of deep-learning, particularly for sequence-to-sequence problems. This is primarily because they can capture long-range dependencies, can be trained in parallel, and offer interpretability through the attention mechanism. However, training such models requires substantial amounts of data due to the large number of parameters involved. With the development of hardware, training such model proved to be possible. BERT (Devlin et al. 2019) that stands for **Bidirectional Encoder Representations from Transformers** is a model based on transformers that replaces the encoder-decoder architecture by an encoder that conditions both on the left and the right context. This pre-trained model can then be fine-tuned on a specific task. The release of BERT permitted to get eleven SOTA on NLP tasks. Then, **Robustly Optimized BERT-Pretraining Approach** (RoBERTa) (Y. Liu et al. 2019) is an updated version of BERT trained on more data and with some changes in the pre-training process that improved the performance of BERT.

## 2.2 Named Entity Recognition with Deep-learning architecture

As (Keraghel, Morbieu, and Nadif 2024) explain, deep-learning architecture used for NER are usually based on three stages : data representation, context encoding and entity decoding.

**Data representation for NER** The task of this stage is to transform the sequence of words into vectors that can be fed into the encoder or/and decoder architecture. Many word embeddings have been developed, including Word2vec (Mikolov et al. 2013), GloVe (Pennington, Socher, and Manning 2014), fastText (Bojanowski et al. 2017) and ELMo (Peters et al. 2018) or a concatenation of these. These embeddings are trained on large datasets of text and aim to project words into a multi-dimensional space that captures semantic and syntactic similarities between them. Compared to one-hot encoding or TF-IDF, the aforementioned embeddings take context into account. This allows the same word to have different representations depending on the context in which it is used.

In addition to word embeddings, embeddings based on characters have been developed. The idea is to associate a vector with each character depending on the context and then merge these embeddings to obtain the embedding of the word using a merging algorithm. This

enables the model to produce embeddings for out-of-vocabulary words and spelling variations, which can serve as good indicators for finding named entities.

**Context encoding and entity decoding** The purpose of these two stages is to process the embeddings and then assign a label to each token in the sentence based. In (Collobert et al. 2011), the authors propose to use CNN that takes the input sentence as a whole and concatenate the embeddings so that we get a matrix. From this, we use filters to find features around the different token until we reach the last layer that has the dimension of the number of NER classes. (X. Li et al. 2017) went further by adding feedback link in the CNN so that the model can change its previous decision.

Others authors focused on RNN that can be easily implemented for NER as each token can be fed into the model and a label can be outputted for each input while taking the context into account with the hidden state. However, vanishing gradient and the capacity of only going in one direction made their use limited. To address these issues BiLSTM architecture alongside a CRF like Flair (Akbik, Blythe, and Vollgraf 2018) or alongside a MLP showed promising results. (Ma and Hovy 2016) showed that adding a CRF on top of BiLSTM proved to improve the performance on NER.

Finally, Transformers based model have been used widely either directly as named entity extractors or as embedding for other models like BiLSTM or CRF. (Labusch et al. 2019) showed that BERT as a tagger outperforms BiLSTM and CRF architectures. (J. Liu et al. 2020) used transformers to encode the features of the input and on top of it, they added a BiLSTM that encodes the context of the input and a Multi-headed attention mechanism to encode a wider context consisting of more than one or two sentences.

## 2.3 Large language Models

**Architecture of LLM** LLMs are based on the transformer architecture and are distinguished from traditional deep-learning models by their vast number of parameters, ranging from tens of billions to trillions. This has led to a new paradigm: while deep-learning models are typically pre-trained and then fine-tuned using labeled datasets, LLMs can perform various tasks without the need for fine-tuning. Instead, users provide a prompt to the LLM containing the task it will perform alongside demonstrations of input/output pairs. This is advantageous as it eliminates the need for big datasets of labeled data to achieve good performance, and the need to fine-tune models of billions of parameters which is costly and time-consuming. However, as it will be explained later, new techniques have been developed to further adapt LLMs to specific tasks by performing fine-tuning on only certain parameters and enhance their performance even further.

The Generative Pre-trained Transformers (GPT) family are well know Large Language models. While GPT-1 (Yenduri et al. 2018) can still be classified as a deep-learning model, it paved the way forward to GPT-2/3/4. In the paper, the authors introduce the concept of generative pre-training with transformers to get an wide understanding of the language.

GPT-2 (Radford et al. 2019) is the first model of the family that can be classified as a LLM as it introduces the concept of task conditioning where the user can prompt the model with the description of a task. Compared to GTP-1 that had 117M parameters, OpenAI released GPT-2 with 1.5B parameters. GPT-2 focused on zero-shot learning, giving no examples of

the task and took into account only the description. GPT-3 extends this with the capacity of using few-shots. Moreover, it had the capacity to generate not only human like texts but also code in many languages. GPT-3 was the base of InstructGPT (Ouyang et al. 2022) where Reinforcement Learning with Human Feedback (RLHF) was used to fine-tune GPT-3 so that the model is able to write like a human. While GPT-3 has 175B parameters, InstructGPT had only 1,3B and its output was preferred by humans. InstructGPT is the basis of ChatGPT and the main difference is the addition of safety in ChatGPT. Finally, OpenAI recently released GPT-4 that is a multi-modal model that can handle images, has a wider context window and its number of parameters has been estimated to 1.7 trillions parameters.

However, these models are really big and not open-source which makes us totally dependent on OpenAI and forces us to share our content with them. Moreover, the models can not be fine-tunes and we can not use the logits of the model as we need to use their API. Therefore, plenty of smaller open source models were released like Llama-2 (Touvron et al. 2023) or MistralAI (Jiang, Sablayrolles, Mensch, et al. 2023). When it was released in October 2023, the 7 billions parameters model of MistralAI was the best model of this size that uses Grouped Query Attention (GQA) which tries to find a good balance between the performance and the computational cost of the attention mechanism in the Transformers. Recently, MistralAI developed Mixstral7B8 (Jiang, Sablayrolles, Roux, et al. 2024) and Mistral Large. The former adopt a novel approach based on a mixture of experts. This architecture features layers of multiple experts instead of traditional dense FFN layers, selected by a pre-trained gate network router, resulting in significantly faster training and inference.

**Capabilities and challenges of LLMs** To harness LLMs' full potential, recent studies converge on several key principles. Increasing the number of demonstrations in the prompt is assumed beneficial, with the optimal threshold often dictated by the context window. Additionally, the order and diversity of demonstrations play crucial roles, with later examples carrying more weight and diversity aiding in maintaining generalization. Techniques like Chain-of-Thought, incorporating a sequence of Query-Rationale-Answer, have shown promise in enhancing LLM performance, particularly in complex reasoning tasks (Luo et al. 2024). Finally, llama.cpp let the possibility to incorporate grammars to the LLM. This tools forces the format of the LLM output and help to structure the response.

To further enhance performance, LLMs are often fine-tuned, albeit with challenges due to model size and memory requirements. Methods like Parameter Efficient Fine-tuning (PEFT)(L. Xu et al. 2023) and Low-rank Adaptation (LoRA)(Hu et al. 2021) aim to mitigate these challenges by reducing the number of trainable parameters while maintaining performance. The idea is decompose matrices into smaller ones and use these smaller matrices as the training parameters. With this technique, a 13B model, can train a LoRA that has only 228K parameters which represents 0.001% of its parameters. Finally, the LoRA is merged with the model weights to get the final model.

(Jha et al. 2023) also found that subsets of 1k-6k instruction fine-tuning samples can be sufficient to achieve good performance on both traditional NLP benchmarks and model-based evaluation. Moreover, they showed that mixing different datasets permits the generalization of the model and improves the performance. However, consensus on optimal fine-tuning strategies still need to be found.

Efforts to reduce model size and improve efficiency have led to research in quantization

techniques where the number of bit used for the representation of number is reduced efficiently. Innovations like GPTQ (Frantar et al. 2023), and bitsandbytes (Younes 2023) demonstrate that significant model size reductions are achievable with minimal performance trade-offs. Additionally, Quantization-Aware LoRA (QA-LoRA)(Y. Xu et al. 2023) introduces a novel approach of training a LoRA that can then be integrated on different quantized models without the need to merge the LoRA with the base model.

Moreover, focus has been made on how to accelerate the inference of the models. Starting from the fact that more than 50% of the tokens can accurately be generated with only the first layer at the inference process, a lot of computational cost can be saved. (Sun et al. 2024) propose a technique where the layers can be skipped during inference which led to a speedup of 11.2% with only marginal reduction of performance. On the other side, (Wu et al. 2023) simplified the LLM decoder layer by fusing data movement and element-wise operations to reduce the memory access frequency and lower system latency and improve the general memory and cache policy leading to 7x lower token latency and 27x higher throughput.

### 2.4 Named Entity Recognition with Large Language Models

Despite advancements, leveraging LLMs for NER tasks remains challenging, with supervised methods often outperforming LLM-based approaches. The disparity arises from the inherent differences between NER, a token-level tagging task, and LLMs, primarily used for text generation. Efforts to bridge this gap include approaches like sentence rewriting and sentence-to-sentence tasks, aiming to adapt LLMs to the NER context. (Wang et al. 2023) proposed GPT-NER, a method where the LLM rewrites the input sentence and wraps all the entities of a specified type with `@@##`. The number of calls to the LLM therefore depends on the number of entity types which will not be scalable. (Ashok and Lipton 2023) further explored transforming NER into a sentence-to-sentence task. Their method involves the LLM to extract all potential entities and then to determine whether each one is valid, followed by assigning entity types. Conversely, (P. Li et al. 2023) observed the difficulty in reinterpreting the NER task as a sentence-to-sentence task and opted to maintain the output structure while employing generative coding LLMs like Codex to outperform few-shot fine-tuned models.

The rapid evolution of Large Language Models presents both opportunities and challenges for academic tasks like NER and for applications like Ketl. With advancements in model architectures, fine-tuning techniques, quantization methods, and inference optimizations, there is a lot of potential for improving performance and efficiency. By leveraging the latest research findings and methodologies, Ketl can strive towards more effective information extraction from official documents while optimizing resource utilization.

## 3 Experimental Setup

### 3.1 Datasets

For our experiment, we focus on two well known NER datasets : Conll2003 and OntoNote5.

**Conll2003** The Conll20003 dataset (Tjong Kim Sang and De Meulder 2003) is an English corpus that is one of the most used dataset in the domain of Named Entity Recognition.

It comprises 14,000 training samples and 3,500 test samples, each containing sentence tokens paired with their respective tags. The tags belong to one of the categories `['PER', 'LOC', 'ORG', 'MISC']` which represents person entities, location entities, organisation entities and miscellaneous entities respectively. Person entities can be celebrities, historical figures, fictional characters or simple names. Organization entities are business, educational organization, broadcaster, sports organization, scientific organization, political organization, institute or government agency. Location entities are all countries, human-geographic territorials, geographical regions, areas in a single country or natural geographic objects. and finally, Miscellaneous entities are all events, languages or adjectives to describe things particular to a country.

The dataset includes mainly small informal sentences, sports results (football match scores, ski race results), and other examples containing numerous numerical values. The ultimate aim of the research centers on assessing Large Language Models for the extraction of named entities from official documents. Consequently, samples containing numerous numeric terms, duplicate entries, and those lacking verbs, as identified by `nltk` (Bird, Klein, and Loper 2009), were excluded from the corpus. This resulted in a training dataset of 7,577 samples and a test dataset of 1,588 samples.

It should be kept in mind that the dataset contains wrongly labeled examples as (Reiss et al. 2020) explains leading to difficulties in reaching perfect score.

**OntoNote5** The OntoNote5 dataset (Hovy et al. 2006) is another well known dataset for NER. It differs from Conll2003 as it contains 18 tags instead of 4. the tags are `['CARDINAL', 'ORDINAL', 'WORK_OF_ART', 'PERSON', 'LOC', 'DATE', 'PERCENT', 'PRODUCT', 'MONEY', 'FAC', 'TIME', 'ORG', 'QUANTITY', 'LANGUAGE', 'GPE', 'LAW', 'NORP', 'EVENT']`.

`CARDINAL` are numerals that do not fall under another type (e.g. 1, 100, twenty-nine). `ORDINAL` are words or expressions indicating order (e.g. first, 60th). `WORK_OF_ART` are titles of creative works like books, films or artistic work. `PERSON` are names of people, including fictional and real characters. `LOC` are geographical locations, both physical and political. `DATE` are temporal expressions indicating dates or periods, weekday, months, `PERCENT` are percentage values (e.g. 70%). `PRODUCT` are names of products or services. `MONEY` are monetary values, including currency symbols. `FAC` are named facilities, such as buildings, airports, or highways. `TIME` are temporal expressions indicating times of the day. `ORG` are names of organizations, institutions, or companies. `QUANTITY` are measurements or counts, including units (e.g. 10 grams, 1 litre) `LANGUAGE` are names of languages. `GPE` are geopolitical entities, such as countries, cities, or states. `LAW` are legal references, including laws and legal concepts. `NORP` are nationalities, religious group, or political groups. Can be adjective for nationality like "Canadian". And `EVENT` are named occurrences and social, political, cultural, or genera incidents.

The dataset comprises 60,000 training samples and 8,500 test samples. Each row includes the tokens of a sentence and the tags associated to these tokens. There exists longer well written sentences than in the Conll2003 and less errors in the dataset were found. In alignment with the project's objective to extract entities from documents, we take advantage of these long sentences and retained only sentences with a minimum of 30 tokens. The final dataset consists of 9,873 training samples and 1,403 test samples.

### 3.2 Deep-learning Models

We tested different language models that uses different architecture and are close to SOTA but with differences.

We first tested two version of `SpaCy3.0` (Honnibal and al. 2021). The first one is based on CNN and the second uses the transformer architecture based on `RoBERTa` (Y. Liu et al. 2019). This model is the one used by Ketl for extracting entities in their documents. We compare it with `Flair` (Akbik, Blythe, and Vollgraf 2018) that is a BiLSTM-CRF sequence tagger. Instead of using classical word embedding, Flair uses contextual string embedding. We also tested `Babelscape/wikineural-multilingual-ner` (Tedeschi et al. 2021) which is a 4 tag multilingual model based on `BERT` (Devlin et al. 2019). As Ketl has client who have documents in many languages - mainly French English and German -, it was interesting to test the performance of a multilingual model. The training dataset of the model is based on Wikipedia where knowledge based graph in addition to other NER model were used to increase the quality of the data.

`Flair` has a version for both 4 tags and 18 tags datasets. On the contrary, `SpaCy` is a 18 tags tagger while `Babelscape/wikineural-multilingual-ner` has only a version for 4 tags. Therefore, for the task using the Conll2003 dataset, the tags were remapped for SpaCy to facilitate a direct comparison with the other models.

### 3.3 LLMs Models

A substantial challenge of using LLMs is to be able to explain the task so that the model can understand it and produce the expected output. In order to do this, we need to clear on the task itself which is not easy for NER. Questions like *what constitutes a 'MISC' entity?* appears and the definition used by dataset may differ. Moreover, as annotators were humans, it may be possible that the definition was interpreted differently across the annotators. Another part of the challenge is that the LLM might have encountered various explanations during its training. Consequently, the challenge lay in customizing task descriptions to a particular dataset.

In this section, we examine all the techniques that were utilized to improve the performance of NER with LLMs. We begin by exploring different methods of prompting the LLM, then focus on the selection of the optimal few-shots. Following this, we discuss the implementation of a verifier, and conclude by addressing strategies for fine-tuning the LLMs.

#### Prompts

The structure of the prompt can have a significant impact on the performance of the LLM. Therefore, we experimented with two different prompts. The first one is a raw prompt where only a sentence is used to explain that we need to extract named entities and what is the format of the output. Then we created a second prompt where we added a definition of what named entities are and provided descriptions for all the tags.

#### Output format

When working with LLMs, before grammars were available, it was challenging to restrict the output. However, even with grammars, we can have big differences in performance between

different output format. Therefore, we tested different methods to format the results. All prompts used for the different techniques can be found in Appendix 1.

**GPT-NER** Initially, we sought to replicate the approach proposed in the GPT-NER paper (Wang et al. 2023), where they advocate for a generation-based approach where the input sentence is rewritten appending '@@' at the beginning and '##' at the end of each entity belonging to a specific tag. For instance, querying the LLM for person entities in the sentence "Washington is the president of the United States" is expected to yield the output `@@Washington## is the president of the United States`.

However, as explained in the literature review, this technique does scale well as it will require 18 calls for the OntoNote5 dataset and need to rewrite the whole input which can be costly. While parallelization of requests can alleviate some of these costs, it may not provide a complete solution.

**Direct Output of Named Entity Spans** To address the issues mentioned above, we explored an alternative technique by instructing the LLM to directly output a list containing all entities along with their corresponding tags. For the previous example, the expected output would be `[(Washington, PER), (United States, LOC)]`. This method requires only a single call to obtain all tags simultaneously.

However, potential downsides include the abundance of punctuation that contributes to an increased number of tokens to generate. Additionally, generating a Python list may be less efficient for the LLMs than generating well-formed sentences.

**Wrapping Named Entities with Tags** To test the aforementioned hypotheses that LLMs prefer generating well-formed sentences, an hybrid approach was implemented, aiming to retrieve all entities in one run while allowing the LLM to generate full sentences. This was achieved by instructing the LLM to rewrite the sentence and encapsulate each entity with specific HTML-like tags. For the previous example, the anticipated output would be `<PER>Washington</PER> is the president of the <LOC>United States</LOC>`.

**Outputting a JSON** A last technique was tested, that would output a JSON as answer. This is a known format that the LLM might understand better than a list. For the previous example, the output would be { `"PER" : ["Washington"], "LOC" : ["United States"], "MISC" : [], "ORG" : []` }.

**Few-shot extraction techniques**

Relying solely on the provided prompts often proves insufficient, as Large Language Models may struggle to generate accurate answers without an understanding of the specific task. To address this limitation, in-context learning by providing examples directly within the prompt was introduced. Based on the GPT-NER paper, three distinct few-shot techniques were implemented.

**Random retrieval** The first technique involves selecting random examples from the training dataset and presenting them as few-shots to the LLM. However, a drawback lies in the potential difference of context between these random examples and the actual sentence, hindering the effectiveness of the few-shots.

**Sentence embedding based retrieval** To mitigate the limitations of the random approach, the second technique suggests using as few-shots, the sentences in the dataset whose embeddings are closest to the embedding of the actual sentence. By aligning embeddings, the hope is to provide more relevant examples to the LLM.

**Token-level embedding based retrieval** In the third approach, instead of relying on sentence embeddings, the technique utilizes the embeddings of individual tokens in the sentence, obtained from `dslim/bert-base-NER` (Devlin et al. 2019) a BERT based model fine-tuned on NER. The goal is to identify, within the dataset, sentences containing tokens that closely resemble those in the actual sentence. By considering token-level embeddings, a finer level of specificity is achieved, with the final step involving the selection of $k$ sentences that contain tokens most similar to those in the actual sentence.

### Verifier

Addressing the issue of hallucinations and excessive labeling by LLMs, the authors of GPT-NER implemented a verifier mechanism in their study. In order to implement this, all the named entities extracted are examined, and for each named entity, we prompt the LLM with the sentence of interest and ask whether the named entity extracted has really this tag associated in the sentence.When the LLM acknowledges an error, the identified named entity is excluded from the list.

### LLMs fine-tuning

In the pursuit of increasing the model performance, two fine-tuning strategy were employed, using both the prompt technique of direct output and wrapper.

The initial fine-tuning methodology centered on succinct prompts with only a description sentence of the task, the input sentence and the output. Concurrently, a more expansive fine-tuning approach was implemented. This involved the formulation of more elaborated prompts including a task description. Additionally, between one and four few-shots were introduced, selected using the sentence embedding technique. For the training 2000 samples were randomly taken from the training set of the dataset. We used a learning rate of $2.5e-4$, the model was only trained in one epoch and the batch size was 4.

To optimize computational efficiency, the model was loaded from HuggingFace [1] utilizing 4-bit quantization. Then, Parameter-Efficient Fine-Tuning and the integration of an accelerator to facilitate fine-tuning were used in order to get the LoRa Adapter. The adapter was merged with the base model through the utilization of the Llama.cpp toolkit (Gerganov 2023). The fine-tuning prompt are available in the Appendix 2.

[1] Hugging Face is a company and an open-source community that develops and maintains various natural language processing (NLP) models and tools. The website can be found here

# 4 Results

## 4.1 Deep-learning Models

The four models were executed on the resulting test sets, and the multi-class weighted F1-scores that uses the number of instances per class as weights in the F1-score were reported.

The results, as shown in Table 1, highlight Flair's superiority in the Conll2003 task compared to other deep-learning models, achieving an F1-score of 0.87. Conversely, the other models failed to attain a similar score on this dataset. SpaCy faced challenges in remapping tags as it was not originally designed for 4-tag tagging. In contrast, while Flair was specifically trained on Conll2003, Wikineural-multilingual was trained on Wikipedia, where the sentence structures differ from those in the Conll2003 dataset. This discrepancy may partially account for the performance gap between the two models but it might not be true in the context of large document as for Ketl. Notably, Flair's performance may not generalize well to datasets it was not trained on.

On the Ontonote5 dataset, Flair and SpaCy yielded similar results. The transformer architecture outperformed the BiLSTM-CRF and CNN architectures. However, accuracy in named entity extraction is not the sole determinant for Ketl, especially when the scores are closely aligned. Therefore, the processing time for the 1406 samples in the test set was also taken into consideration. Notably, SpaCy-CNN exhibited rapid processing speed and could be a viable choice for production despite its slightly lower performance. Additionally, SpaCy-RoBERTa, while achieving a higher F1-score than Flair, also demonstrated faster processing. The disparity in processing time between Flair and SpaCy is likely attributed not solely to the model architecture, but also to implementation differences between SpaCy and HuggingFace. SpaCy is primarily designed for industrial use, whereas HuggingFace models are more geared towards research purposes.

In summary, Flair demonstrates strong performance on both datasets, but the SpaCy model with the transformer architecture outperforms it on the Ontonote5 dataset. Moreover, despite SpaCy-CNN not yielding the best results, it is by far the fastest model and this should be taken into account for Ketl, where processing speed is an important factor.

In the following sections we will compare the deep-learning models with the performance of LLMs with different settings.

| **Model** | **Architecture** | **Conll2003** | **OntoNote5** | **Runtime** |
|---|---|---|---|---|
| Flair | BiLSTM-CRF | 0.87 | 0.79 | 5 min |
| SpaCy | CNN | 0.57 (w/ mapping) | 0.75 | 0 min 16 sec |
| | RoBERTa | - | 0.82 | 2 min 45 sec |
| Wikineural multilingual | BERT | 0.65 | (0.09) | 59 sec |

**Table 1:** The multi-class weighted F1-scores on the 2 datasets for the deep-learning models. The runtime is for the resulting 1406 sample of the Ontonot5 test set.

## 4.2 LLM Models

All the experiments done with LLMs were computed using Mistral-7B on 3 times 100 samples. Bootstrapping was used to get a confidence interval that was computed with a Student. As

for the deep-learning models, we report the multi-class weighted F1-score where only perfect match between the named entity extracted by the LLM and the gold label was accepted.

**Qualitative analysis of the results** One of the challenges was to clarify what was the task, what is a named entity and align with the dataset requirements knowing the LLM has been trained on instances of NER. Here are examples of instances where the gold labels and the LLM predictions closely align but are not identical. These differences resulted in a low F1-score as perfect match is used, even though the model's responses were not incorrect.

**Gold :** `[('China', 'GPE'), ("Northeast Teacher 's University", 'ORG'), ('the Northeast Asia region', 'LOC'), ('the next twenty years', 'ORG')]`
**Predictions :** `[("China 's Northeast Teacher 's University", 'ORG'), ('Northeast Asia', 'GPE'), ('twenty years', 'DATE')]`

**Gold :** `[('Finnish', 'NORP'), ('the Tumen River Development', 'GPE'), ('Sweden', 'GPE'), ('Chinese', 'NORP')]`
**Predictions :** `[('Finnish government', 'ORG'), ('Tumen River', 'LOC'), ('Sweden', 'ORG'), ('Chinese', 'GPE')]`

What we can see is first that there still are errors in the gold labels as `'the next twenty years'` is classified as an organization. Moreover, one challenge is where to split entities : does `China 's Northeast Teacher 's University` contain one entity or should it be split in two entities ? Moreover, in the second example, is `Finnish` the named entity, or is it `Finnish government`? There is no final answer, but this non-alignment between the gold labels and the LLM output should be taken into account in the comparison of the models as a bad F1-score does not necessarily means bad performance.

**The impact on the way of choosing the few-shots** We tested the three few-shots methods on the Conll2003 dataset with different number of few-shots. Table 2 shows that the methods based on embeddings outperform the method with random choices. However, it is not clear whether the entity embedding is better than the sentence embedding as the few-shots method might depend of the prompt technique used. In GPT-NER, they reported that the entity embedding outperformed the sentence embedding and both were way better than the random choices. This difference might be due to the difference in the implementation of the entity embedding method or with the choice of the model used for entity embeddings.

Not surprisingly, as the number of few-shots increased, the random retrieval method caught up with the embeddings one and even at some point surpassed them. This can be explained as random retrieval will provide a broader view of the task, whereas the embeddings method will often retrieve very similar sentences[2]. This can result in a form of over-fitting where the model has lost its generalization power and lead to poorer results.

**The impact of the verifier** In our experimentation, we implemented the verifier in scenarios without few-shots, observing only on average an increase of 0.004 point in the F1-scores.

[2]This is even truer with the Conll2003 dataset that contains very similar sentences

| Output format | Few-shot technique random | entity | sentence |
|---|---|---|---|
| GPT-NER | - | 0.379 | 0.452 |
| Direct Output | 0.504 | **0.632** | 0.595 |
| Wrapper | 0.475 | 0.574 | **0.691** |

**Table 2:** All F1-scores when comparing performance of Mistral-7b with the different few-shots techniques on the Conll2003 dataset with 5 few-shots by prompt.

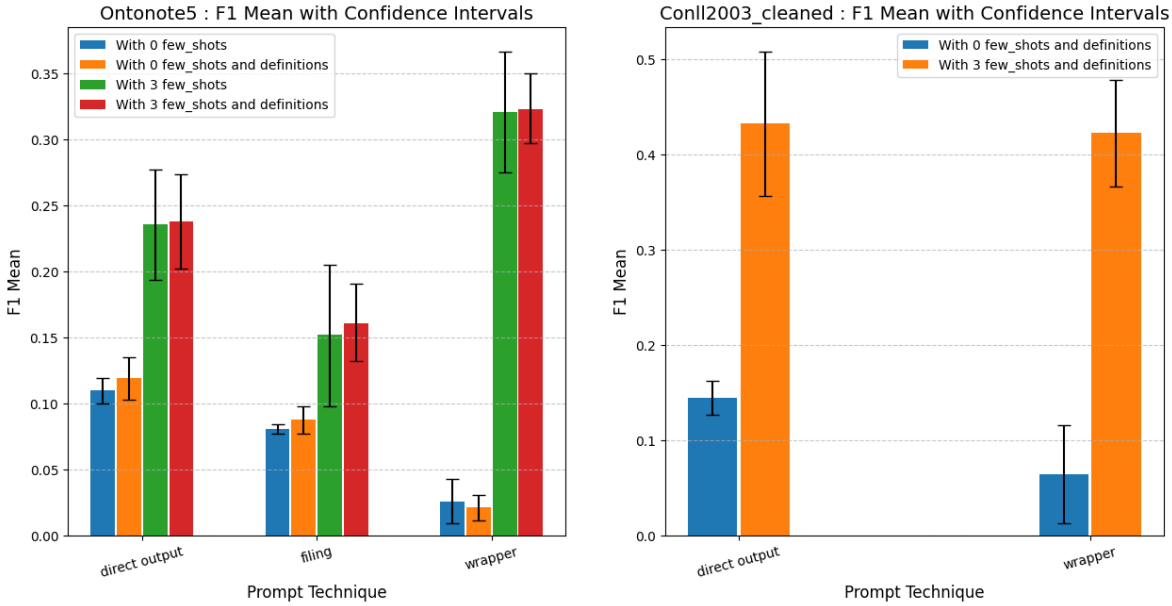


**Figure 1:** F1-score of a Mistral-7B on the two dataset with different prompt technique.

The authors of the GPT-NER paper also reported a modest improvement in results, approximately 0.005 of the F1-score. However, given that the improvement mirrored the findings in the paper and considering the additional computational resources required for verification, we opted not to delve further into this approach.

## Experimentation on the output format, the addition of definitions and the number of few-shots

In this subsection we use a Mistral-7B model and a fine-tuned version both quantized with on 8 bits which is almost indistinguishable from `float16`, and tested on both datasets. For the OntoNote5 dataset 3 out of 4 output format were used excluding GPT-NER as it would need 18 calls by samples which became costly. For the Conll2003 dataset focus was put on the direct output and the wrapper methods.

The results without fine-tuning can be found in Figure 1. The results with the model fine-tuned in Figure 2. The impact of few-shots is analysed in Figure 3.

**Main results** Our findings can be summarized as follows:

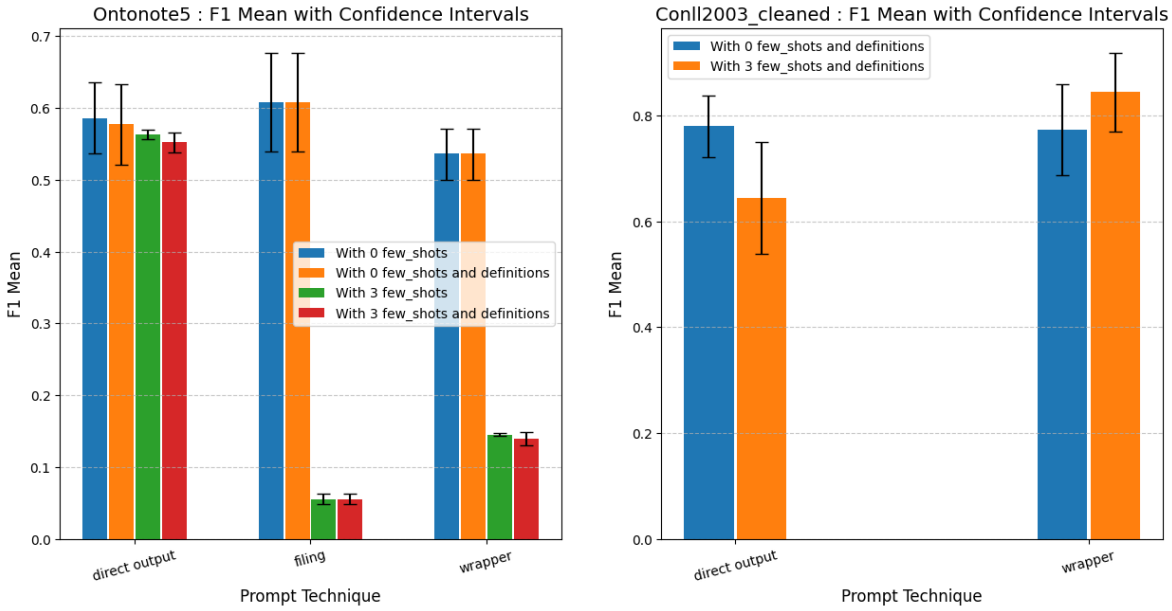


**Figure 2:** F1-score of a Mistral-7B fine-tuned on the different prompting method and the two dataset.

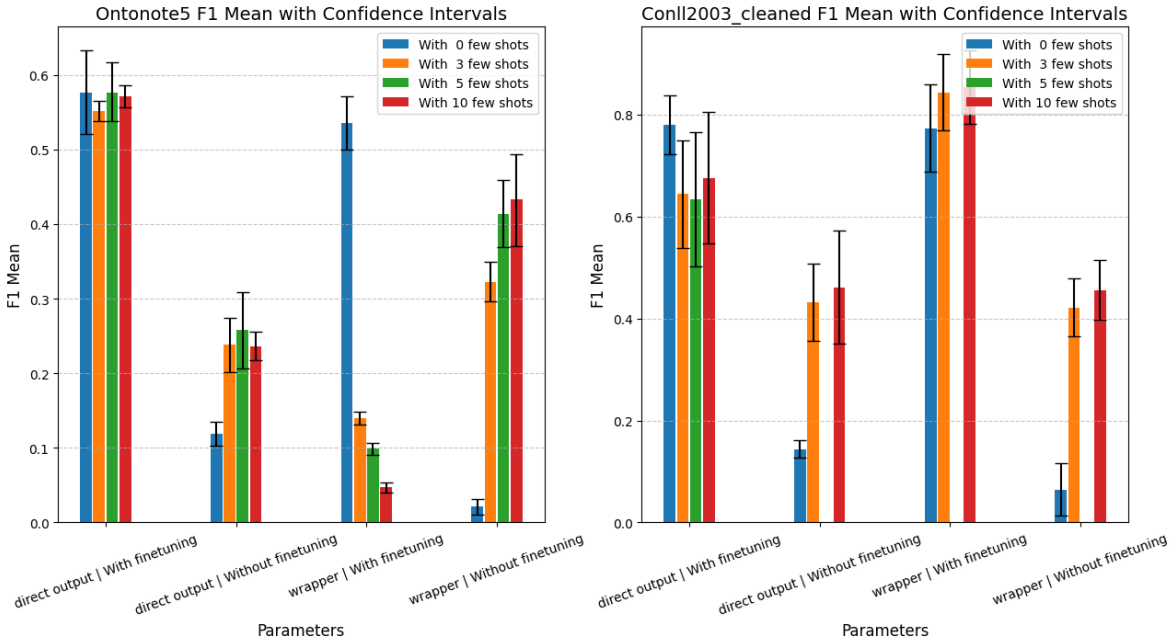


**Figure 3:** F1-score of a Mistral-7B fine-tuned on the different prompting method and the two dataset with different number of few-shots.

| | **OntoNote5** | **Conll2003** |
|---|---|---|
| Base line | 0.069 | 0.104 |
| Adding definitions | 0.073 (+0.004) | - |
| Adding fine-tuning | 0.576 (+0.507) | 0.777 (+0.673) |
| Adding few-shots | 0.256 (+0.184) | 0.427 (+0.323) |

**Table 3:** Impact of adding techniques on base line for the two datasets. Average F1-score is given with the increase compared to baseline in parenthesis.

Firstly, adding few-shots and fine-tuning improve drastically the performance of the model. In addition our experiments revealed that fine-tuning is more significant in improving model performance compared to the addition of more few-shot examples. Starting from a baseline with no shots and no fine-tuning, we observed an average increase in F1-score of 0.323 for Conll2003 and 0.184 for Ontonote5 by incorporating few-shots into the prompt. In contrast, fine-tuning the model on 2000 samples resulted in a more substantial improvement, with an average increase of 0.673 for Conll2003 and 0.507 for Ontonote5. Consequently, when feasible, we recommend using fine-tuning to achieve optimal performance.

Secondly, we also examined the impact of adding a definition of named entities and for all the tags. Surprisingly, without fine-tuning, this yielded only a marginal improvement of 0.004 in the F1-score on average whereas in the fine-tuned setup, the influence of adding definitions is inconclusive. This suggests that the model may already possess knowledge about the tag meanings and the task of named entity tagging. Therefore, adding more and more information in the prompt might not help the LLM and small concise prompt might be better, faster and cheaper.

Furthermore, the introduction of few-shot examples after the raw fine-tuning did not contribute positively to the performance. This could be attributed to the loss of generalization during fine-tuning, causing the model to struggle when faced with longer prompts that differ from those encountered during the fine-tuning process. Notably, while adding more few-shots in the direct output setting maintained consistent scores, the wrapper setting exhibited a drastic drop in performance. This discrepancy suggests a potential over-fitting issue in the wrapper setting, leading to the same loss of model generalization. One improvement might be the addition of classical Q&A samples in the fine-tuning process so that we keep the generalization power of the LLMs. This will be tested in the training process of Ketl.

Finally, fine-tuning by including a random number of few-shot was not able to surpass the raw fine-tuning technique. On the Conll2003 dataset and using the direct output technique we were only able to reach a F1-score of 0.685 without any few-shots and 0.682 with 5 few-shots compared to the score of 0.777 with the raw fine-tuning technique. This supposes that adding more information in the prompt does not necessarily mean better performance.

For a concise overview of our results, please refer to Table 3.

**Comparing with the Capabilities of ChatGPT** Our experiments with ChatGPT involved direct output and the wrapper technique, employing 0, 3, and 10 few-shots for both settings. Detailed results are presented in Table 4. Notably, the wrapper technique exhibited considerably higher performance than the direct output of a python list of tuples. This underscores the hypothesis that LLMs prefer generating natural language text over formatted

responses, especially for larger models trained on extensive datasets.

It is important to observe that while not surprisingly ChatGPT in superior in settings without fine-tuning, we found out that Mistral-7B's fine-tuned results surpassed those of ChatGPT. This observation holds significant implications for Ketl as it implies the potential for achieving high performance with a smaller model, freeing us from dependency on OpenAI.

However, one important drawback when using few-shots is the speed of ChatGPT compared to the deep-learning which is an important factor for production applications. In the setting with no-shots, ChatGPT is twice slower than SpaCy-RoBERTa and 20 times slower than SpaCy-CNN. When adding few-shots - the setting were ChatGPT get good performance - it is 13 times slower than SpaCy-RoBERTa and 138 (!) times slower than SpaCy-CNN. This factor should be taken into account for Ketl. A good balance between performance and computational cost needs to be found.

| **Technique used** | **Nb few-shots** | **Conll2003** | **OntoNote5** | **Runtime** |
|---|---|---|---|---|
| Direct output | 0 | 0.081 | 0 | 4 min 25 sec |
| | 3 | 0.224 | 0.077 | 30 min 56 sec |
| | 10 | 0.159 | 0.081 | 32 min 38 sec |
| Wrapper | 0 | 0.351 | 0.305 | 6 min 17 sec |
| | 3 | 0.607 | **0.515** | 36 min 49 sec |
| | 10 | **0.670** | 0.482 | 38 min 53 sec |
| Filing | 0 | - | 0.202 | 19 min 28 sec |

**Table 4:** F1-scores on the two datasets with different technique and the ChatGPT model. The runtime is the equivalent of running ChatGPT on 1406 samples so that we can compare the time with the deep-learning models

# 5 Conclusions

Evaluating LLMs on NER proves challenging due to the need for precision and restricting LLM output. Despite LLM outputs closely resembling gold entities, they do not result in perfect match, decreasing F1-scores. Fine-tuning the LLM or introducing few-shots significantly alleviated this challenge.

The conclusive results are encapsulated in Table 5, highlighting the comparison between different models. Flair continues to outperform LLMs in Named Entity Recognition. However, as LLM is a fast developing field, this might change in the following months. Moreover, we decided to compare the models based on perfect match which might not be the fairest way of comparing them. A metric based on whether all the important information is extracted from the sentence could be better. In conclusion, these results does not mean that LLM are doomed to perform worse on the extraction of official documents where the results would not need perfect match. We also showed that ChatGPT takes way more time than deep-learning approaches. The gap between the computing performance might be reduce by new research that try to reduce the size of the LLM and the speed of inference. However, it is an important factor to take into account.

Finally, the noteworthy achievement is the out-performance of ChatGPT by a fine-tuned Mistral-7B version with only 2000 samples. This accomplishment is motivating for the Ketl

project, aiming for model independence from OpenAI. This suggests that with fine-tuning a small model on a limited sample set, comparable and even better performance than ChatGPT can be achieved.

| Model | Technique used | Conll2003 | Ontonote5 |
|---|---|---|---|
| Mistral-7B | Direct output | 0.144 | 0.119 |
| Mistral-7B | Wrapper | 0.064 | 0.026 |
| Mistral-7B | Direct output-3-shots | 0.432 | 0.236 |
| Mistral-7B | Wrapper-3-shots | 0.422 | 0.321 |
| Mistral-7B fine-tuned | Direct output | **0.780** | 0.586 |
| Mistral-7B fine-tuned | Wrapper | 0.773 | 0.536 |
| Mistral-7B fine-tuned | Filing | - | **0.608** |
| Flair | - | **0.87** | **0.79** |
| Chat-GPT | Wrapper-3-shots | 0.670 | 0.515 |

**Table 5:** Summary of the F1-scores on the two datasets. The fine-tuning was performed on 2000 samples with no few-shots and only a basic description of the task.

**Future Work** Continuing this line of research holds promise. Experimenting with Codex to assess its effectiveness in transforming the task from a sequence-to-sequence problem to more of a coding task is intriguing. Evaluating newer models such as ChatGPT-4, Mixstral7B8, and Mistral Large would be important to assess their impact on LLM performance. However, ChatGPT-4 and Mistral Large might not be the most cost-effective solutions for Ketl.

A significant factor affecting LLM performance was our strict requirement for a perfect match between LLM output and gold labels, resulting in low F1-Score even when the LLM output was humanly acceptable. Relaxing this condition to allow partial matches could potentially enhance the LLM's performance. Additionally, leveraging grammars to construct Chains of Thought for the model could yield improvements worth exploring.

The key takeaway of this study is that incorporating few-shot learning and fine-tuning techniques can significantly enhance model performance. However, implementing these approaches for Ketl poses challenges. The documents are lengthy, limiting the ability to provide more than one document by prompt to the LLM. Moreover, Ketl's reluctance to share data across organizations means some clients have only a small number of documents available, make fine-tuning more difficult. In the following part, we delve deeper in how to apply these results to Ketl.

# Part II
# Using LLM at Ketl

## 1 LLM at Ketl

**Current Solution at Ketl** Ketl specializes in providing a document management solution, aiding companies in retrieving and categorizing information within their documents and accessing previously filed records.

A central aspect of Ketl's application involves document filing (see Figure 4, where users upload documents to the company's document manager. At this point, an AI system scans the document, and depending of it type, extract different information. For instance, in a payslip, the application would identify and retrieve company names, net and gross salary values, deductions, and more. Then, the user might confirm or change these information and finally will classify the document.

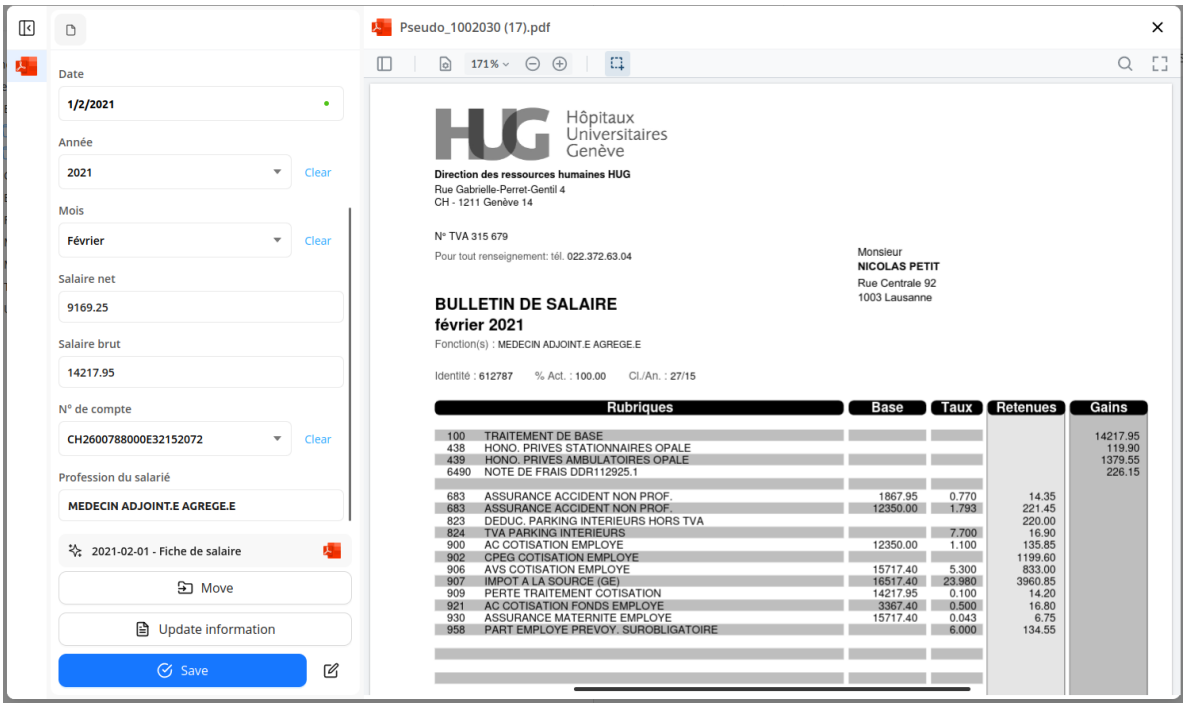


**Figure 4:** Filing application of Ketl on a pay slip. Fields extracted by the AI on the left and the document on the right.

Another integral aspect of Ketl's application is its PDF-split tool (see Figure 5), which handles lengthy documents comprising all the emails received throughout the day, scanned in a single instance. This tool employs AI to identify appropriate points for document segmentation, facilitating document splitting. Upon client confirmation, the tool proceeds to execute the splits.

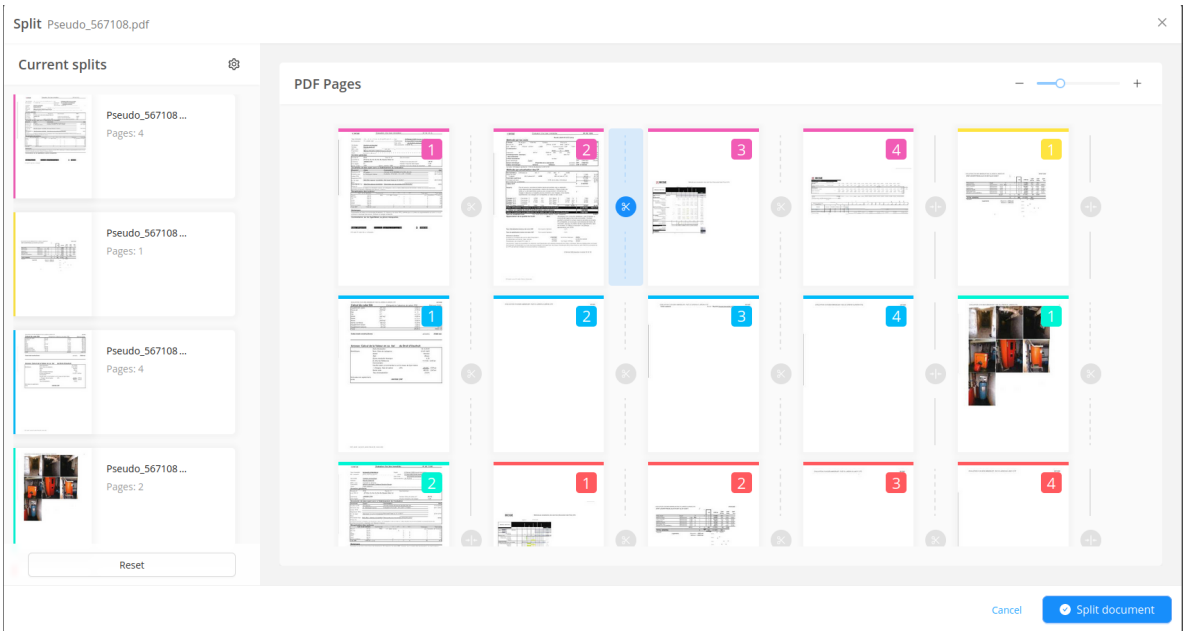


**Figure 5:** PDF-Split of Ketl. On the left the current split document and on the right the full document with the split inferred by the AI. The user can still remove or add more splits.

Ketl employs two distinct machine learning methods for information extraction. The first method employs a classification model that analyzes the document text to determine its type from a predefined list of possibilities or identify the client based on the clients stored in the database. The second method utilizes a named entity recognition model on the document, then uses a second classifier specific for all field on the extracted named entities as date, gross salary, or phone number. Both methods use the documents previously classified by users as dataset. The first method performs well even when the training data is small and there is a large number of possible classes. For example, a classification model for document type would need around 10 samples for each class to get 80% of accuracy. However, it necessitates predefined classes and therefore it is not able to extract specific fields like salaries. On the other hand, the second method gave results only when the training data was large. Empirically, to get 80% of accuracy, we need a little bit more than 100 document that contains this field.

**Challenges Faced by Ketl** Ketl encounters several challenges to deliver a robust document management solution.

First and foremost, the company aim for developing AI models with good efficiency and accuracy. The ultimate goal is to minimize human intervention, ensuring that the propositions generated by the AI require little to no adjustments. This requires the creation of highly accurate models.

To improve efficiency, Ketl could train their models using free to use available data. However, no publicly available data of official documents suitable for training exists. Moreover, Ketl prioritizes client data security and exclude any share of data across different companies. Consequently, during the initial phase of on boarding, companies will find themselves manually classifying documents, a situation Ketl aims to mitigate. This phase is even longer for smaller companies that add a limited number of documents to the application.

Adding to the complexity, each company possesses unique needs, with varying document types and information extraction requirements. Companies may introduce new document types or modify their information extraction preferences over time. Adapting to these diverse and evolving needs becomes a crucial challenge for the AI solution. Always training new models can become costly.

As clients classify many documents, Ketl aim for rapid document classification without compromising accuracy. Fast and precise extraction is of primary importance.

Equally important is the security of the models. Ketl prioritizes safeguarding sensitive information present in documents. To keep data integrity and privacy, Ketl aims to establish proprietary models rather than relying on remote APIs. This ensures that all information processed by the models remains within the Ketl infrastructure, protecting it from external threats and unauthorized access.

**Unlocking Potential: The Impact of LLM for Ketl** Ketl plan to use Large Language Models to improve the document management solution, recognizing the advantages they can bring to the table. Quick tests supposed good results with LLMs at extracting information from official documents that would minimize manual intervention.

One of the key benefits of integrating LLM is its immediate efficiency. The solution proposed by Ketl would be operational with high accuracy from day one. Unlike traditional approaches that necessitate pre-training, LLM present the capacity to recognize and extract information without the need for prior training. This would accelerate the on-boarding process and liberate the user from classifying many documents manually.

Flexibility is another advantage brought by LLM. In instances where a new document type or additional fields need to be extracted, Ketl can query the LLM for the specific information without the need of training models. This adaptability ensures that Ketl's document management solution remains agile and responsive to client needs.

While LLM holds great promise for Ketl, certain challenges need careful examination.

**Forthcoming Challenges** Data security remains an important concern for Ketl. The company is committed to preserving client confidentiality and ensuring that sensitive information does not leave the secure infrastructure. Consequently, Ketl wants to utilize LLM without relying on external services such as Microsoft or ChatGPT.

Furthermore, the size of the model poses a challenge. Balancing accuracy with model size is important to be compatible with low-resource infrastructure. Ketl aims to find a compromise, where the model's efficiency is maintained without needing a lot of resources.

Another limitation revolves around the context window of LLM as document can be of hundred pages. Even if new models tends to have even larger context window which might help in the future, we still need to find a way of putting the most important part of the document in the prompt. In this context, adding few-shots is not possible, at least for the moment. Additionally, the process of sending the entire document at each LLM call add processing time.

In conclusion, while LLMs offer potential advantages for Ketl due to their immediate efficiency, they also present challenges. Ketl aims for fast inference and prefers small-scale LLMs on-premises to conserve GPU resources and reduce dependence on external entities.

The question is then to see if small LLMs can perform quickly and with the same level of performance as models like ChatGPT.

# 2 Experimental setup

## 2.1 Implementation of the LLM Solution at Ketl

Ketl has implemented a systematic approach utilizing a tree structure for efficient information retrieval from documents. The process begins by querying the document type, initiating a cascade of subsequent inquiries tailored to the identified document type. In the foundational method, Ketl solicits information through a multi-step procedure:

1. **Document Type Inquiry:** The system prompts the user to specify the type of document from a predefined list, retrieved from the graph structure. This initial step serves as a crucial anchor for subsequent queries.

2. **Field Extraction Inquiry:** Based on the determined document type, Ketl proceeds to inquire about specific fields relevant to that document type. For instance, fields such as `'Date', 'AVS number', 'Salary'` may be requested, tailoring the information retrieval process to the unique characteristics of each document type.

3. **Object-Specific Information Retrieval:** Ketl also seek for detailed information from specific objects identified within the document, such as `'Tenant' 'Building'` and `'Client'`. To ensure accurate matching with existing database records, the retrieved information is then compared with stored objects using a string-based comparison method.

Despite the effectiveness of this implementation, challenges arise due to the need for multiple LLM calls, each requiring the transmission of the entire document. This drawback prompts ongoing exploration within Ketl to optimize the process while preserving the accuracy and integrity of the information retrieved from the documents.

## 2.2 Evaluation of performance

We want to evaluate how the LLM perform in comparison with traditional models on the extraction of entity in official document.

**Pseudonymized document** One of the Ketl client shared 85 pseudonymized documents that we can use for testing. There are 25 types of document and in each type of document there are different fields to extract. In total, 434 fields need to be extracted.

As explained, the extraction is based on a tree and therefore the extraction of certain fields depend on the precedent. On each document the type of document and the client are always extracted and then a third call is made to extract field specific to a certain document. One problem that can happen is that the type of document is missing in the LLM answer or is wrong and therefore all the other fields of the document are not extracted. Moreover, the client matching is based on the information retrieve from the LLMs (name, surname, type of

entity etc.) Therefore, the LLM might be able to catch some of the field while still not being able to have enough information to match the entity in the database. To be fair, the accuracy if based only on the information extracted.

We report two scores : first, the accuracy over all the fields extracted - the number of fields extracted and correct over the number of fields extracted - and secondly the average over the documents of the accuracy of the fields extracted by document - the number of fields extracted and correct in this document over the number of fields extracted in this document. The second is used to overcome the issue that would be that for documents with a small number of fields we are able to extract all the fields correctly while for document with a larger number of fields the extracted information is incorrect.

The performance evaluation of Large Language Models on Ketl involved assessing ChatGPT and Mistral-7B-Instruct-v0.2. Leveraging our previous work on Academic Named Entity Recognition, we experimented with various techniques, including `filling`, `direct output`, and `wrapper` methods on Ketl. However, the `wrapper` technique showed to be limited. While with other technique, the LLM can deduce the information to extract without needing to have it to be explicitly written in the document, the `wrapper` needs to find the information as text in the document. Therefore, it was not able to find the document type as it is only rarely written in the document and then failed completely as many fields depends on the finding of the document type. Additionally, we introduced a novel approach where ChatGPT first summarizes the document and then outputs the result in JSON format. The idea behind this is to permit the LLM to reflect on the document before giving its response. Finally, we tested the filling method on MistralAI and conducted fine-tuning experiments on a Mistral-7B-Instruct-v0.2 model using two distinct techniques.

In order to construct the dataset for the fine-tuning, we classified the 85 documents manually. From this we deducted the output that was expected for all the techniques. In the initial fine-tuning method, we appended a JSON containing the manually extracted information to the base prompt used by Ketl. The second, more advanced method involved integrating a Chain of Thought mechanism. For document type determination, the model's final response initiates with a deliberative statement, such as "after some thought, I believe the document type is [document type]," followed by a validation process to ascertain if the document type matches any in the predefined list. The validation process consists of a first sentence where the model assess whether the document type extracted is a possible one. Then the model either outputs the JSON with the correct value or suggests the closest alternative document type that is a possible document type. A similar approach is adopted for other fields, where the model prefaces its response with an acknowledgment of having comprehensively processed the document. The used grammars can be found in the Appendix 3.

To maintain model generalization, we also augmented the training data with samples from other datasets. The final dataset comprises one-third from Ketl, one-third from the Quora ( n.d.) dataset, and the remaining one-third from the ChatAlpaca (Bian et al. 2023) dataset.

**Comparing with deep-learning Models** Drawing a fair comparison between deep-learning models and Large Language Models poses a challenge. The conventional approach would involve utilizing a deep-learning named entity tagger like Flair or SpaCy to extract all named entities within a document, followed by constructing a multi-class classifier to assign labels to each entity. Regrettably, the scarcity of data prevents us to train such a model.

| Model | Prompt technique | Accuracy over fields | Accuracy over documents |
|---|---|---|---|
| Chat-GPT-3.5 | Filing | **0.745** | 0.714 |
| | Filing with summary | 0.526 | 0.480 |
| | Wrapper | - | - |
| | Direct output | 0.511 | 0.493 |
| Mistral-7B-Instruct v0.2 | Filing | 0.371 | 0.275 |
| Mistral-7B-Instruct v0.2 Fine-tuned-raw | Filing | 0.463 | 0.405 |
| Mistral-7B-Instruct v0.2 Fine-tuned-CoT | Filing | **0.467** | 0.394 |

**Table 6:** Accuracy on the extracted fields in the documents for the LLM models with different techniques.

| Model | Accuracy over segmented document | Accuracy over full document |
|---|---|---|
| SpaCy-CNN | 0.281 | **0.291** |
| SpaCy-RoBERTa | 0.320 | 0.178 |
| Flair | **0.380** | 0.114 |
| Wikineural-multilingual | 0.289 | 0.181 |

**Table 7:** Upper bound over the accuracies of deep-learning model on ketl documents.

Nevertheless, to establish an upper bound on deep-learning models' capabilities, we can assess whether they successfully retrieves the desired information from a document. To achieve this, we executed Flair 18 tags and SpaCy-transformers also 18 tags on documents in two distinct manners. Initially, the model was tasked with tagging the entire document, then subsequently the document was segmented at each '
n', and the model asked to tag each segment individually and then merge the extracted Named Entities. Then, we can check whether the gold label is part of the named entities extracted with the model. We even accepted when the gold label was a substring of one of the entity extracted. This allowed us to determine a large upper bound over the accuracy of well-retrieved fields. Notably, labels such as 'Document type' and 'client', more aptly extracted via a classification model than an entity extractor, were excluded in the computation of the accuracy.

# 3 Results

Table 6 presents the performance of the LLMs with different prompt techniques on Ketl and Table 7 present an upper bound on the performance of deep-learning models.

From the results with LLMs, we can see that the selection of prompt techniques has a substantial impact on the performance within the Ketl framework, even when complemented by the integration of grammars to constrain LLM responses. Surprisingly, incorporating a document summarization step fails to yield any notable improvements.

Moreover, MistralAI even with fine-tuning exhibits inferior performance compared to ChatGPT, particularly struggling with accurately identifying document types and occasionally misattributing clients as general companies instead of persons. The latter make the client matching fail.

Fine-tuning on 52 documents with the basic technique results in an improvement of nearly 0.1 in accuracy. It is conceivable that augmenting the dataset with additional examples showcasing greater diversity could further enhance the model performance. Interestingly, the alternative fine-tuning method fails to yield significantly better scores than the standard approach. However, expanding the sample size of Ketl data might render this method a viable improvement avenue, as the initial method may risk compromising model generalization.

**Comparing with deep-learning Models** The obtained accuracies were maximized with SpaCy-CNN when the document was not segmented and with Flair when it was segmented with score of 0.291 and 0.380 respectively, already lower than the scores achieved with MistralAI and considerably inferior to ChatGPT's performance even if it is a massive upper bound. It is crucial to emphasize that these scores solely reflect the models' capacity to extract entities rather than its proficiency in labeling named entities into their respective classes. Thus, the actual capabilities of these models fall below these scores.

One interesting observation is that the fastest model with the oldest architecture, not based on transformers, outperforms the transformer-based model when extracting named entities from the full document. This suggests that the CNN architecture, even when processing long inputs, effectively filters crucial information for extraction, while transformer architectures face greater challenges in this regard. Furthermore, the Wikineural-multilingual model, although significantly lagging in the academic NER task, demonstrates a degree of catching up with other models when tasked with larger documents.

Consequently, we conclude that while Flair and SpaCy excels in labeling named entities within specific academic contexts, its performance in applied fields pales in comparison to LLMs, which demonstrate superior adaptability to new conditions. The need for specific labels tailored for official document and that can depend on the client make the use of pretrained models difficult. However,

# 4 Conclusion and future directions for Ketl

Using the capabilities of Large Language Models for Ketl can be a significant step forward. As LLMs continue to evolve, their performance will undoubtedly improve. However, even with models like ChatGPT, we are already approaching highly promising results.

Emerging techniques for fine-tuning LLMs offer promising avenues to tailor their outputs. New prompting methods, such as those being developed with grammars, aim to constrain LLM generation. Although tools like Llama.cpp provide valuable assistance in this regard, they may still lack certain features, such as the ability to restrict outputs to a predefined long list of possibilities. Nonetheless, given the rapid pace of development in this field, such features could become available in the near future.

Moreover, advancements in quantization techniques occur frequently, enhancing LLM performance while reducing memory requirements. For Ketl, the prospect of deploying LLMs on smaller GPUs with high performance is particularly attractive.

Another challenge lies in the speed of inference, especially with tools like Llama.cpp, which can be relatively slow. However, new libraries like vLLM, despite their drawbacks, aim to expedite inference processes. This area of research holds significant promise for substantial improvements in the near future.

Furthermore, there remains no consensus on optimal LLM usage and fine-tuning strategies. Recent research has explored novel approaches, such as negative sampling and leveraging diverse datasets to maintain model generalization during fine-tuning.

Considering these potential advancements surrounding Large Language Models, Ketl stands to benefit from numerous forthcoming improvements in performance, enabling more efficient extraction of information from official documents at a decreasing cost.

# Part III
# Assessing the Confidence of Large Language Model Outputs

## 1 Introduction

In the context of the Ketl application, our aim is to extract diverse information from documents and present it to users for quick confirmation and document classification. However, we aim to avoid presenting information with low confidence and instead prefer informing the user that they need to locate the missing information themselves. To achieve this, it is crucial to assess the confidence of LLM in its responses. To evaluate the feasibility of providing confidence estimates, we began with the Conll2003 dataset and explored methods of assigning confidence to Named Entities extracted from sentences. We used Mistral-7b-v0.1 with llama.cpp to retrieve the logits.

## 2 Literature review

Formally, we seek a calibrated confidence level from the model. As described by (Portillo Wightman, Delucia, and Dredze 2023), confidence is considered calibrated if "the probability that the predicted label $\hat{Y}$ is equal to the correct label $Y$ for input $X$ should be equal to the model's predicted confidence".

In traditional Machine Learning, particularly in multi-class classification tasks, confidence assessment often involves evaluating the logits for different labels and applying a SoftMax function to derive probabilities for each class.

However, for LLMs it might not be the best solution. There are two primary methods to obtain confidence from an LLM. Firstly, we can directly inquire about the model's confidence in its responses. However, it's widely acknowledged that LLMs often exhibit overconfidence, with responses typically accompanied by confidence levels exceeding 80%, and this confidence is not well-calibrated, as observed by researchers (Guo et al. 2017; Mielke et al. 2022). Nevertheless, recent advancements, such as GPT-4 (OpenAI 2023), present intriguing prospects for direct self-confidence estimation. Alternatively, methods proposed by (Portillo Wightman, Delucia, and Dredze 2023) involve prompting the LLM multiple times and analyzing the variance of responses to gauge confidence. While effective, these methods come with a significant computational cost due to the need for multiple prompts and increased processing time.

Conversely, examining the logits of the LLM has not proven to be a viable solution. As outlined by (Xiong et al. 2023) logits present two key issues: as stated above, they often exhibit overconfidence and primarily reflect the model's uncertainty about the next token, rather than assessing the trustworthiness of specific assertions, which is more aligned with human-like responses. Moreover, it has been demonstrated that the confidence indicated by the model may not be well-calibrated with the actual probability of correctness.

# 3 Is there still a way to have a calibrated confidence for Named Entity Extraction ?

Given its significance for Ketl, we sought to validate the hypothesis that obtaining calibrated confidence with Large Language Models is hard. To this end, we experimented with the two methods elucidated above on the Named Entity Recognition task namely directly asking the LLM and using the logits.

### Asking LLM to Provide Direct Confidence on its Outputs

We conducted experiments to assess the ability of the LLMs to express its confidence in its outputs. In essence, after the tagger returned its results, we queried the LLM regarding its certainty that each extracted entity's tag was correct. This involved requesting the LLM to furnish a JSON containing the named entities as keys and one of the confidence levels ('low', 'medium-low', 'medium', 'medium-high', 'high') as values.

As depicted in Figure 6, the LLM encountered challenges in assessing its confidence, and the confidence levels were found to be poorly calibrated. The histogram a calibrated confidence would have would be that for confidence level 'high' we have 100% or right answer, for 'medium-high' around 75%, 'medium' 50%, 'medium-low' 25% and finally for 'low', only incorrect answers. However, in the histogram of the figure, the accuracy for labels categorized as 'medium-high was around 50% instead of 75% and for 'medium' the accuracy is close to 85% instead of 50% . The confidence provided by the LLM itself is not well calibrated and therefore might not be a viable solution for Ketl.

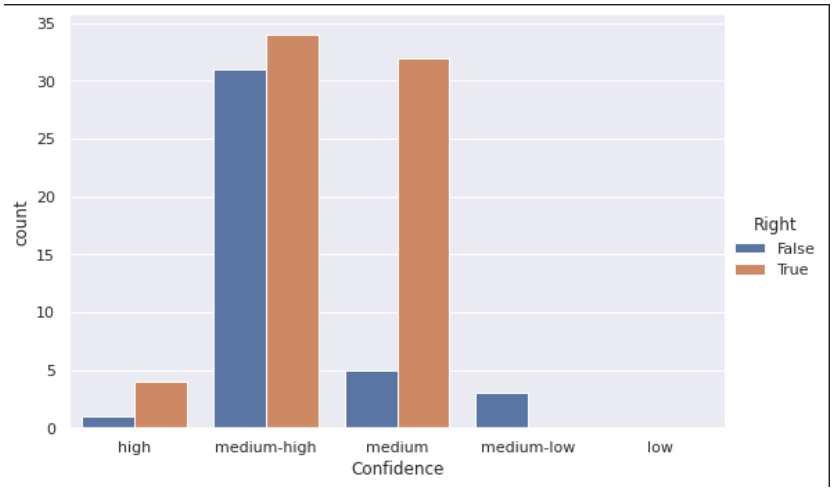


**Figure 6:** Number of time the LLM provide different level of confidence knowing the answer was incorrect (blue) or correct (orange).

### Examining Logits

While not employed as probabilities, looking at logits could still offer insights into result confidence. We conducted experiments by first instructing the LLM to generate a list of the named entities present in the input - the prompt used can be found in the Appendix 1 - and a

second call is made to associate a tag along with the entities - the prompt can be found in the Appendix 1. The logits are then analysed just before the model generates each entity's tag. The challenge lies in mapping the logits' values to the probability of the model's confidence in its response.

Since we are using $top_p = 1$, which is equivalent to $top_k = 1$, the LLM decoder employs greedy decoding, meaning it outputs tokens with the maximum probability at each step. This is significant because if we were to use a more complex decoder with $top_p < 1$, we couldn't directly examine the logits just before tag generation, as the final output might change in the future. We still need to ensure that labels do not share the same starting token to maintain a bijection between the labels and the starting token. If these conditions are fulfilled, we can try to use the logits to get the model's

**Using Tagger for Confidence Evaluation** We attempted to implement the aforementioned concept, aiming to allow the LLM to tag named entities and then comparing the logits at that moment. This approach yielded an F1-score of 0.622 on the NER task. Post-execution, we collected the logits just before the LLM outputted the tags and applied a SoftMax function to derive confidence values. However, as depicted in Figure 7, no clear delineation between correct and incorrect predictions emerged. The histogram that we would like to see only high "probabilities" when the output was right and only low "probabilities" when the answer was wrong. This suggests that utilizing logits may not effectively assess the model's confidence.

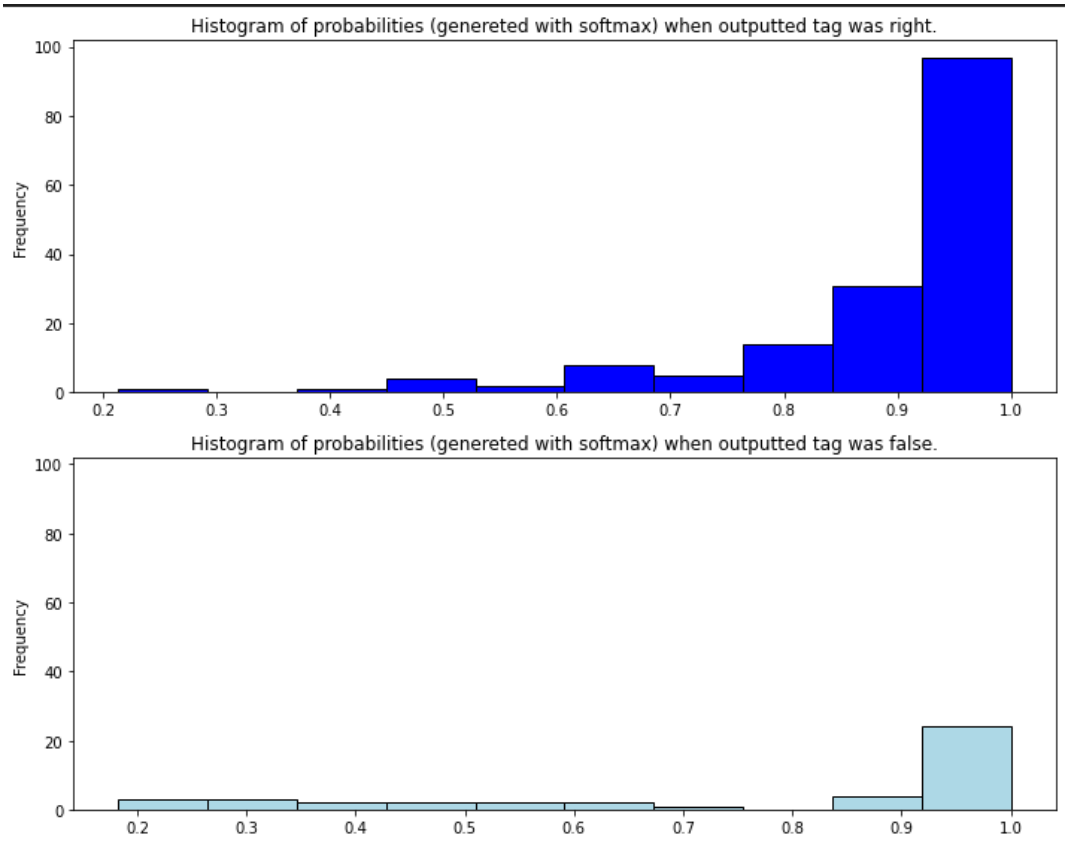


**Figure 7:** "Probability" computed with a SoftMax on the different labels' logits of the tag outputted by the LLM. The histogram in dark-blue is when the outputted tag was correct, in light-blue when it was wrong.

.

**Using Logistic Regression on the Logits** Our approach involved training a logistic regression model to address the question: *"Based on the logits for the five*[3] *tags, the maximum and minimum of the logits, is the tag you outputted previously {tag} the correct prediction?"*. This enabled us to derive a probability from the model's response, which served as an indication of confidence in the results.

With a dataset comprising only 206 named entities extracted from 100 sentences, we partitioned the data into training and testing sets (20% test). We trained a Logistic regression with the seven parameters and the results proved intriguing. As illustrated in Figure 8, although the data split was not perfect, the results were promising for evaluating model confidence. Setting a threshold at 0.5, meaning that below this value we are not confident about the result of the LLM and if it is above we trust the LLM, yielded an accuracy of 0.92. This indicated that the logistic regression, trained on 160 samples, effectively calibrated confidence based on the logits of the relevant tokens. In fact, when we have probability of the 'True' class close to 1, we have 100% of 'True' samples, we this probability is close 0.5

[3]The five tags are composed of the 4 classical tags 'PER', 'ORG', 'LOC' and 'MISC' and a 'None' tag was added when the model extracted the named entity in the first run but then realized it was not a named entity

we have half of the samples that are 'True and finally when the probability is close to one we have almost no 'True' samples.

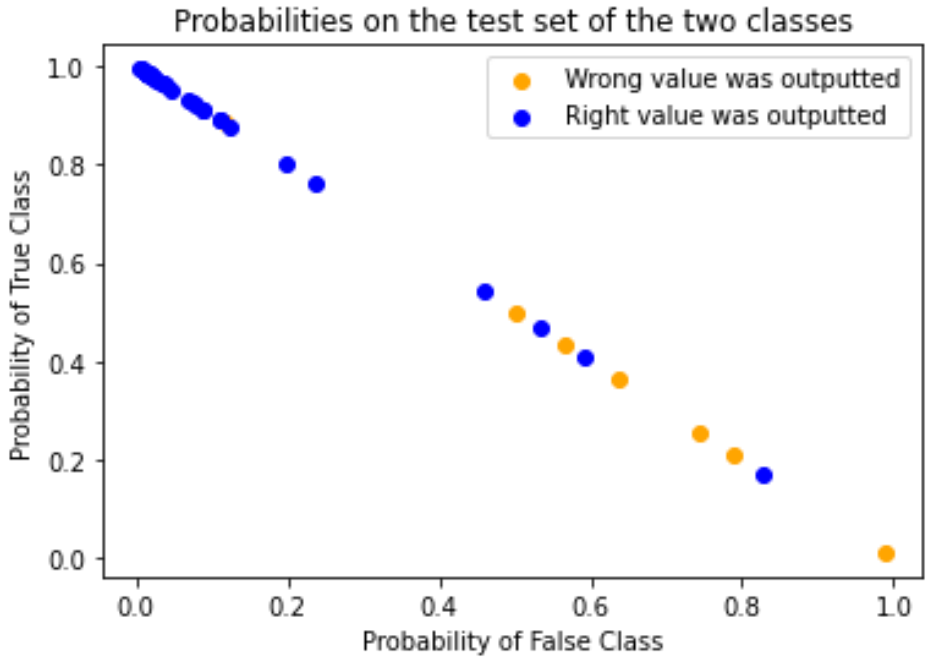


**Figure 8:** Output value of the logistic regression answering the question : *is the label outputted by the LLM correct* on the test set.

# 4 Conclusion

Evaluating confidence of LLMs have shown to be an interesting question because the use of logits is not as easy as with previous machine learning models. We found that directly querying MistralAI for its confidence or employing logits to derive probabilities did not yield well-calibrated confidence scores. However, through training a logistic regression model on 160 labels, we obtained promising outcomes, achieving calibrated confidence estimates that could aid in evaluating model confidence.

This might be interesting for Ketl to train such small models on their data. The idea would be to extract all the interesting named entities present in the document and then ask for a label on each of them. Based on the logits of the LLM at this moment, we could retrieve the logits, use the logistic regression based on the already seen data and give a notion of confidence to the user on the answer of the LLM. One challenges that would face Ketl, is that the number of classes depends on the client and can reach more than 40 and results in more challenges to have data to train these models.

However, recent and future developments in the field of LLMs may be able to unlock new possibilities. The already promising results achieved using ChatGPT-4 by directly querying its confidence serve as examples of how it could be leveraged in the coming months. If such a method proves effective, during the extraction process of Ketl, the LLM would provide a confidence score alongside the extracted information. This would enable us to decide whether

to present the extracted information to the client or consider it insufficiently relevant without needing more prompt.

# Part IV
# General conclusion

In conclusion, this thesis has explored various aspects of utilizing Large Language Models for the named entity recognition task, particularly focusing on its implications for the Ketl project.

Firstly, we have demonstrated that while LLMs has potential for NER tasks, they present challenges, including precision issues and the need to restrict output. However, employing few-shot techniques and fine-tuning LLMs and have proven effective in mitigating these challenges, with the fine-tuned Mistral-7B model showing particularly promising results for classical NER task, even outperforming ChatGPT. We were not able to attain the performance in term of accuracy and in computational cost of the deep-learning models but this might change in the following months.

Secondly, we have discussed the broader implications of integrating LLMs into the Ketl project using fine-tuning techniques, and the way of prompting. These advancements reveal performance enhancements in extracting information from official documents with LLMs compared to approach using deep-learning methods. This ultimately benefiting Ketl by enabling more efficient and cost-effective operations. However, we still have challenges as it takes a lot of time and resource to process the documents with LLMs.

Finally, we have addressed the aspect of evaluating the confidence of LLMs, acknowledging the complexity of this task due to the nature of logits. Through experimentation, we have shown that training a logistic regression model on labeled data can yield well-calibrated confidence estimates, which could be valuable for Ketl's operations. By leveraging this approach, Ketl can potentially train smaller models on their dataset, providing users with a measure of confidence in the LLM's responses.

In summary, while challenges remain in utilizing LLMs for NER tasks, the potential benefits for Ketl are substantial. With ongoing advancements in LLM technology and innovative approaches to address key challenges, Ketl stands to significantly enhance its capabilities in extracting entities from official documents, ultimately contributing to its mission of efficient document retrieval and management.

# Part V
# Bibliography

## References


(N.d.). In: URL: https://huggingface.co/datasets/toughdata/quora-question-answer-dataset.

Akbik, Alan, Duncan Blythe, and Roland Vollgraf (2018). "Contextual String Embeddings for Sequence Labeling". In: *COLING 2018, 27th International Conference on Computational Linguistics*, pp. 1638–1649.

Ashok, Dhananjay and Zachary C. Lipton (2023). *PromptNER: Prompting For Named Entity Recognition*. arXiv: 2305.15444 [cs.CL].

Bian, Ning et al. (2023). *ChatAlpaca: A Multi-Turn Dialogue Corpus based on Alpaca Instructions*. https://github.com/cascip/ChatAlpaca.

Bird, Steven, Ewan Klein, and Edward Loper (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Bojanowski, Piotr et al. (2017). *Enriching Word Vectors with Subword Information*. arXiv: 1607.04606 [cs.CL].

Collobert, Ronan et al. (2011). *Natural Language Processing (almost) from Scratch*. arXiv: 1103.0398 [cs.LG].

Devlin, Jacob et al. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv: 1810.04805 [cs.CL].

Frantar, Elias et al. (2023). *GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers*. arXiv: 2210.17323 [cs.LG].

Gerganov, Georgi ggerganov (2023). *llama.cpp*. https://github.com/ggerganov/llama.cpp.

Guo, Chuan et al. (2017). *On Calibration of Modern Neural Networks*. arXiv: 1706.04599 [cs.LG].

Honnibal, Matthew and Montani Ines et al. (2021). "What's New in v3.0". URL: https://spacy.io/usage/v3.

Hovy, Eduard et al. (June 2006). "OntoNotes: The 90% Solution". In: *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. New York City, USA: Association for Computational Linguistics, pp. 57–60. URL: https://aclanthology.org/N06-2015.

Hu, Edward J. et al. (2021). *LoRA: Low-Rank Adaptation of Large Language Models*. arXiv: 2106.09685 [cs.CL].

Jha, Aditi et al. (2023). *LIMIT: Less Is More for Instruction Tuning Across Evaluation Paradigms*. arXiv: 2311.13133 [cs.LG].

Jiang, Albert Q., Alexandre Sablayrolles, Arthur Mensch, et al. (2023). *Mistral 7B*. arXiv: 2310.06825 [cs.CL].

Jiang, Albert Q., Alexandre Sablayrolles, Antoine Roux, et al. (2024). *Mixtral of Experts*. arXiv: 2401.04088 [cs.LG].

Keraghel, Imed, Stanislas Morbieu, and Mohamed Nadif (2024). *A survey on recent advances in named entity recognition*. arXiv: 2401.10825 [cs.CL].

Labusch, Kai et al. (Oct. 2019). "BERT for Named Entity Recognition in Contemporary and Historical German". In.

Li, Peng et al. (2023). *CodeIE: Large Code Generation Models are Better Few-Shot Information Extractors*. arXiv: 2305.05711 [cs.CL].

Li, Xin et al. (2017). *Learning with Rethinking: Recurrently Improving Convolutional Neural Networks through Feedback*. arXiv: 1708.04483 [cs.CV].

Liu, Jian et al. (2020). *A hybrid deep-learning approach for complex biochemical named entity recognition*. arXiv: 2012.10824 [cs.CL].

Liu, Yinhan et al. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv: 1907.11692 [cs.CL].

Luo, Man et al. (2024). *In-context Learning with Retrieved Demonstrations for Language Models: A Survey*. arXiv: 2401.11624 [cs.CL].

Ma, Xuezhe and Eduard Hovy (2016). *End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF*. arXiv: 1603.01354 [cs.LG].

Mielke, Sabrina J. et al. (2022). "Reducing Conversational Agents' Overconfidence Through Linguistic Calibration". In: *Transactions of the Association for Computational Linguistics* 10, pp. 857–872. DOI: 10.1162/tacl_a_00494. URL: https://aclanthology.org/2022.tacl-1.50.

Mikolov, Tomas et al. (2013). *Distributed Representations of Words and Phrases and their Compositionality*. arXiv: 1310.4546 [cs.CL].

OpenAI (2023). *GPT-4 Technical Report*. arXiv: 2303.08774 [cs.CL].

Ouyang, Long et al. (2022). *Training language models to follow instructions with human feedback*. arXiv: 2203.02155 [cs.CL].

Pennington, Jeffrey, Richard Socher, and Christopher D. Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

Peters, Matthew E. et al. (2018). *Deep contextualized word representations*. arXiv: 1802.05365 [cs.CL].

Portillo Wightman, Gwenyth, Alexandra Delucia, and Mark Dredze (July 2023). "Strength in Numbers: Estimating Confidence of Large Language Models by Prompt Agreement". In: *Proceedings of the 3rd Workshop on Trustworthy Natural Language Processing (TrustNLP 2023)*. Toronto, Canada: Association for Computational Linguistics, pp. 326–362. DOI: 10.18653/v1/2023.trustnlp-1.28. URL: https://aclanthology.org/2023.trustnlp-1.28.

Radford, Alec et al. (2019). *Language Models are Unsupervised Multitask Learners*.

Reiss, Frederick et al. (2020). "Identifying Incorrect Labels in the CoNLL-2003 Corpus". In: *Conference on Computational Natural Language Learning*. URL: https://api.semanticscholar.org/CorpusID:226283626.

Sakib Shahriar, Kadhim Hayawi (2023). "Let's have a chat! A Conversation with ChatGPT: Technology, Applications, and Limitations". In: URL: https://arxiv.org/abs/2302.13817.

Sun, Zhensu et al. (2024). *When Neural Code Completion Models Size up the Situation: Attaining Cheaper and Faster Completion through Dynamic Model Inference*. arXiv: 2401.09964 [cs.SE].

Tedeschi, Simone et al. (Nov. 2021). "WikiNEuRal: Combined Neural and Knowledge-based Silver Data Creation for Multilingual NER". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 2521–2533. URL: https://aclanthology.org/2021.findings-emnlp.215.

Tjong Kim Sang, Erik F. and Fien De Meulder (2003). "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. URL: https://www.aclweb.org/anthology/W03-0419.

Touvron, Hugo et al. (2023). *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv: 2307.09288 [cs.CL].

Vaswani, Ashish et al. (2017). *Attention Is All You Need*. arXiv: 1706.03762 [cs.CL].

Wang, Shuhe et al. (2023). *GPT-NER: Named Entity Recognition via Large Language Models*. arXiv: 2304.10428 [cs.CL].

Wu, Hui et al. (2023). *Efficient LLM inference solution on Intel GPU*. arXiv: 2401.05391 [cs.AR].

Xiong, Miao et al. (2023). *Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs*. arXiv: 2306.13063 [cs.CL].

Xu, Lingling et al. (2023). *Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment*. arXiv: 2312.12148 [cs.CL].

Xu, Yuhui et al. (2023). *QA-LoRA: Quantization-Aware Low-Rank Adaptation of Large Language Models*. arXiv: 2309.14717 [cs.LG].

Yenduri, Gokul et al. (2018). *Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions*. arXiv: 2305.10435 [cs.CL].

Younes, Belkada et al. (2023). *Making LLMs even more accessible with bitsandbytes, 4-bit quantization and QLoRA*. URL: https://huggingface.co/blog/4bit-transformers-bitsandbytes.

# Part VI
# Appendix

## 1 Prompts

**GPT-NER technique for person entity** ### SYSTEM : The task is to extract all the named entites that are person entities in the following sentence.
### USER : Your goal is to add '@@' at the begining and '##' at the end of all the enities that are person entities. Person entities are all the names you can find in the text. That can be celebrities, historical figures, fictional characters or just random names..
### ASSISTANT : Yes I can do that. Can you provide me examples ?
### USER : Yes of course, there are some examples :

### INPUT : <start_input> Organisers hope to get that run in on Saturday morning , conditions permitting , and stage the race later in the day or on Sunday . <end_input>
### OUTPUT : <start_output> Organisers hope to get that run in on Saturday morning , conditions permitting , and stage the race later in the day or on Sunday . <end_output>

### INPUT : <start_input> Police smashed two drugs smuggling rings and arrested 30 people after a taxidriver in Spain alerted them to a suitcase of heroin left in his cab , Belgian police said on Friday . <end_input>
### OUTPUT : <start_output> Police smashed two drugs smuggling rings and arrested 30 people after a taxidriver in Spain alerted them to a suitcase of heroin left in his cab , Belgian police said on Friday . <end_output>

### INPUT : <start_input> Trade and Industry Secretary Ian Lang added that even if the conditions were met by both airlines , final clearance would hinge on an open skies deal between Britain and the United States to liberalise trans-Atlantic air traffic , which would create greater competition on the routes . <end_input>
### OUTPUT : <start_output> Trade and Industry Secretary @@Ian Lang## added that even if the conditions were met by both airlines , final clearance would hinge on an open skies deal between Britain and the United States to liberalise trans-Atlantic air traffic , which would create greater competition on the routes . <end_output>

### INPUT : <start_input> Faulk carried 16 times , including a 13-yard TD run in the first quarter and a seven-yard score early in the final period . <end_input>

### OUTPUT : <start_output> @@Faulk## carried 16 times , including a 13-yard TD run in the first quarter and a seven-yard score early in the final period . <end_output>

### INPUT : <start_input> -- London Newsroom +44 171 542-7768 <end_input>
### OUTPUT : <start_output> -- London Newsroom +44 171 542-7768 <end_output>

### ASSISTANT : Ok now I understand I need to rewrite the sentence and add '@@' at the begining and '##' at the end of all the enities that are person entities. Can you now provide me the sentence ?
### INPUT : <start_input> The long-distance operator will offer international services , Horinouchi said . <end_input>
### OUTPUT : <start_output>

**Direct output** ### SYSTEM : The task is to extract all the named entites in the following sentence.
### USER : Your goal is to extract all the enities that are either person, organization, location or miscallaneous and output the entities in a list of tuples. In each tuple put the named entity and the tag alongside it.
### ASSISTANT : Can you give me clarification on the different type of entities ?
### USER : Yes. Person entities are all the names you can find in the text. That can be celebrities, historical figures, fictional characters or just random names.
Organization entities all the organizations you can find in the text. That can be business, educational organisation, broadcaster, sports organisation, scientific organisation, political organisation, institute or government agency.
Location entities are all the human-geographic territorials, geographical regions, areas in a single country or natural geographic objects.
Miscellaneous entities are all events or, names, entities and adjectives that are specific but do not have a well-defined category and do not fit in person, organization or location entities
### ASSISTANT : Yes I can do that. Can you provide me examples ?
### USER : Yes of course, there are some examples :

### INPUT : <start_input> Organisers hope to get that run in on Saturday morning , conditions permitting , and stage the race later in the day or on Sunday . <end_input>
### OUTPUT : <start_output> [] <end_output>

### INPUT : <start_input> A newspaper reported allegations in April that diplomats had directed Australian government aid to certain foreign or-

phanages to secure sex with children . <end_input>
### OUTPUT : <start_output> [['Australian', 'MISC']] <end_output>

### INPUT : <start_input> Brazil was in seventh position with 54,333 bags ( 29,055 ) . <end_input>
### OUTPUT : <start_output> [['Brazil', 'LOC']] <end_output>

### INPUT : <start_input> Two ships loaded on the East Coast , three waited to load , six were due . <end_input>
### OUTPUT : <start_output> [] <end_output>

### INPUT : <start_input> Spain 's police seize petrol bombs , arrest five . <end_input>
### OUTPUT : <start_output> [['Spain', 'LOC']] <end_output>
### ASSISTANT : Ok now I understand I need to only output a list with the entities that are in the sentence and the tag along it. Can you now provide me the sentence ?

### INPUT : <start_input> The long-distance operator will offer international services , Horinouchi said . <end_input>
### OUTPUT : <start_output>

**Wrapper** ### SYSTEM : The task is to extract all the named entites in the following sentence.
### USER : Your goal is to extract all the entities that have either tag person, organization, location or miscallaneous.
In order to do this, you have to rewrite the sentence and wrap the named entity by <tag> and </tag>.
### ASSISTANT : Can you give me clarification on the different type of entities ?
### USER : Yes. Person entities are all the names you can find in the text. That can be celebrities, historical figures, fictional characters or just random names.
Organization entities all the organizations you can find in the text. That can be business, educational organisation, broadcaster, sports organisation, scientific organisation, political organisation, institute or government agency.
Location entities are all the human-geographic territorials, geographical regions, areas in a single country or natural geographic objects.
Miscellaneous entities are all events or, names, entities and adjectives that are specific but do not have a well-defined category and do not fit in person, organization or location entities
### ASSISTANT : Yes I can do that. Can you provide me examples ?
### USER : Yes of course, there are some examples :

### INPUT : <start_input> At stake are billions of dollars and the future of the electronic information industry -- the coming medium for the distribution of music , films , literature , software and commerce . <end_input>
### OUTPUT : <start_output> At stake are billions of dollars and the future of the electronic information industry -- the coming medium for the distribution of music , films , literature , software and commerce . <end_output>

### INPUT : <start_input> In the 18 weeks to November 30 , sales were up 13.6 percent year-on-year . <end_input>
### OUTPUT : <start_output> In the 18 weeks to November 30 , sales were up 13.6 percent year-on-year . <end_output>

### INPUT : <start_input> Along with leaders Vicenza , fourth-placed Bologna represent the biggest surprise of this Italian autumn . <end_input>
### OUTPUT : <start_output> Along with leaders <organisation>Vicenza</organisatic , fourth-placed <organisation>Bologna</organisation> represent the biggest surprise of this <miscellaneous>Italian</miscellaneous> autumn . <end_output>

### INPUT : <start_input> China 's State Council , or cabinet , has given a port in the southern province of Hainan permission to open to foreign vessels , the Xinhua news agency said on Friday . <end_input>
### OUTPUT : <start_output> <location>China</location> 's <organisation>State Council</organisation> , or cabinet , has given a port in the southern province of <location>Hainan</location> permission to open to foreign vessels , the <organisation>Xinhua</organisation> news agency said on Friday . <end_output>

### INPUT : <start_input> New Zealand Prime Minister Jim Bolger , emerging from coalition talks with the nationalist New Zealand First party on Friday afternoon , said National and NZ First would meet again on Sunday . <end_input>
### OUTPUT : <start_output> <location>New Zealand</location> Prime Minister <person>Jim Bolger</person> , emerging from coalition talks with the nationalist <location>New Zealand</location> First party on Friday afternoon , said <organisation>National</organisation> and <organisation>NZ First</organisation> would meet again on Sunday . <end_output>
### ASSISTANT : Ok now I understand I need to rewrite the sentence and wrap the named entity by <tag> and </tag>. Can you now provide me the sentence ?

### INPUT : <start_input> The long-distance operator will offer international services , Horinouchi said . <end_input>

### OUTPUT : <start_output>

**Tagger** ### SYSTEM : The task is to tag all the named entites that were extracted from a sentence.
### USER : Your task is to to tag all the named entites that were extracted from a sentence with either
'P' for person entities,
'O' for organization entities,
'L' for location entities,
'M' for miscallaneous entities or
'N' is it is none of the above.
Output a json with the named entities as keys and the tag as values.
### ASSISTANT : Can you give me clarification on the different type of entities ?
### USER : Yes. Person entities are all the names you can find in the text. That can be celebrities, historical figures, fictional characters or just names but not pronouns.
Organization entities all the organizations you can find in the text. That can be business, educational organisation, broadcaster, sports organisation, scientific organisation, political organisation, institute or government agency.
Location entities are all countries, the human-geographic territorials, geographical regions, areas in a single country or natural geographic objects.
Miscellaneous entities are all events, languages, adjectives to describe thing particular to a country. It cannot be verbs, numbers or time related word like weekdays and months.
### ASSISTANT : Yes I can do that. Can you provide me examples ?
### USER : Yes of course, there are some examples :

### INPUT : <start_input> ['Turkish'] in 'The taxidriver alerted police who arrested a 33-year-old Turkish man when he came to pick up the suitcase at a lost luggage office .' <end_input>
### OUTPUT : <start_output> {
'Turkish' : 'M',} <end_output>

### INPUT : <start_input> ['Japan', 'Asian Cup', 'Syria'] in 'Two goals in the last six minutes gave holders Japan an uninspiring 2-1 Asian Cup victory over Syria on Friday .' <end_input>
### OUTPUT : <start_output> {
'Japan' : 'L',
'Asian Cup' : 'M',
'Syria' : 'L',} <end_output>

### INPUT : <start_input> ['Dariusz Rosati', 'Poland', 'Swiss'] in 'For-

eign Minister Dariusz Rosati , unveiling first findings of a special government commission , said that in 1970s the then communist Poland received 460,000 Swiss francs from the accounts .' <end_input>
### OUTPUT : <start_output> {
'Dariusz Rosati' : 'P',
'Poland' : 'L',
'Swiss' : 'M',} <end_output>

### INPUT : <start_input> ['I. Healy', 'P. Reiffel', 'S. Warne', 'J. Gillespie'] in 'Did not bat : I. Healy , P. Reiffel , S. Warne , J. Gillespie .' <end_input>
### OUTPUT : <start_output> {
'I. Healy' : 'P',
'P. Reiffel' : 'P',
'S. Warne' : 'P',
'J. Gillespie' : 'P',} <end_output>

### INPUT : <start_input> ['Umar', 'Bre-X', 'Barrick', 'Busang'] in 'The Ministry 's Umar said on Thursday that both Bre-X and Barrick had responded positively to a government letter recommending a 25-75 split in the Busang gold property .' <end_input>
### OUTPUT : <start_output> {
'Umar' : 'P',
'Bre-X' : 'O',
'Barrick' : 'O',
'Busang' : 'O',} <end_output>
### ASSISTANT : Ok now I understand I need to only output json with the named entities as keys and the tag as values. Can you now provide me the list of extracted entities and the sentence ?

### INPUT : <start_input> after Saturday 's matches ( tabulated - played , won , drawn , lost , <end_input>
### OUTPUT : <start_output> {

**Get Entities** ### SYSTEM : The task is to extract all the named entites in the following sentence.
### USER : Your goal is to extract all the entities that are either person, organization, location or miscallaneous and output the entities in a list. Output all entities even if you are not completely sure it is an entity.
### ASSISTANT : Can you give me clarification on the different type of entities ?
### USER : Yes. Person entities are all the names you can find in the

text. That can be celebrities, historical figures, fictional characters or just names but not pronouns.
Organization entities all the organizations you can find in the text. That can be business, educational organisation, broadcaster, sports organisation, scientific organisation, political organisation, institute or government agency.
Location entities are all countries, the human-geographic territorials, geographical regions, areas in a single country or natural geographic objects.
Miscellaneous entities are all events, languages, adjectives to describe thing particular to a country. It cannot be verbs, numbers or time related word like weekdays and months.
### ASSISTANT : Yes I can do that. Can you provide me examples ?
### USER : Yes of course, there are some examples :

### INPUT : <start_input> West Indies batsman Brian Lara suffered another blow to his Australian tour , after already being disciplined for misconduct , when he was dismissed cheaply in the first limited overs match against Australia on Friday . <end_input>
### OUTPUT : <start_output> ['West Indies', 'Brian Lara', 'Australian', 'Australia'] <end_output>

### INPUT : <start_input> A city official , who declined to be named , explained that Goldman , Sachs , which this summer was demoted to the second tier of the syndicate , proposed the floating rate issue and as a result was promoted to book runner for this offering . <end_input>
### OUTPUT : <start_output> ['Goldman , Sachs'] <end_output>

### INPUT : <start_input> The bishop will jointly receive the Nobel award next Tuesday with East Timorese-born activist Jose Ramos Horta , who lives in self-exile in Australia . <end_input>
### OUTPUT : <start_output> ['Nobel', 'East Timorese-born', 'Jose Ramos Horta', 'Australia'] <end_output>

### INPUT : <start_input> Leeds had already fined Bowyer 4,000 pounds ( $ 6,600 ) and warned him a repeat of his criminal behaviour could cost him his place in the side . <end_input>
### OUTPUT : <start_output> ['Leeds', 'Bowyer'] <end_output>

### INPUT : <start_input> Pace outdistanced three senior finalists -- Virginia Tech defensive end Cornell Brown , Arizona State offensive tackle Juan Roque and defensive end Jared Tomich of Nebraska . <end_input>
### OUTPUT : <start_output> ['Pace', 'Virginia Tech', 'Cornell Brown', 'Arizona State', 'Juan Roque', 'Jared Tomich', 'Nebraska'] <end_output>
### ASSISTANT : Ok now I understand I need to only output a list with

the entities. Can you now provide me the sentence ?

### INPUT : <start_input> after Saturday 's matches ( tabulated - played , won , drawn , lost , <end_input>
### OUTPUT : <start_output> [

**Precision added at the beginning of the prompt for Ontonote5 dataset** A named entity refers to a specific, named object, concept, location, person, organization, or other entities that have a proper name. Named entities are
typically unique and distinguishable entities that can be explicitly named or
referred to in text. Named entities are stopwords like 'the', verbs like 'serving' or question words like 'why'.
The goal of named entity extraction is to identify and classify these entities within a given text.
We are working with 18 types of entities of the OntoNote5 dataset that are listed below with their description :
"CARDINAL": "Numerals that do not fall under another type (e.g. 1, 100,
twenty-nine).",
"ORDINAL": "Words or expressions indicating order (e.g. first, 60th).",
"WORK_OF_ART": "Titles of creative works like books, films or artistic work.",
"PERSON": "Names of people, including fictional and real characters.",
"LOC": "Geographical locations, both physical and political.",
"DATE": "Temporal expressions indicating dates or periods, weekday, months,",
"PERCENT": "Percentage values (e.g. 70"PRODUCT": "Names of products or services.",
"MONEY": "Monetary values, including currency symbols.",
"FAC": "Named facilities, such as buildings, airports, or highways.",
"TIME": "Temporal expressions indicating times of the day.",
"ORG": "Names of organizations, institutions, or companies.",
"QUANTITY": "Measurements or counts, including units (e.g. 10 grams, 1 litre)",
"LANGUAGE": "Names of languages.",
"GPE": "Geopolitical entities, such as countries, cities, or states.",
"LAW": "Legal references, including laws and legal concepts.",
"NORP": "Nationalities, religious group, or political groups. Can be adjective for nationality like "Canadian".",
"EVENT": "Named occurrences and social, political, cultural, or genera incidents.

## 2 Finetuning

**Finetuning on OntoNote5 with raw text** The task is to label all the person entities in the given sentence. Person entities all the organizations you can find in the text. That can be business, educational organization, broadcaster, sports organization, scientific organization, political organization, institute or government agency. When you find one add '@@' at the beginning and '##' at the end of the entity.
### QUESTION: Camilla Martin ( Denmark ) beat Wang Chen ( China ) 11-0 12-10
### ANSWER: <start_answer> @@Camilla Martin##( Denmark ) beat @@Wang Chen## ( China ) 11-0 12-10<end_answer>

**Finetuning on OntoNote5 with random number of fewshots between 1 and 4** ###
SYSTEM : The task is to extract all the named entites in the following sentence.
### USER : Your goal is to extract all the enities that are either person, organization, location or miscallaneous and output the entities in a list of tuples. In each tuple put the named entity and the tag alongside it.
### ASSISTANT : Can you give me clarification on the different type of entities ?
### USER : Yes. Person entities are all the names you can find in the text. That can be celebrities, historical figures, fictional characters or just random names.
Organization entities all the organizations you can find in the text. That can be business, educational organisation, broadcaster, sports organisation, scientific organisation, political organisation, institute or government agency.
Location entities are all the human-geographic territorials, geographical regions, areas in a single country or natural geographic objects.
Miscellaneous entities are all events or, names, entities and adjectives that are specific but do not have a well-defined category and do not fit in person, organization or location entities
### ASSISTANT : Yes I can do that. Can you provide me examples ?
### USER : Yes of course, there are some examples :

### INPUT : <start_input> Organisers hope to get that run in on Saturday morning , conditions permitting , and stage the race later in the day or on Sunday . <end_input>
### OUTPUT : <start_output> [] <end_output>

### INPUT : <start_input> A newspaper reported allegations in April that diplomats had directed Australian government aid to certain foreign or-

```
phanages to secure sex with children .  <end_input>
### OUTPUT : <start_output> [['Australian', 'MISC']] <end_output>

### INPUT : <start_input> Brazil was in seventh position with 54,333
bags ( 29,055 ) .  <end_input>
### OUTPUT : <start_output> [['Brazil', 'LOC']] <end_output>

### INPUT : <start_input> Two ships loaded on the East Coast , three
waited to load , six were due .  <end_input>
### OUTPUT : <start_output> [] <end_output>

### INPUT : <start_input> Spain 's police seize petrol bombs , arrest
five.<end_input>
### OUTPUT : <start_output> [['Spain', 'LOC']] <end_output>
### ASSISTANT : Ok now I understand I need to only output a list with
the entities that are in the sentence and the tag along it.  Can you
now provide me the sentence ?

### INPUT : <start_input> The long-distance operator will offer inter-
national services , Horinouchi said .  <end_input>
### OUTPUT : <start_output>[['Horinouchi', 'PER']] <end_output>
```

## 3 Grammars

**Basic grammar for JSON from llama.cpp**

```
root ::= object value ::= object
| array | string | number | ("true" | "false" | "null") ws
  object ::= "\" ws ( string ":" ws value ("," ws string ":" ws value)*
)?  "\" ws
  array ::= "[" ws ( value ("," ws value)* )?  "]" ws
  string ::= "\"" ( [^\\] | "\\" (["\\/bfnrt] | "u" [0-9a-fA-F] [0-9a-
fA-F] [0-9a-fA-F] [0-9a-fA-F]) # escapes )* "\"" ws
  number ::= ("-"?  ([0-9] | [1-9] [0-9]*)) ("." [0-9]+)?  ([eE] [-
+]?  [0-9]+)?  ws
  # Optional space:  by convention, applied in this grammar after lit-
eral chars when allowed ws ::= ([ \t\n] ws)?
```

**Grammar used for document type extraction with chain-of-tough**

```
root ::=
(reflection "Therefore, here is the json with the type of document :
" json)
  reflection ::= ("The document you sent is a document of type " string
".  " isInArray)
  isInArray ::= ("This type of document is in the possible document
types array.  " | ( "This type of document is not in the possible doc-
```

```
ument types array.  The type of document in the list that corresponds
the most to this type of document is " string ".  "))
   json ::= "\ \"Type de document\" :  " string "\"
   string ::= "\"" ( [^\\\[\]] | "\\" (["\\/bfnrt] | "u" [0-9a-fA-F] [0-
9a-fA-F] [0-9a-fA-F] [0-9a-fA-F]) # escapes )* "\""
   ws ::= ([ \t\n] ws)?
```

**Grammar used for field extraction with chain-of-tough**

```
root ::= "I have read
the document and after thinking a little bit I think I have found the
information you wanted.  Here is a json with all the informations :  "
object value ::= object | array | string | number | ("true" | "false"
| "null") ws
   object ::= "\" ws ( string ":" ws value ("," ws string ":" ws value)*
)?  "\" ws
   array ::= "[" ws ( value ("," ws value)?  ("," ws value)?  )?  "]"
ws
   string ::= "\"" ( [^\\] | "\\" (["\\/bfnrt] | "u" [0-9a-fA-F] [0-9a-
fA-F] [0-9a-fA-F] [0-9a-fA-F]) # escapes )* "\"" ws
   number ::= ("-"?  ([0-9] | [1-9] [0-9]*)) ("." [0-9]+)?  ([eE] [-
+]?  [0-9]+)?  ws
   # Optional space:  by convention, applied in this grammar after lit-
eral chars when allowed ws ::= ([ \t\n] ws)?
```