ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

Department of Computer Science and Engineering
Artificial Intelligence

Leveraging Large Language Model Distillation to Enhance Zero-Shot Named Entity Recognition and Classification

Master Thesis in
Natural Language Processing

Supervisor
Prof. Cluadio Sartori

Candidate
Alessio Cocchieri

Co-supervisors
Prof. Gianluca Moro
Giacomo Frisoni
Marcos Martínez Galindo

Second Session
Academic Year 2022 – 2023

KEY WORDS

Large Language Models

Natural Language Processing

Zero-shot Learning

Named Entity Recognition

Knowledge Distillation

Abstract

Named entity recognition and classification (NERC) is a crucial task in natural language processing. Annotations help a lot in this task but, in the real world, annotations are chronically difficult to obtain and generalization to unseen types loom large. Embracing zero-shot learning becomes essential to surmount the absence of training examples. However, substantial prior knowledge is required to achieve remarkable outcomes, particularly in domain-specific scenarios. Although large language models (LLMs) hold great potential, computational cost and inefficiency severely hamper their applicability, favoring smaller specialized networks. In this paper, we propose JUICER, the first LLM distillation framework for zero-shot NERC in resource-constrained environments. Mechanically, JUICER transfers LLM knowledge to BERT-based models with a preliminary fine-tuning process centered on generative data augmentation above massive pre-training corpora. Generalizability is further promoted by injecting textual target class descriptions through cross-attention. We conduct extensive experiments on three zero-shot adapted BIO-format datasets. In this pursuit, we center our distillation process on biomedicine and assess adaptability to news and legal domains. Our knowledge-distilled models outperform state-of-the-art baselines across all benchmarks up to 0.27 macro-averaged F1 points, proving the benefit of numerous observed classes. Compared to zero-shot and reparametrized LLMs, we achieve superior overall results across all datasets using 510× fewer parameters. Interestingly, when trained in cross-domain setups, JUICER models experience a further increase of up to 0.07 points.

Introduction

The task of named entity recognition and classification (NERC) entails the identification of named entity mentions and their subsequent categorization into specific entity-types from a predefined set. Despite the upswing in harnessing the expressive power of pre-trained language models [1], existing solutions often require substantial datasets and prioritize a limited subset of frequently encountered classes (e.g., person, location, and organization) [2]. In real-world scenarios, NERC applications frequently confront challenges related to specialized domains, grappling with a scarcity of readily available labels for non-standard classes, while constantly encountering emerging new targets. Within this landscape, the practicality of adopting or recurrently retraining state-of-the-art (SOTA) models becomes hindered. Simultaneously, the creation of comprehensive ground-truth annotations proves resource-intensive, liable to human errors and inconsistencies that could impact the efficacy of models. As a result, there is a shift in the focus toward zero-shot learning (ZSL) [3], which requires networks to extrapolate, adjust to entities not encountered during training, and effectively transfer knowledge derived from observed classes. Prior knowledge emerges as a crucial determinant of success [4].

Embracing the in-context learning approach, large language models (LLMs)[1] have showcased remarkable achievements by generating high-quality text tailored to specific directives [5]. Benefiting from extensive pre-training on substantial volumes of unprocessed text, LLMs serve as valuable reservoirs of knowledge to fortify NERC in zero-shot learning scenarios. Conversely, the considerable dimensions of these models, ostensibly correlated with their per-

[1]The literature has yet to establish consensus on the minimum scale for LLMs, but we consider ≥ 1B parameters.

formance [6], present significant challenges during deployment. To illustrate, the deployment of a 175B LLM necessitates a dedicated infrastructure with at least 350GB of GPU RAM [7]. Furthermore, the latest state-of-the-art LLMs now exceed 500B [8], considerably escalating computational demands. This situation renders equivalent resources prohibitively expensive for most product teams, particularly those aiming for efficient low-latency inference. This is where small language models (SLMs) come into play, offering a more feasible, transparent, and effective alternative. Due to their limited capacity, researchers have delved into innovative training methodologies, with a focus on knowledge distillation (KD) playing a prominent role.

We postulate that LLMs encapsulate pertinent task knowledge that would otherwise require a substantial number of human-labeled instances for SLMs to attain. Expanding on this concept, we introduce JUICER, an innovative framework that strengthens the proficiency of SLMs in zero-shot NERC by using KD from LLMs. Initially, we utilize a frozen LLM to create an extensive and varied domain distillation dataset through example-guided data augmentation on raw corpora, thus bridging the void resulting from manual annotation. Following this, we employ a dual-phase KD process. In the first phase, a BERT-based student model is trained to replicate the unbounded entity annotations inferred by the teacher LLM using the synthetic dataset. In the second phase, the student model is fine-tuned to specialize in the downstream NERC domain.

We conduct experiments using zero-shot-adapted versions of three widely used NERC datasets: MedMentions [9], OntoNotes 5.0 [10], and LegalNER [11]. To adhere to the ZSL requisites [12], we painstakingly ensure the absence of any entity class overlap between training, validation, and test sets, specifically reserving the scarcer types for evaluation purposes. Our distillation procedure is centered on biomedicine, with a subsequent assessment of its adaptability to other domains. Remarkably, our findings demonstrate that high-quality, extensive synthetic data derived from LLMs, even when tailored to a specific domain, have a positive impact on enhancing the predictive capability of models. This enhancement applies to foreseeing entities across various classes and fields. Our distilled models achieve a token-level macro F1 score of 0.32 in MedMentions, 0.46 in OntoNotes, and 0.38 in LegalNER, markedly outperforming the

previous SOTA ZSL SLMs and LLMs in inference mode. Moreover, compared to fine-tuned reparameterized LLMs, our models demonstrate superiority or comparability while maintaining a significantly reduced architectural size, 510× smaller. Further improvements are realized through cross-domain fine-tuning, resulting in improved scores of 0.07 and 0.05 on OntoNotes and LegalNER respectively.

Building upon the contextual foundation we have laid out, the forthcoming chapters of this thesis provide an in-depth exploration of our research, elucidating our methodologies, and presenting our findings:

- **Chapter 1 - Theoretical Framework**: In this foundational chapter, we lay the groundwork for understanding the core concepts that underpin our research. We begin by exploring the significance of NER as an information processing task. Then, we delve into the world of LLMs, providing an overview of their role in modern language processing. This chapter serves as the gateway to our thesis, equipping readers with the necessary knowledge to better understand the investigations in the subsequent chapters.

- **Chapter 2 - Related Work**: In this chapter, we navigate the landscape of existing research and studies that have paved the way for our investigation. A comprehensive review of the relevant literature sheds light on the fundamental concepts and approaches in the field, establishing a context for our work.

- **Chapter 3 - Method**: This chapter provides an in-depth exploration of our proposed methodology, JUICER. We dissect the details of how we harness large language models and their knowledge distillation to enhance zero-shot NERC capabilities of SLMs. Step by step, we elucidate the process of crafting our framework and the rationale behind its design.

- **Chapter 4 - Experimental Setup**: In this chapter, we detail the experimental setup used for our research. We provide insights into the datasets employed for evaluation, the metrics used to measure performance, implementation specifics, and hyperparameters. This comprehensive account of

our experimental configuration sets the stage for the subsequent chapter where we present our results.

- **Chapter 5 - Results and Discussion**: Within this chapter, we present the outcomes of our experiments and analyses. We delve into the data, revealing the quantitative and qualitative results that underscore the efficacy of our approach. Through comprehensive discussions, we provide insightful interpretations of these results, leading to a deeper understanding of the implications and contributions of our research.

- **Conclusion and Future Work**: Wrapping up our thesis, this final part synthesizes the key findings, implications, and contributions of our research. We reflect on the significance of our work and its potential impact. In addition, we chart the course for future research directions, identifying unexplored avenues and opportunities to build upon the foundation we have established.

The content of this thesis pertains to the research carried out throughout my internship at the IBM Research Lab in Dublin, within the framework of the IBM Accelerated Discovery program.[2]

[2] `https://research.ibm.com/topics/accelerated-discovery`

Index

List of Figures

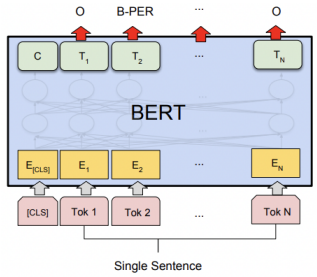
List of Tables

Chapter 1

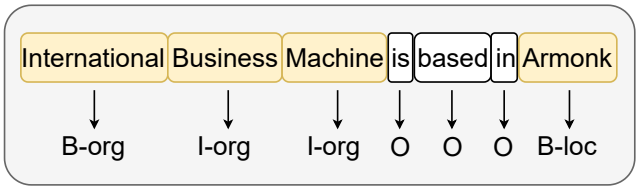
Theoretical Framework

1.1 Named Entity Recognition

Named-entity recognition (NER), also referred to as named entity identification, entity chunking, or entity extraction, represents a pivotal subtask in information processing, focusing on the identification and categorization of crucial information (entities) within text. These entities encompass any words or word sequences that consistently reference a particular concept or entity. The NER process centers on a fundamental two-step approach: (i) detecting a named entity and (ii) categorizing it. During the detection process, the model identifies tokens or sequences of tokens that form an entity, with each token representing a word or a word piece. Commonly utilized tagging formats, such as the IOB format (short for for inside, outside, beginning) and also known as the BIO format, are employed to demarcate the boundaries of entities within the text. While there are several variants of the IOB format, in this work, we employ the widely recognized IOB2 format, which involves labeling tokens at the beginning of a chunk with the prefix *B-*, tokens within a chunk with the prefix *I-*, and tokens unrelated to any chunk with the tag *O* (Figure 1.1b). Throughout this thesis, our focus centers on Named Entity Recognition and Classification (NERC). The "C" in NERC underscores the critical aspect of "Classification". This places significant emphasis not only on recognizing named entities within text but also on classifying them into predetermined categories



(a) BERT model for NER. Source: [13].



(b) Example of token classification in BIO format applied to a sentence.

Figure 1.1: BERT for NER and token classification in BIO format.

(a) demonstrates the adaptation of BERT's architecture for NER, where a distinct output embedding is added for each token. (b) provides an illustrative example of token classification following the IOB2 format. In this example, International *and* Armonk *are prefixed with* B- *since they represent the first token of their respective entities, while* Business *and* Machine *are prefixed with* I- *as they are successive tokens within the entity.*

or types. While NER and NERC are often used interchangeably, the NERC framework distinctly highlights the classification component.

In our research, we leverage BERT-based models for NER tasks. What distinguishes BERT for text classification from its application to the NER problem is the manner in which we configure the model's output. For text classification, we exclusively utilize the embedding vector output generated from the special `[CLS]` token. However, when employing BERT for NER tasks, we harness the embedding vector output from all tokens, facilitating token-level text classification (Figure 1.1a). This enables us to predict the entity associated with each individual token, aligning with our NER objectives.

1.2 Large Language Models

Large language models (LLMs) refer to *large*, *general-purpose* language models that can be pre-trained and then fine-tuned for specific purposes. The

term *large* indicates two meanings: (i) the enormous size of the training data set (sometimes up to petabyte scale), (ii) the tremendous number of parameters. An LLM is essentially a Transformer-based neural network [14] and is trained to solve common NLP tasks like text classification, question-answering, text summarization and text generation. The goal of the model is to predict the text that is likely to follow. With the advent of transformers and transfer learning, the adaptation of language models for various tasks initially required only a modest extension of the network's final layers (the head) and subsequent fine-tuning. However, this approach has now become outdated. Modern developments have evolved to the extent that a multitude of tasks can be effectively executed using the same LLM by simply switching the instructions within the prompt.

From this perspective, effective prompts are key, while skill in crafting them (*Prompt Engineering*) is crucial to harness the potential of LLM. The complexity and efficacy of a model can be evaluated based on its parameter count—indicating the range of factors it accounts for during output generation. To familiarize themselves with the intricacies and interconnections of language, extensive data pre-training is undertaken for LLMs. Pre-training necessitates substantial computational resources and cutting-edge hardware to accomplish this crucial step. Using techniques such as *Fine-tuning* and *In-context learning*, these models can be adapted for downstream (specific) tasks.

1.3 Prompt Engineering

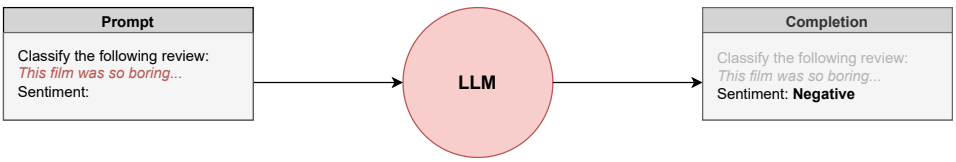


Figure 1.2: Zero-shot inference of LLMs.

The term *prompt* refers to the input text provided to the model, while the process of generating text is termed as *inference*, and the resulting text is termed as the *completion* (Figure 1.2). The full amount of text or the memory

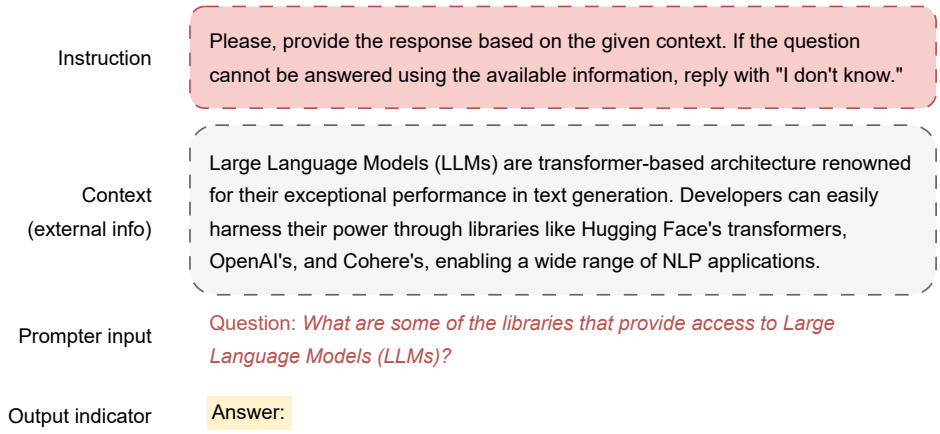


Figure 1.3: Typical key components of a well-designed prompt.

This image illustrates a structured prompt utilized for instructing language models. ***Instructions*** *provide directives that guide the model's actions and define its task.* ***Context*** *represents additional knowledge sources, which can be manually inserted or retrieved from databases, APIs, or other means.* ***Prompter Input*** *typically refers to a query input into the system by a human user (the prompter).* ***Output Indicator*** *serves as a marker indicator for the beginning of the to-be-generated text.*

that is available to use for the prompt is called the *context window*. It is often necessary to iteratively refine the language or structure of your prompt to achieve the desired behavior from the model. This iterative effort to develop and enhance the prompt is called *prompt engineering*.

Prompt Engineering involves designing and crafting effective input instructions to guide the language model towards the generation of desired responses. The main goal of prompt engineering is to influence the behavior of the language model. These prompts can be in the form of questions, statements, or incomplete sentences. Effective prompt engineering involves a combination of experimentation, understanding the underlying task, analyzing the model behavior, and iteratively refining the prompts to achieve the desired results. Although there is no one-size-fits-all formula for crafting effective prompts, four key components are commonly found in well-designed prompts, as illustrated in Figure 1.3.

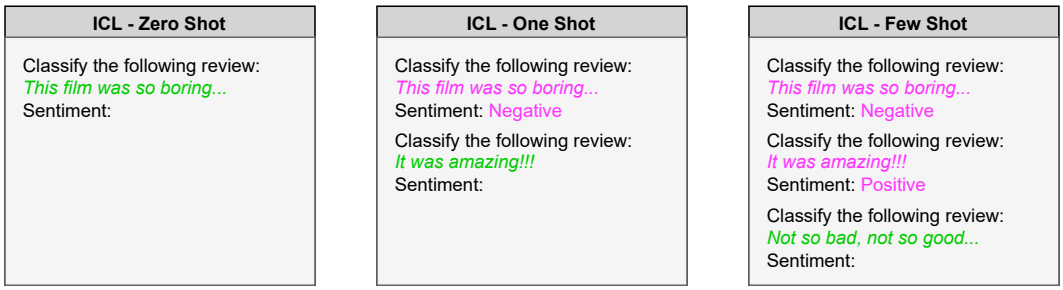


Figure 1.4: Different types of In-context Learning.

In the image, various prompts are presented for conducting sentiment analysis using LLMs. Sentences highlighted in violet serve as examples to guide the model in generating completions. Sentences highlighted in green are used as new input for classification. ***Left Image:*** *The model is tasked with sentiment classification without any provided examples.* ***Center Image:*** *A single example is included within the context window to aid the model in its classification task.* ***Right Image:*** *Multiple examples are presented to further assist the model in performing sentiment analysis.*

1.4 In-context Learning

The two fundamental approaches for imparting knowledge to an LLM encompass: (i) *Parametric knowledge*, wherein knowledge resides within the model's learned weights or parameters during training, and (ii) *Source knowledge*, referring to any information supplied to the model during inference through the input prompt, which includes context and example-guided prompts. In-context learning (ICL) refers to the ability of LLMs to learn and improve their performance on a specific task or dataset by fine-tuning them on a small amount of relevant data. This approach is also known as *Zero-/one-/few-shot learning* based on the number of examples within the context window (Figure 1.4). Traditionally, training a machine learning model requires a large amount of labeled data, where the model learns to recognize patterns and relationships between inputs and outputs. However, with ICL, the goal is to train a model that can quickly adapt to new situations or tasks with minimal additional data within the input prompt.

LLMs have been shown to be particularly suitable for ICL [15]. The largest

models demonstrate remarkable proficiency in zero-shot inference, successfully handling a diverse array of tasks for which they weren't explicitly trained. This proficiency arises from their training on vast volumes of text data, which equips them to encode language at multiple levels of abstraction. Consequently, they can effortlessly transfer their acquired knowledge to novel contexts and tasks, showcasing their adaptability and flexibility. In contrast, smaller models, typically only excel at a limited number of tasks, primarily those closely resembling their training tasks. However, smaller models can significantly enhance their performance through one-shot or few-shot inference methods.

1.5 Generative Parameters Configuration

Each model exposes a set of configuration parameters that can influence the model's output during inference. In this section, we will go through their exploration to understand how they can actually affect the final decision of the model.

1.5.1 Max New Tokens

Among these parameters, `max_new_tokens` stands out as one of the most straightforward. It serves as a mechanism to restrict the total number of tokens the model will generate. Think of it as a limit on the number of iterations the model will undergo in the decision-making process. It's important to note that this value isn't set in stone; the model may generate fewer tokens than the specified maximum if another stopping condition is met, such as predicting the end of a sequence.

1.5.2 Decoding Strategy

The output from the transformer's softmax layer represents a probability distribution across the entire vocabulary of words utilized by the model. In the default mode, most large language models employ a technique known as *greedy decoding* for next-word prediction. This approach is straightforward: the model always selects the word with the highest probability. While effective for

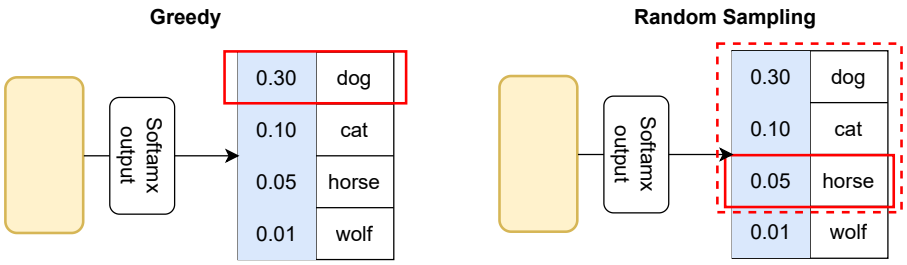


Figure 1.5: Comparing Greedy and Random decoding strategies.

In the left panel, we illustrate the greedy approach where the word/token with the highest probability is selected. In contrast, the right panel showcases random sampling, a strategy where a token is chosen using a randomized weighting based on the probabilities of all tokens. In this example, there is a 30% chance of selecting dog, *but* horse *is the token actually chosen.*

short text generation, it can lead to repetitive words or sequences. To generate more natural and creative text that avoids word repetition, alternative controls are necessary. One such method is *random sampling*. Instead of consistently choosing the most probable word, random sampling involves selecting an output word randomly, weighted by its probability distribution. For instance, if a word like *horse* has a probability score of 0.05, random sampling assigns a 5% chance for it to be selected. This sampling technique reduces the likelihood of word repetition. However, depending on the specific settings, random sampling may introduce excessive creativity, resulting in text that veers into unrelated topics or includes nonsensical words. A comparison between the two approaches is illustrated in Figure 1.5.

1.5.3 Top-k

Another sampling technique called *top* k can be employed to mitigate the potential side-effects of pure random sampling. With *top* k, you can control the options available to the model while introducing some variability. By specifying a *top* k value, you instruct the model to consider only the k tokens with the highest probability for selection. For instance, if you set $k = 3$, the model will choose from the three most probable options. It does so by applying probability

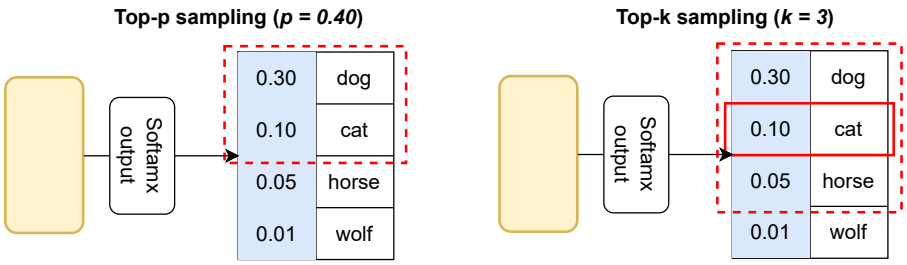


Figure 1.6: Comparing Top-p and Top-k sampling strategies.

*In the left image, we demonstrate the top-*p *approach, where an output is selected using a random-weighted strategy from the consecutive results with the highest probabilities, ensuring that the cumulative probability remains* <= p. *On the right, we illustrate the top-*k *approach, where an output is chosen from the top-*k *results obtained by applying a random-weighted strategy based on their probabilities.*

weighting, and that's why in Figure 1.6 (right), *cat* is selected as the next word. This approach allows the model to maintain a degree of randomness while preventing the selection of highly improbable completion words.

1.5.4 Top-p

Alternatively, the *top-p* setting provides another method to control random sampling by limiting the predictions to those whose combined probabilities do not surpass a specified value, denoted as p. For example, as shown in Figure 1.6 (left), if you set $p = 0.4$, you include options like *dog* and *cat* because their individual probabilities of 0.3 and 0.1 add up to 0.4. The model then employs the random probability weighting approach to select from these tokens. In summary, while *top-k* lets you determine the number of tokens to randomly choose from, *top-p* allows you to specify the total probability that the model should consider when making its selection.

1.5.5 Temperature

Another parameter that allows control over the model's output randomness is known as *temperature* (t). This parameter plays a crucial role in shaping

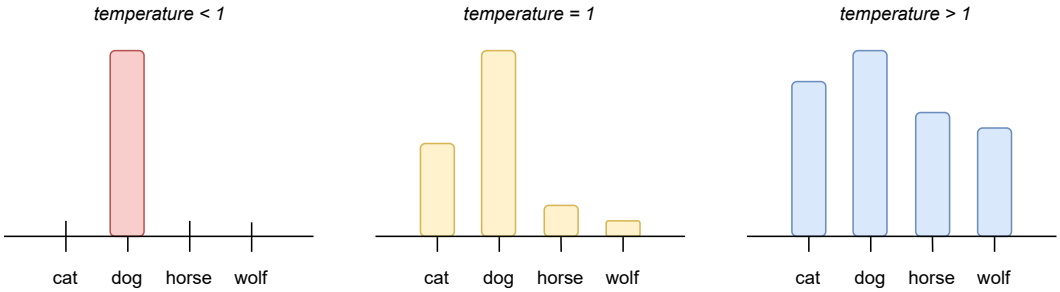


Figure 1.7: Comparing different temperature settings.
With a low temperature ($t < 1$), the probability distribution becomes sharply peaked, as seen in the red bars. Most of the probability concentrates on a single word, such as dog. *When random sampling is applied to this distribution, the generated text becomes less random, closely following the most likely word sequences the model learned during training. Conversely, setting a higher temperature ($t > 1$) results in a broader and flatter probability distribution for the next token (blue bars). This encourages the model to produce text with a higher degree of randomness and variability, fostering more creative-sounding output. Leaving $t = 1$ (yellow bars) retains the default softmax function, utilizing the unaltered probability distribution.*

the probability distribution that the model calculates for the next token. Temperature acts as a scaling factor applied within the final softmax layer of the model, directly influencing the shape of the probability distribution for the next token. Unlike top-k and top-p parameters, altering the temperature directly impacts the model's predictions. In general, a lower temperature leads to a less-random generated text, while a higher temperature produces text with a higher degree of randomness and variability (Figure 1.7).

1.6 Fine-tuning

ICL has a couple of limitations. First, it may not consistently produce desired results for smaller models, even when incorporating five or six examples. Second, any examples included in the prompt consume precious space within the context window, diminishing the available space for other valuable information. Fine-tuning is a supervised learning process that makes use of labeled examples

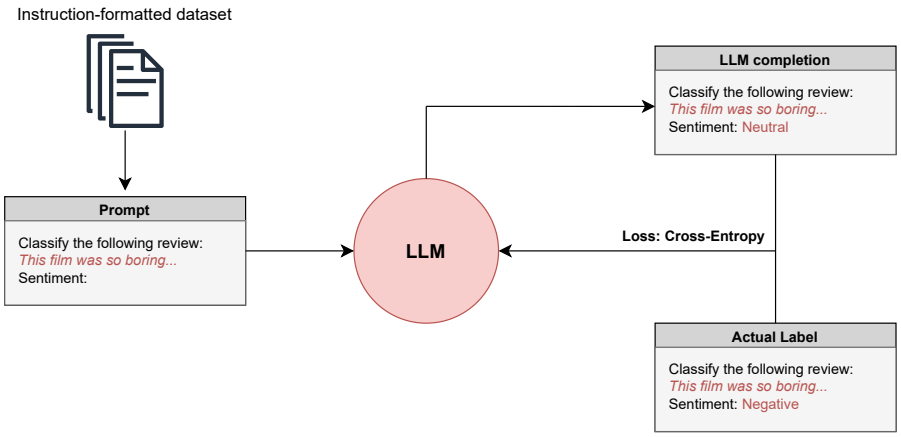


Figure 1.8: Overview of the Fine-tuning process.

In this instance, the model misclassifies a negative *review as* neutral*, necessitating corrective action. It initiates a comparison between the completion and training label distributions, forming the basis for loss calculation using standard cross-entropy. The resulting loss updates model weights via backpropagation across multiple epochs, enhancing its performance.*

to update the weights of the LLM with two primary objectives:

1. Behavior Enhancement:
    - *Improving Consistency*: Adapting the LLM to respond more consistently in various contexts.
    - *Focusing Abilities*: Enhancing specific capabilities, such as moderation or conversational skills.
2. Knowledge Acquisition:
    - *Enhancing Conceptual Knowledge*: Increasing the LLM's understanding of new, specific concepts.
    - *Correcting Previous Misinformation*: Rectifying inaccuracies or outdated information within the model's knowledge base.

Fine-tuning with instruction prompts is the most common way to fine-tune LLMs these days, where instruction refers to the description of tasks.

Henceforth, when referring to fine-tuning in the context of LLMs, it implies instruction fine-tuning (IT). It involves updating all of the model's weights and is commonly referred to as *full fine-tuning*. This process leads to the creation of a new model version with updated weights. IT is a technique that involves fine-tuning pre-trained LLMs on a collection of instruction-formatted datasets [16]. Tuning in this way, LLMs can generalize to unseen tasks by following new instructions, thus boosting ICL performance [16]. Figure 1.8 provides an illustrative example of the fine-tuning process.

This simple yet effective idea has sparked the success of subsequent works in the realm of NLP, such as ChatGPT, InstructGPT [17], FLAN [16, 18].

1.7 Different Types of LLMs

Transformer models encompass three distinct variations: encoder-only models, encoder-decoder models, and decoder-only models. These variations undergo training with diverse objectives, equipping them with specialized capabilities for various tasks. Figure 1.9 provides a comparative overview of these model architectures alongside their respective pre-training objectives.

1.7.1 Autoencoding

Encoder-only models, also referred to as *autoencoding* models, undergo pre-training via *Masked Language Modeling* (MLM). In this training approach, tokens within the input sequence are randomly masked by making use of *mask tokens* (e.g. `<MASK>`), and the primary objective is to predict these masked tokens in order to reconstruct the original sentence. This process is commonly known as a *denoising objective*. Autoencoding models excel in capturing bidirectional representations of the input sequence, enabling them to comprehend the complete context of a token rather than solely considering preceding words. These models find application in various natural language processing tasks, including sentence classification tasks such as sentiment analysis, as well as token-level tasks like NER and word classification. Prominent examples of autoencoding models include BERT and RoBERTa [13, 19]

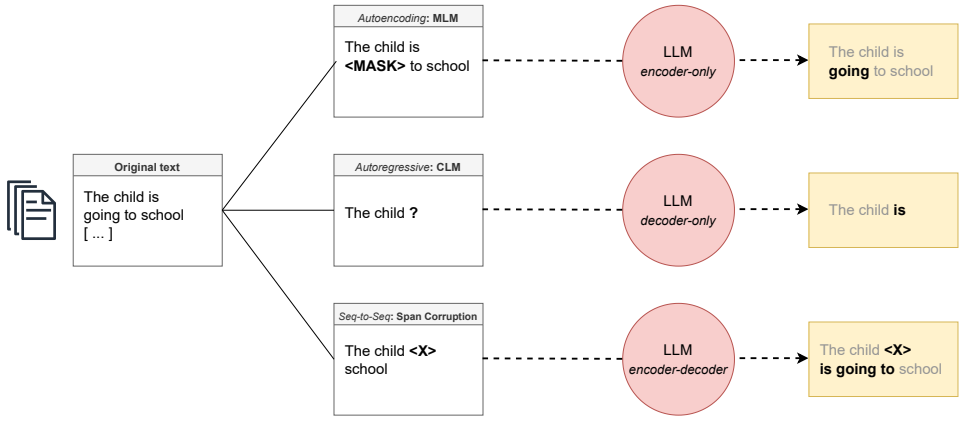


Figure 1.9: Model architectures and pre-training objectives.

Summary of different model architectures and targets of the pre-training objectives. Autoencoding models employ Masked Language Modeling *and correspond to the encoder in the original Transformer, suitable for tasks like text classification. Autoregressive models, based on* Causal Language Modeling*, use the decoder for text generation. Sequence-to-sequence models leverage both encoder and decoder components for various pre-training objectives, aimed at tasks such as translation and summarization.*

1.7.2 Autoregressive

Decoder-only models are commonly referred to as *autoregressive* models. These models undergo a pre-training process utilizing *Causal Language Modeling* (CLM), where the primary objective is to predict the next token in a sequence based solely on the preceding tokens. This predictive task is commonly termed *full language modeling* in academic literature. Decoder-based autoregressive models differ significantly from their encoder-based counterparts. They mask the input sequence and can only access information from tokens that precede the one currently under consideration. As a consequence, these models lack knowledge of the sentence's end and instead progress through the input sequence token by token, predicting each subsequent token individually. Unlike encoder-based architectures, decoder-only models operate within a unidirectional context. Through extensive training on a vast corpus of textual data, these models acquire the ability to generate coherent and contextually appropriate text. While they are frequently employed for text generation tasks, it is worth noting that larger

decoder-only models exhibit robust zero-shot inference capabilities, allowing them to excel across a wide spectrum of natural language processing tasks. Prominent examples of decoder-based autoregressive models in the field of natural language processing include GPT and BLOOM [20, 21].

1.7.3 Sequence-to-Sequence

The final variant within the transformer model family is the sequence-to-sequence model, which amalgamates both the encoder and decoder components from the original transformer architecture. These models exhibit varying pre-training objectives across different implementations. For instance, one prominent sequence-to-sequence model, T5 [22], employs *span corruption* during the encoder's pre-training phase, where it randomly masks sequences of input tokens. These masked sequences are then replaced by a distinct token referred to as the *Sentinel token* (e.g. `<X>`). Sentinel tokens are special tokens introduced into the model's vocabulary but do not correspond to any specific word in the input text. Subsequently, the decoder's role is to reconstruct the sequences of masked tokens in an auto-regressive manner, with the output consisting of the Sentinel token followed by the predicted tokens. Sequence-to-sequence models find applications in tasks such as translation, summarization, and question-answering, proving particularly valuable when dealing with text input and text output. In addition to T5, another noteworthy encoder-decoder model in this category is BART [23].

1.8 Limits of LLMs

In the realm of LLMs, it is imperative to acknowledge and address the inherent limitations that accompany their unparalleled language generation capabilities. These limitations extend beyond the scope of language itself, venturing into critical domains such as reliability, bias, and environmental impact. In particular, we can distinguish several critical points:

- **Hallucinations and Bias**: LLMs have the main goal of producing language that will be perceived as *natural* by humans. They are not

designed to produce truthful information. As a result, being trained on large amounts of data, these models can inadvertently perpetuate false or inaccurate information (a.k.a. *hallucinations*). Additionally, they are not immune to bias, including race, gender, and sex bias, often reflecting the biases present in their training data. Instances where these models generate racist or sexist comments underscore the critical need for vigilant data curation and ongoing model evaluation.

- **Context Window**: Another limitation arises from the practical constraints of the context window. Each large language model has finite memory, allowing it to process only a specific number of tokens as input. For example, LLAMA 2 is constrained by a limit of 4096 tokens [24], which translates to roughly 3K words. Inputs exceeding this token limit challenge the model's ability to provide meaningful responses, highlighting the importance of brevity and clarity in interactions with these models.

- **System Cost**: The development and utilization of LLMs require substantial investment. This encompasses not only computational resources but also human expertise in the form of engineers, researchers, and scientists. The sheer resource intensity of these models makes them accessible primarily to large enterprises with substantial means. For instance, projects like *Megatron-Turing* from NVIDIA and Microsoft are estimated to involve total costs nearing $100 million [25]. These financial barriers can limit the democratization of LLM technologies.

- **Environmental Impact**: LLMs, particularly during training, exert a considerable environmental footprint. The cooling requirements of these systems further escalate their energy usage and environmental impact. A notable study equated the training of BERT [13] on GPU to the environmental impact of a trans-American flight [26]. These environmental concerns underscore the need for sustainable practices in the development and deployment of LLMs.

- **Reasoning**: The term *reasoning* in LLMs pertains to their ability to logically derive conclusions and draw inferences using the data they have

been trained on. This skill is vital for generating coherent and meaningful text. Progress in NLP suggests that as these models increase in size, they might demonstrate enhanced reasoning abilities [27, 28]. Although the introduction of Chain of Thoughts (CoT) [29] has undeniably improved their reasoning skills through *step-by-step* thinking, the exact scope of this enhancement remains uncertain. For instance, they still struggle with knowledge-demanding tasks that pose no issues for humans [30].

- **Explainability**: A significant limitation of LLMs lies in their lack of explainability. These models, known for their complexity and black-box nature, often provide outputs without clear insights into how they reached their decisions or predictions. Achieving explainability in LLMs is a pressing challenge due to their intricate inner workings. The ability to offer human-understandable explanations for their outputs is crucial for building trust among users and ensuring alignment with ethical and human expectations.

Chapter 2

Related Work

2.1 Zero-shot NERC

As already anticipated (see Section 1.1), NERC is a fundamental NLP task that entails the identification and categorization of named entity mentions within textual data into predefined entity-types. These named entities encompass words or textual spans that represent tangible entities in the real world, encompassing individuals' names, organizational entities, geographical locations, temporal expressions, numerical values, and more.

A pervasive challenge encountered in numerous real-world applications is the scarcity of annotated data for specific entity classes, rendering traditional supervised learning approaches impractical. This motivates the focus on the zero-shot setting [31, 32], wherein annotated data for the classes of interest is unavailable. ZSL methodologies are pivotal in mitigating this data scarcity challenge by enabling the learning of models capable of transferring knowledge from *observed* classes in the training dataset to novel, previously *unseen* classes.

In this research endeavor, we delineate two primary avenues of investigation:

(i) augmenting the comprehension of target entity classes by providing supplementary contextual details.
(ii) harnessing the robust generalization capabilities of LLMs

Within the first realm, Aly, Vlachos, and McDonald [33] have illustrated the advantages of infusing manually crafted textual descriptions of entity classes

into the model's encoders. Concurrent studies have delved exclusively into specific aspects of NERC, such as named entity typing [34]. Meanwhile, Nguyen, Gelli, and Poria [35] have leveraged knowledge graphs to imbue the model with a nuanced understanding of the intricate interconnections between entities spanning diverse domains.

Moving to the second domain, it is essential to recognize that LLMs, due to the inherent distinction between sequence labeling and text generation, still lag behind supervised encoder-only SLMs in classification tasks [24]. Wang et al. [36] introduced a GPT-3 prompting strategy aimed at identifying entity mentions within the text without classifying them. In contrast, our approach tackles zero-shot NERC by integrating elements (i) and (ii), involving the training of SLMs on LLM-distilled datasets and enriched with artificially generated entity-type descriptions.

The subsequent paragraph will provide a detailed exploration of the utilization of entity-type descriptions to address the first pathway, while Section 2.2 will dive into the methodology employed to tackle the second.

2.1.1 Leveraging Entity-type Descriptions

Aly et al. [33] propose an innovative approach to address zero-shot NERC. This approach involves harnessing entity-type descriptions as additional contextual cues to aid the model in identifying previously unseen entity-types during the training phase. During the testing phase, the model is provided with two key inputs: a sentence and a set of target entity classes, each accompanied by a corresponding description. The primary objective is to recognize and classify entities within these target classes. The significance of these descriptions becomes evident when considering the example depicted in Figure 2.1.

To operationalize this concept, Aly et al. propose an architectural framework for NERC that employs cross-attention mechanisms between the input sentence and the entity-type descriptions, leveraging transformer models [14]. The implementation of cross-attention allows entities like *Shantou Harbour* to attend to relevant information within the type description of the *Facility* class, thereby generating representations that encapsulate the contextual relationship

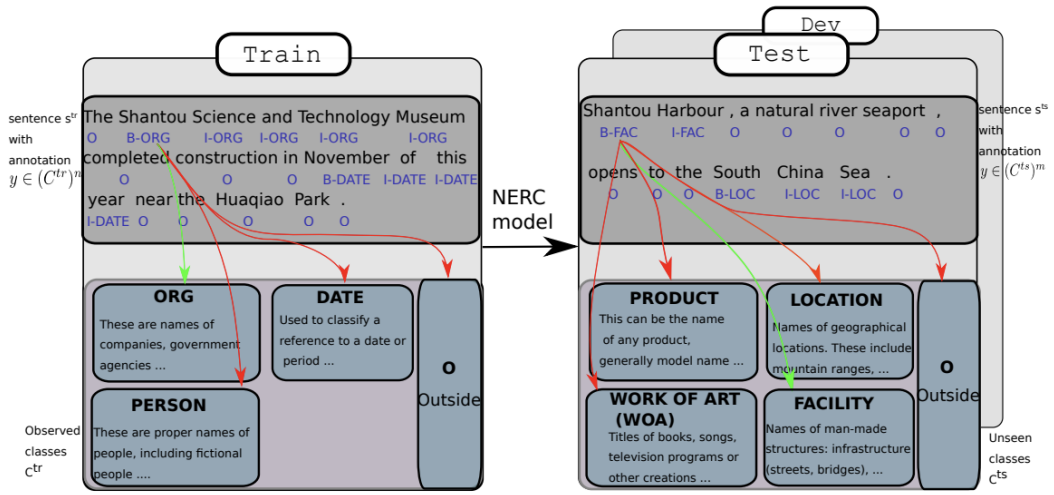


Figure 2.1: Zero-shot named entity recognition and classification. Source: [33].

When presented with the input sentence `Shantou Harbour, a natural river seaport, opens to the South China Sea` *and a target class* `Facility`*, and armed with a description like* `Names of human-made structures: infrastructure (streets, bridges), [...]`*, the model can establish a connection between the class* `Facility` *and the specific entity* `Shantou Harbour` *without relying on annotated examples encountered during training.*

between the entity and its associated description within the sentence.

For a more comprehensive understanding of this approach, further insights will be presented in Chapter 3.

2.2 Towards Affordable LLMs

Our primary goal is to develop a framework that enhances the performance of language models in performing NERC in a zero-shot setting, with a specific emphasis on domain-specific scenarios. As previously discussed, in the zero-shot approach, models must make the most of their existing knowledge to classify entities that were not encountered during the training phase, especially when dealing with specific and complex entities. LLMs can be particularly valuable in this context because they possess a wealth of knowledge that can

be leveraged to achieve this objective. One of the most significant challenges associated with LLMs is their high cost and resource-intensive training process. To address these deployment challenges, practitioners often opt for deploying smaller specialized models using one of two common paradigms: fine-tuning or knowledge distillation (KD).

The primary distinction between these approaches lies in the way the model is trained. Fine-tuning involves training our model on a human-labeled dataset, refining its parameters to adapt to a specific task. On the other hand, distillation involves training our model using a dataset generated by LLMs. Notably, fine-tuning has been evolving towards the direction of Parameter-Efficient Fine-Tuning (PEFT) [37], which involves training only a subset of the total model parameters compared to the full fine-tuning approach where all model weights are updated.

In this section, we will compare PEFT and KD to highlight their differences and implication.

2.2.1 PEFT

Full fine-tuning necessitates memory not only for model storage but also for housing various other parameters vital to the training phase. You must also be able to allocate memory for optimizer states, gradients, forward activations, and temporary memory throughout the training process (Table 2.1). These additional components can be many times larger than the model and can quickly become too large to handle on consumer hardware. For instance, to train a 1B parameter model at 32-bit full precision, you will need approximately 80 gigabyte of GPU RAM. In contrast to full fine-tuning, where every model weight is updated during supervised learning, parameter efficient fine-tuning methods only update a small subset of parameters. Some PEFT techniques freeze most of the model weights and focus on fine-tuning a subset of existing model parameters, for example, particular layers or components. This makes the memory requirements for training much more manageable. In fact, PEFT can often be performed on a single GPU. Furthermore, traditional full fine-tuning yields a distinct model version for each task, each with the same size

Component	Bytes per parameter
Model Parameters (weights)	+ 4 bytes
Adam Optimizer (2 states)	+ 8 bytes
Gradients	+ 4 bytes
Activation and Temporary Memory	+ 8 bytes
TOTAL	4 bytes + 20 extra bytes

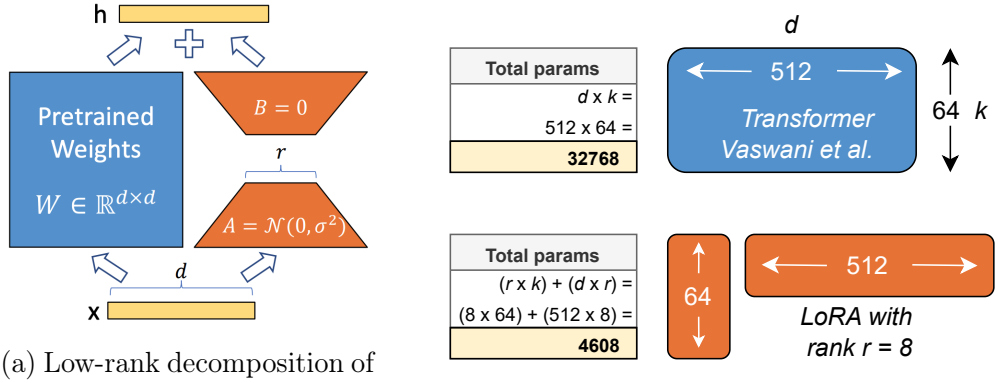
Table 2.1: Additional RAM GPU needed to train 1B parameters.

as the original model. This approach can lead to storage challenges when fine-tuning for multiple tasks.

Conversely, PEFT involves training lightweight task-specific weights (only a few megabytes in size) for each task. These task-specific weights can be easily swapped during inference, enabling seamless adaptation of the base model to multiple tasks while addressing storage concerns. Among various PEFT methods, Low-Rank Adaptation (LoRA) has gained significant popularity in recent research [38].

2.2.2 LoRA

LoRA is a technique that streamlines the fine-tuning process by reducing the number of parameters that need training. This is accomplished by preserving the original model's parameters as frozen while introducing a pair of rank decomposition matrices (Figure 2.2a). The dimensions of these compact matrices are set so that their product forms a matrix with the same dimensions as the weights they are intended to modify. The original weights of the LLM remain fixed, while the smaller matrices undergo training. During inference, the two low-rank matrices are multiplied together, yielding a matrix with the same dimensions as the frozen weights. This product is then added to the original weights, effectively updating them in the model. Consequently, a LoRA fine-tuned model is obtained, specialized for the desired task. Importantly, this fine-tuned model retains the same parameter count as the original model, ensuring minimal to negligible impact on inference latency. Figure 2.2b shows



(a) Low-rank decomposition of initial weights. Source: [38].

(b) LoRA impact on Transformer architecture.

Figure 2.2: Illustration of Low-Rank Adaptation and its impact.

(a) demonstrates the low-rank decomposition into trainable matrices A and B, while maintaining the initial pretrained weights frozen. (b) showcases LoRA's ability to achieve an impressive 86% reduction in trainable parameters compared to the standard Transformer architecture.

the actual impact of LoRA over transformer architecture described in *Attention is all you need* [14], leading to a reduction of 86% of trainable parameters.

2.2.3 QLoRA

In recent times, there has been a significant push in research towards making LLMs more accessible and efficient. This effort has led to the development of QLoRa [39], a groundbreaking advancement that enables users to run models with exceptional efficiency, utilizing 4-bit precision. QLoRA achieves this by employing 4-bit quantization to compress a pretrained language model. In this process, the parameters of the LLM are quantized to 4-bit precision, significantly reducing the memory footprint and computational requirements. These quantized parameters are then locked or frozen in place, preserving their efficiency gains. To adapt the model for specific tasks, QLoRA introduces a relatively small number of trainable parameters (a.k.a. Low-Rank Adapters, as discussed in Section 2.2.2). These adapters are added to the model and are designed to work in tandem with the quantized pretrained language model.

During fine-tuning, QLoRA efficiently backpropagates gradients through the frozen 4-bit quantized pretrained language model into the Low-Rank Adapters. This approach allows users to leverage the benefits of both quantization and fine-tuning, achieving remarkable efficiency while tailoring the model's capabilities to their desired tasks. However, it's crucial to consider certain practical implications of (Q)LoRA. During inference, (Q)LoRA mandates the complete loading of the entire model into memory, even when dealing with models that have billions of parameters. Additionally, as LoRA constitutes a form of fine-tuning, it relies on human-labeled data for implementation. As we have emphasized throughout our discussion, domain-specific scenarios often grapple with a scarcity of available labeled data. This constraint underscores the compelling choice of KD as the most natural and pragmatic solution for our specific case. KD enables us to harness existing knowledge, including the valuable insights encoded within LLMs, without the stringent requirement for extensive domain-specific labeled data.

2.2.4 Knowledge Distillation

KD is a general-purpose method for training a smaller student model to mimic the behavior of a slower, larger, but better-performing teacher [40, 41]. Originally introduced in 2006 in the context of ensemble models [42], it was later popularized in 2015 by Hinton et al. [41] that generalized the method to deep neural networks and applied it to image classification and automatic speech recognition. Given the trend toward pre-training language models with ever-increasing parameter counts, knowledge distillation has also become a popular strategy to compress these huge models and make them more suitable for building practical applications [43, 44, 45]. By transferring the knowledge from a powerful but computationally expensive teacher model to a smaller student model, knowledge distillation enables the student model to achieve competitive performance with significantly lower computational requirements.

The process is visually depicted in Figure 2.3. The fundamental concept behind KD involves enhancing the ground truth labels by incorporating a distribution of *soft probabilities* derived from the teacher model. These *soft*

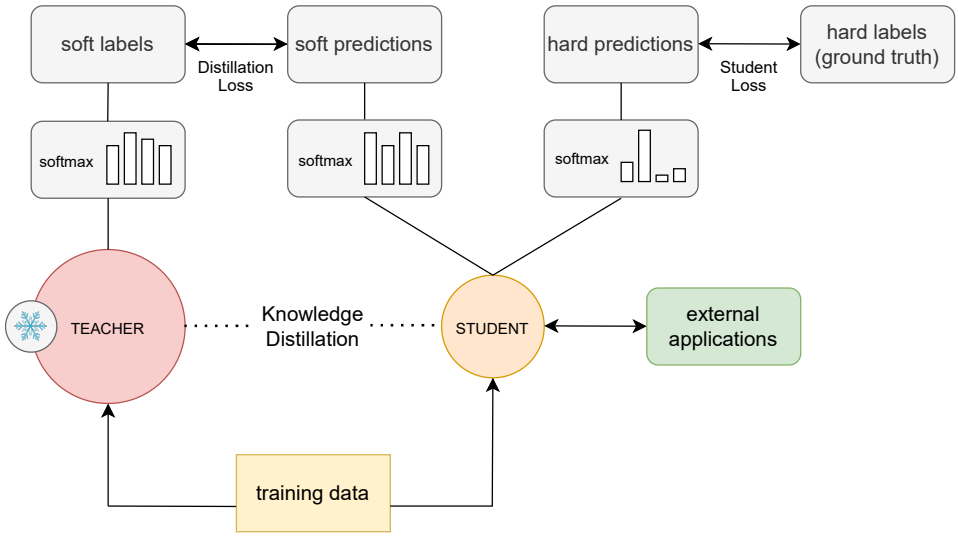


Figure 2.3: Overview of Knowledge Distillation process.

The knowledge distillation process entails a larger Teacher Model, fine-tuned on training data, guiding a smaller Student Model. The Teacher's fixed weights generate predictions for the training data, as does the student. The transfer of knowledge between the teacher and the student is based on minimizing the distillation loss, achieved by introducing a temperature *parameter t (typically $t > 1$) to broaden the softmax distribution (see Section 1.5.5). Simultaneously, the student refines its predictions using ground truth data without temperature adjustment. The student loss, which represents the gap between the student predictions and the ground truth, is computed. The combined distillation and student losses serve to update the student's weights via backpropagation.*

probabilities are essentially a reflection of the teacher's confidence in its predictions and can reveal nuanced relationships between different intents or classes. When the teacher assigns high probabilities to multiple intents, it suggests that these intents are not only relevant, but might also be close to each other in the feature space. By training the student model to mimic these *soft probabilities*, we aim to transfer some of the *dark knowledge* embedded in the teacher's predictions. This *dark knowledge* encompasses valuable insights, such as intra-class relationships or fine-grained distinctions, which go beyond what can be learned from the standard *hard labels* provided in the training data. In essence, KD

enables the student model to leverage the teacher's expertise, making it more adept at capturing intricate patterns. This specific variant of KD, in practice, is often more effective for encoder-only models like BERT, especially when the teacher model has a lot of representation redundancy. This is because encoder-only models tend to capture valuable semantic and contextual information in their representations, which can be effectively distilled into a smaller model.

2.2.5 Sequence-level Knowledge Distillation

For sequence-to-sequence or generative models, the concept of sequence-level distillation was introduced by Kim and Rush [46]. The student is trained on the teacher-generated data, which is the result of running beam search and taking the highest-scoring sequence with the teacher model (Figure 2.4). This approach involves generating a synthetic output by performing inference with the teacher model, which is then used to train the student model. Sequence-level distillation is efficient as it only requires running the typically large teacher model once. Prior studies have highlighted the success of sequence-level distillation. For instance, Costajussà et al. [47] effectively employed sequence-level distillation to shrink the size of NLLB, a machine translation system, to just 600 million parameters. Similarly, Wu et al. [48] utilized a sequence-level distillation strategy, training multiple decoder-decoder and decoder-only models on a massive dataset of 2.58 million examples generated by *gpt-3.5-turbo*. The results demonstrate performance comparable to Alpaca [49], despite the significantly smaller size of the models.

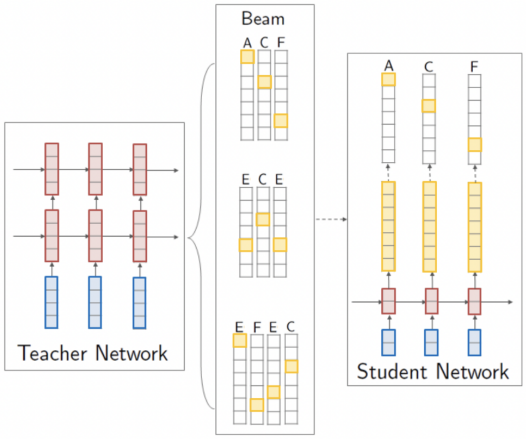


Figure 2.4: Sequence-level KD. *In sequence-level distillation, the student network is trained on the output from beam search of the teacher network that had the highest score.* Source: [46].

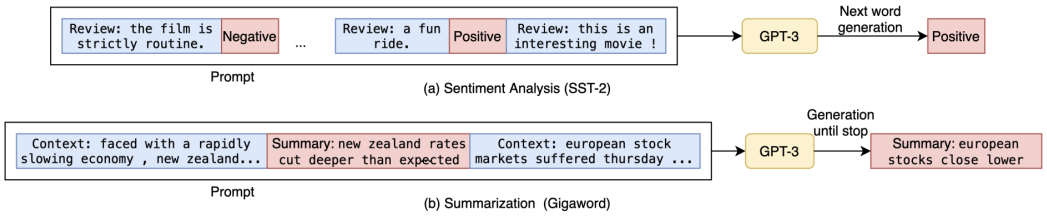


Figure 2.5: Two input prompt examples for generating annotated datasets with GPT-3. Source: [50].

The image demonstrates how GPT-3 can be leveraged to annotate an unsupervised dataset. Each prompt consists of n labeled data points for n-shot learning, along with the task input for label generation.

2.2.6 Machine-to-Corpus-to-Machine Paradigm

Recent publications have moved away from directly imitating teacher probabilities. Instead, similar to us, they adopt a *machine-to-corpus-to-machine paradigm*, wherein models are trained using substantial volumes of annotations generated by LLMs [4, 50, 51, 52, 53, 54]. The larger teacher model is used to generate a training dataset with noisy pseudo-labels, while the smaller student model is trained over the newly generated data. This approach leverages the power of LLMs to create annotations for *unlabeled data* (Figure 2.5), which are then utilized to train other models. These surrogate annotations, while not originating from human annotators, serve as valuable proxies for ground truth in scenarios where labeled data is scarce or absent. Thus, it is no surprise that this new paradigm has found resonance in our research work by bridging the gap of missing labeled data through the guidance of LLM-generated annotation. Moreover, the *machine-to-corpus-to-machine* paradigm is well-suited for harnessing the rich knowledge reservoirs possessed by LLMs. These models can distill their nuanced understanding of language and domain-specific intricacies into the generated annotations, imbuing the student model with a deeper and more sophisticated comprehension of the task at hand. The one limitation that this approach often faces is its reliance on large amounts of unlabeled data required to create a useful noisy training dataset.

2.3 The Pile

Recent advancements in the realm of general-purpose language modeling have highlighted the effectiveness of training massive models on extensive text corpora to enhance their utility in downstream applications [55, 56, 57]. Furthermore, research has underscored the pivotal role of augmented training dataset diversity in bolstering the overall cross-domain knowledge and downstream generalization capabilities of large-scale language models [57, 58, 59]. In response to this perspective, the Pile dataset has been introduced [60]. Comprising an impressive 825.18 GiB of English text data, the Pile dataset has been meticulously crafted to facilitate the training of large-scale language models. It amalgamates 22 diverse and high-quality datasets, encompassing both well-established NLP datasets and a collection of novel additions. In addition to its primary role in facilitating the training of LLMs, the Pile dataset assumes an equally significant role as a comprehensive benchmark for evaluating the cross-domain knowledge and generalization capabilities of language models. In the context of our current work, our focus has primarily centered on a specific subset of the Pile dataset: PubMed Abstracts. This subset comprises abstracts extracted from a staggering 30 million publications in PubMed, a prominent online repository for biomedical articles managed by the National Library of Medicine. Notably, PubMed also encompasses MEDLINE, extending its coverage to encompass biomedical abstracts dating back to 1946. As we will expound upon in Section 3.3.4, this dataset has proven to be of paramount importance in our distillation process. It not only encompasses entities pertinent to the biomedical domain but also encapsulates entities from various domain-specific contexts. This diverse representation enhances the adaptability of our zero-shot model to novel domains.

2.4 BLOOM

BLOOM is a large multilingual language model introduced by the BigScience project [61]. It represents a pioneering 176-billion-parameter open-access language model. It stands as a testament to the collaborative efforts of a global

community of over a thousand researchers hailing from +70 countries and +250 institutions. This colossal decoder-only Transformer model was meticulously trained on the ROOTS corpus, an extensive dataset encompassing a staggering 59 languages, including 46 natural languages and 13 programming languages. Notably, BLOOM's release marked a significant milestone, especially for languages like Spanish, French, and Arabic, where it became the first language model to breach the 100-billion-parameter threshold. The extensive pre-training phase spanned 117 days, reflecting the dedication and perseverance of the research community. BLOOM has demonstrated its mettle by achieving competitive performance across a diverse range of benchmarks, further solidifying its status as a pivotal asset in natural language understanding. In our research, we harnessed BLOOM as the teacher model in our knowledge distillation process, leveraging its expansive language coverage and formidable capabilities. This strategic choice allows us to effectively transfer its vast pre-trained knowledge to smaller models like BERT, thereby enhancing their performance in zero-shot NERC. In the upcoming chapters, we delve into the rationale behind selecting BLOOM over other LLMs and provide deeper insights into its advantages in our specific research context.

Chapter 3

Method

We propose that LLMs contain valuable prior knowledge, potentially reducing the need for extensive human-labeled data for training SLMs. Expanding on this concept, we present JUICER, a novel framework designed to bolster SLMs' zero-shot NERC capabilities through KD from LLMs (Figure 3.1). First, we leverage a frozen LLM to craft a comprehensive and diverse domain distillation dataset via example-guided data augmentation on raw corpora, bridging the gap left by manual annotation. To counteract the excessive confidence of the LLM and mitigate hallucination concerns, we propose multiple verification strategies to normalize pseudo-labels and filter out meaningless cases. Subsequently, we follow a two-stage KD pipeline in which a BERT-based student model is (i) trained to mimic the type-unbounded entity annotations decoded by the teacher LLM on the synthetic dataset and (ii) specialized on the downstream NERC domain.

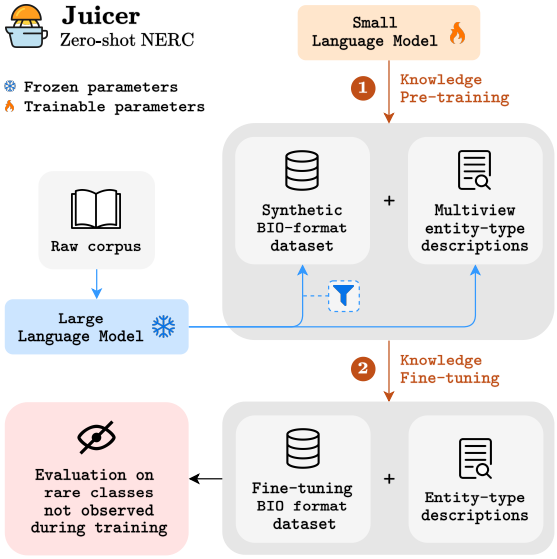


Figure 3.1: The JUICER framework.

To fulfill the ZSL constraints [12], we meticulously prevent any overlap between entity classes across training, validation, and test sets, reserving the rarest types for evaluation. Importantly, we incorporate textual descriptions of target entity-types as a supplementary input signal. Their representations are merged with the provided sentence through cross-attention, allowing SLMs to better disambiguate and make informed predictions about classes not seen during training. To this end, we further exploit LLMs to automatically generate entity-type descriptions with a multiview domain-aware technique. As anticipated during the introduction, we tested our method with the zero-shot-adapted versions of three widely used NERC datasets: MedMentions [9], OntoNotes 5.0 [10], and LegalNER [11]. We focus our distillation procedure on biomedicine, testing adaptability to the other domains.

3.1 Task Definition

We formulate NERC as a sequence labeling task. Given a sentence $\mathbf{s} = \{t_1, \ldots, t_n\}$ comprising n tokens and a description $\mathbf{d_c}$ for each entity class $c \in \mathbb{C}^{test}$ in the test set, we predict a sequence of annotations $\hat{\mathbf{y}} \in (\mathbb{C}^{test})^n$. The classification of a token at position i is determined by $\arg\max_{c \in \mathbb{C}^{test}} \mathcal{F}_\theta(\mathbf{s}, t_i, \mathbf{d_c})$, where $\mathcal{F}$ models the semantic affinity between t_i and $\mathbf{d_c}$ in the context of $\mathbf{s}$. Parameters θ must be acquired without labeled data for $\mathbb{C}^{test}$, but with labeled data and descriptions for training classes in $\mathbb{C}^{train}$. JUICER augments the training data with a wealth of LLM-generated examples; each input-output pair $<\tilde{\mathbf{s}}, \tilde{\mathbf{y}}>$ refers to a sentence $\tilde{\mathbf{s}}$ from an external corpus $\mathcal{R}$ and a target sequence $\tilde{\mathbf{y}}$ linked to a large volume of artificial classes $\tilde{\mathbb{C}}$ and their descriptions. Note that $\tilde{\mathbb{C}} \cap \mathbb{C}^{train} \cap \mathbb{C}^{dev} \cap \mathbb{C}^{test} = \varnothing$.

3.2 Teacher Model Selection

The first step of our work was to define the best suited LLM for our purposes. We tested four LLMs as potential knowledge distillation teacher networks:

- FLAN-T5-xxl, 11B, [62], T5 [22] encoder-decoder model fine-tuned on a

mixture of +1,000 tasks.[1]

- FLAN-UL2, 20B, encoder-decoder model based on the T5 architecture, trained on a collection of datasets via Flan prompting [63]. It is initialized with the UL2 checkpoint [64], which combines various pre-training paradigms together (Mixture-of-Denoisers).[2]
- BLOOM [61], 176B, open source GPT-3 [57], autoregressive decoder-only model for next token prediction trained on 46 different languages and 13 programming languages (Section 2.4).[3]
- BLOOMZ [65], 176B, BLOOM model fine-tuned on crosslingual task mixture.[4]

We contrasted their performances in zero-shot NERC proficiency and term description generation. Testing LLMs in entity-mention descriptions rather than entity-class ones was a strategic decision to precisely gauge their effectiveness. Given that the mentions provide a richer and more detailed perspective than class-oriented assessments, this strategy enabled us to capture the intricacies and variations within the data. BLOOM surfaced as the sole LLM demonstrating precise adherence to multi-task prompts (Table 3.1) while crafting rich term descriptions (Table 3.2). Table 3.3, instead, shows all the experimented prompts for each task.

3.3 Synthetic BIO-format Dataset

The following paragraphs break down the LLM-to-dataset workflow, visually summarized in Figure 3.2. Orienting our focus to biomedicine, we set $\mathcal{R}$ by randomly sampling 50K instances[5] from the PubMed Abstracts subdataset of the Pile (Section 2.3), which includes the abstracts of PubMed/MEDLINE publications from 1946 to the present day (English language). Although JUICER

[1] `https://huggingface.co/google/flan-t5-xxl`

[2] `https://huggingface.co/google/flan-ul2`

[3] `https://huggingface.co/bigscience/bloom`

[4] `https://huggingface.co/bigscience/bloomz`

[5] A number of passages known to be sufficient for satisfactory NERC distillation [2].

Prompt	Flan-T5	Bloom
List me the entities and the corresponding class within the following sentence: Ledipasvir-sofosbuvir and sofosbuvir plus ribavirin in patients with chronic hepatitis C and bleeding disorders. Please, use the following output format as example: entity_1 (Class_1), entity_2 (Class_2), ...	ledipasvir-sofosbuvir, sofosbuvir, ribavirin	Ledipasvir-sofosbuvir (Drug), ribavirin (Drug), chronic hepatitis C (Medical condition), bleeding disorders (Medical condition)
Context: The novel *To Kill a Mockingbird* by Harper Lee is a classic of American literature, telling the story of a young girl named Scout growing up in the 1930s in the fictional town of Maycomb, Alabama. Domain: literature Named Entities: To Kill a Mockingbird (Literary work), Harper Lee (Author), American literature (Literary genre), young girl (Character), Scout (Character), 1930s (Historical period), fictional town (Geographic feature), Maycomb (Geographic feature), Alabama (Geographic feature) Context: The Hubble Space Telescope is a powerful tool used by astronomers to observe the universe from Earth's orbit, allowing for high-resolution images of distant galaxies and other celestial objects. Domain: astronomy and astrophysics Named Entities: Hubble Space Telescope (Scientific instrument), astronomers (Occupation), universe (Astronomy), Earth's orbit (Astrophysics), high-resolution images (Technology), distant galaxies (Astronomy), celestial objects (Astronomy) Context: Ledipasvir-sofosbuvir and sofosbuvir plus ribavirin in patients with chronic hepatitis C and bleeding disorders Domain:	Named Entity: Hepatitis C Named Entity: Hepatitis C virus (HCV) Named Entity: Hepatitis C (Virus) Named Entity: Ledipasvir-sofosbuvir, cirrhotic, genotype 1 Named Entity: hepatitis C, Treatment, Sofosbuvir, Rivabrin	Domain: medicine Named Entities: Ledipasvir-sofosbuvir (Drug), ribavirin (Drug), chronic hepatitis C (Medical condition), bleeding disorders (Medical condition)

Table 3.1: Large Language Model comparison on prompt completion.

While Flan-T5 *does not exhibit the capability to simultaneously handle multiple tasks, such as named entity recognition, named entity classification, and domain classification, BLOOM demonstrates proficiency in handling both two tasks and three tasks concurrently.*

Term	Flan-T5	Flan-UL2	Bloom	BloomZ
`headache`	a severe pain in the head	a pounding pain in the head	It is a common and complex disorder that is associated with significant disability, reduced quality of life, and high economic burden.	headache is a common symptom
`intranasal`	The passage of an object or substance into the nose.	Breathing in an object or substance through the nose.	Intranasal is a term that describes a drug that is administered into the nasal cavity. It is often used to describe a nasal spray, nasal drop, or nasal gel.	The drug is administrated intranasally, meaning it is sprayed into the nose.
`oxytocin`	hormone that is released during sex	The hormone oxytocin is produced by the pituitary gland.	Oxytocin is a neuropeptide hormone that is synthesized in the hypothalamus and released into the bloodstream. It is also produced in the brain, uterus, and placenta.	The human body produces a hormone called oxytocin. The hormone is released during orgasm.
`trigeminal ganglia`	The trigeminal ganglia is a group of three ganglia, each located in the cranium.	There are three ganglia in the skull known as the trigeminal ganglia.	The trigeminal ganglion (TG) is a collection of nerve cell bodies located in the trigeminal ganglion fossa, which is a small dural depression on the lateral aspect of the middle cranial fossa.	A ganglion is a collection of nerves. The trigeminal ganglia are the collection of nerves that control the facial muscles.

Prompt: `Question: In {{domain}}, what is {{term}}? Answer:`

Table 3.2: Large Language Model comparison on term description generation.

ID	Prompt
1	***Named Entity Recognition***
1.1	Context: {{sentence}} List me all the entities:
1.2	Context: {{sentence}} Entities:
1.3	Context: {{sentence}} List me all the terms:
1.4	Context: {{sentence}} Terms:
1.5	Which are the entities in the following sentence? {{sentence}}
1.6	Which are the terms in the following sentence? {{sentence}}
1.7	Which are the most relevant entities in the following sentence? {{sentence}}
1.8	Which are the most relevant terms in the following sentence? {{sentence}}
1.9	Question: Which are the relevant terms inside the following sentence? {{sentence}} Answer:
2	***Named Entity Classification***
2.1	Context: {{sentence}} Term: {{term}} Category:
2.2	Term: {{term}} Category:
2.3	Term: {{term}} Class:
2.4	Context: {{sentence}} Question: What is the category of the following term? {{term}} Answer:
2.5	Question: What is the category of the following term? {{term}} Answer:
2.6	Question: What is the class of the following term? {{term}} Answer:
2.7	Context: {{sentence}} Question: To what category {{term}} belongs to? Answer:
2.8	Question: To what category {{term}} belongs to? Answer:
2.9	Question: To what class {{term}} belongs to? Answer:
3	***Multi-Task***
3.1	List me the entities and the corresponding class within the following sentence:{{sentence}}. Please, use the following output format as example: entity_1 (Class_1), entity_2 (Class_2), ...
3.2 *	Context: The novel "To Kill a Mockingbird" by Harper Lee is a classic of American literature, telling the story of a young girl named Scout growing up in the 1930s in the fictional town of Maycomb, Alabama. Domain: literature Named Entities: To Kill a Mockingbird (Literary work), Harper Lee (Author), American literature (Literary genre), young girl (Character), Scout (Character), 1930s (Historical period), fictional town (Geographic feature), Maycomb (Geographic feature), Alabama (Geographic feature) Context: The Hubble Space Telescope is a powerful tool used by astronomers to observe the universe from Earth's orbit, allowing for high-resolution images of distant galaxies and other celestial objects. Domain: astronomy and astrophysics Named Entities: Hubble Space Telescope (Scientific instrument), astronomers (Occupation), universe (Astronomy), Earth's orbit (Astrophysics), high-resolution images (Technology), distant galaxies (Astronomy), celestial objects (Astronomy) Context: {{sentence}} Domain:
3.2.N	Different number of BLOOM-generated in-context learning examples (i.e., $k = 1, 3, 4, 5$).

Table 3.3: Experimented prompts during Teacher Model selection.

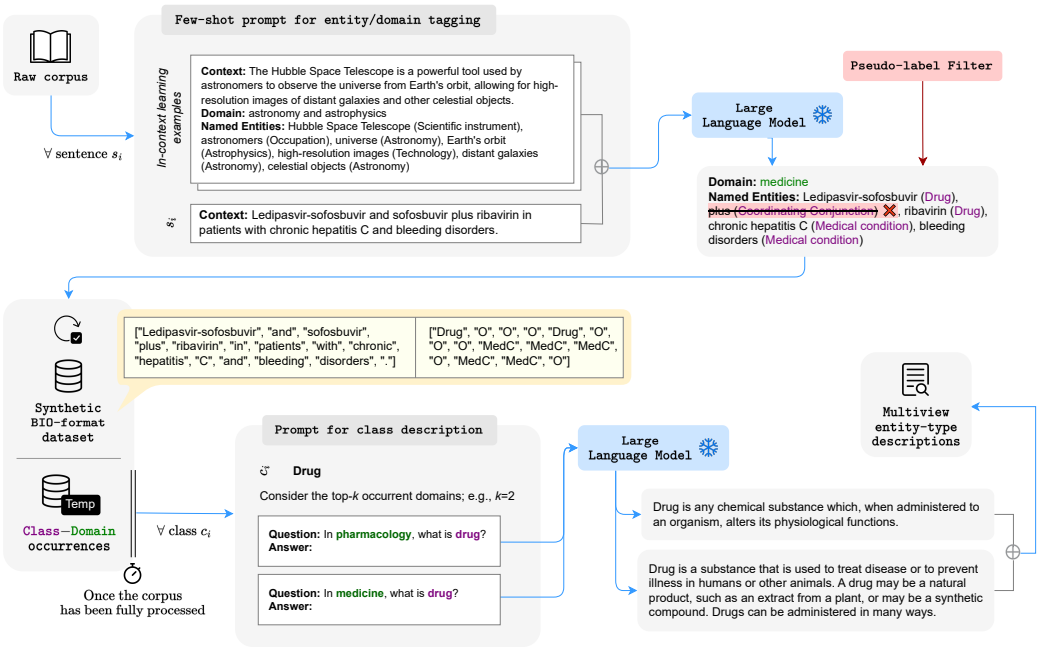


Figure 3.2: JUICER pipeline for dataset construction.

The image illustrates the JUICER *pipeline for constructing the transfer dataset (i.e., silver entity labels and type descriptions). Reported examples are from the Pile PubMed Abstracts. The white boxes denote LLM prompts.* ⊕ *stands for string concatenation.*

is model-agnostic, we select BLOOM-176B as the LLM teacher.[6]

3.3.1 Prompt-based Tagging

We approximate expert labelers by few-shot prompting [57] on an off-the-shelf LLM. We run sentence segmentation on top of $\mathcal{R}$, resulting in a total of ≈600K transfer sentences. Then, we curate a prompt template to simultaneously operate (i) entity recognition, (ii) entity classification, and (iii) domain

[6]Please note that at the time of building the transfer dataset, BLOOM had the distinction of being the largest Open LLM available and one of the top-notch LLMs in general (FALCON and LLAMA were not yet published). With 1.6TB of preprocessed text, BLOOM was also one of the LLMs trained with more data, making it perfect for this work [66, 67]. We also tested smaller LLMs to contrast the results and preferred BLOOM for its ability to handle multi-task prompts and produce more accurate descriptions.

extraction (see Table 3.1). The prompt starts with k packed demonstrations, which serve to regulate the output format—vital for parsing—and provide direct evidence about the task. In light of rigorous prompt engineering, **Context: [*sentence*]**\n **Domain: [*keyword*]**\n **Named Entities: [*mention (label)*]** emerges as our preferred choice. After the demonstration, an incomplete instance is appended, asking the LLM to infer missing domain and entity tags. We shape our data under an open-world assumption, liberating the LLM from predefined entity-types, thus maximizing $\tilde{\mathbb{C}}$ coverage. Drawing from empirical findings, we set $k = 2$; we maintain zero-shot constraints by incorporating general off-target examples from BLOOM itself.

3.3.2 Filtering

We devise a series of *critic rules* to control the accuracy and confidence of the knowledge being transferred:

- We exclude sentences accompanied by improperly formatted generations or without any positive annotations—1.44% of the total.
- We filter out recognized entities not mentioned in the input sentence.
- We remove entities with a mention solely composed of stopwords, verbs, conjunctions, or prepositions.
- We use lemmatization to tackle the complexity arising from entity mentions having different labels, such as synonyms or inflected forms. For example, *patients*, *male patient*, and *old patients* can all be assigned to the root *patient*. To contain distillation classes and mitigate inconsistencies, we normalize the predicted labels with the most frequent class for each lemma at the corpus level; in the event of a tie for rank 1, the decision is made randomly.
- We discard irrelevant entity classes with only one occurrence (e.g., *21*, ®, *+/*), ≈500 cases.

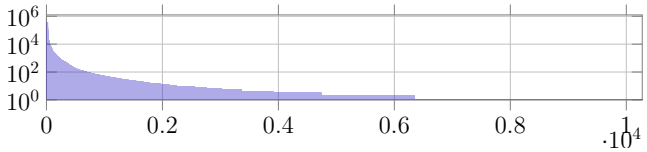
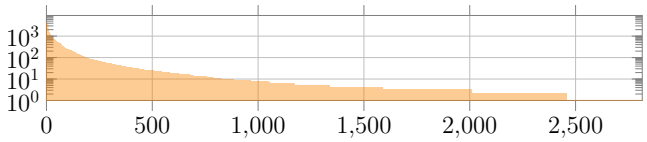
3.3.3 BIO-format Instances

We parse the filtered LLM output and represent named entities in BIO format (see Section 1.1). Since the LLM thrives on free generation, the derived

annotations are not accompanied by offset indices communicating their token position in the original sentence. When a predicted entity mention repeatedly occurs in the input, all mention tokens receive the same class. Any predicted entity-type overlapped with $\mathbb{C}^{train}$, $\mathbb{C}^{dev}$, or $\mathbb{C}^{test}$ is overwritten with the negative class (O).

3.3.4 Data Statistics

,

Property	Distributions
# sentences 591,363	**entity-type distribution** 
# entities 2,063,280	
# distinct entity-types 10,273	
# distinct mention–type links 806,525	**domain distribution** 
# distinct domains 2,813	

Frequency	Entity-types
Top 1% (82%)	Biological process, Medical condition, Medical procedure, Occupation, Chemical compound, Disease, Protein ...
2%–10% (15%)	Environmental hazard, Sleep disorder, Book, Edition, Ability, Public health intervention, Drink ...
11%–100% (3%)	Plant process, Anatomical entity, Biomedical field, Health care technology, ECG finding, Habit, Political science ...
Frequency	**Domains**
Top 1% (33%)	medicine, biochemistry, psychology, oncology, neuroscience, microbiology, pharmacology ...
2%–10% (48%)	ecology, virology, molecular biology, nursing, orthopedics, pediatrics, obstetrics and gynecology ...
11%–100% (19%)	pharmacovigilance, medicinal chemistry, human computer interaction, social media, food and drink ...

Table 3.4: Pseudo-label statistics.

Table 3.4 analyzes our transfer BIO-format dataset. The distributions of entity-types and domains exhibit a pronounced heavy-tailed pattern, where the top 1% accounts for 82% and 33% of the cumulative frequencies, respectively. Although primarily anchored in biomedicine, we unveil a broad spectrum of classes that span various disciplines. Among these, are politics, law, finance, literature, history, music, and math. A noteworthy observation is the existence

of granularity variations within particular entity-types, e.g., *Protein* and *Gene* are subsets of *Biological entity*. These attributes greatly enhance the dataset's potential to universally capture LLM capabilities.

3.4 Entity-type Descriptions

As extra guidance during $\mathcal{F}$ training, descriptions establish a more explicit connection between t_i and the expected class y_i. For instance, given a sentence *Pantoprazole is commonly prescribed to reduce stomach acid production* and a class *Drug*, the description *Drug is any chemical substance which, when administered to an organism, alters its physiological functions* can help the model to make the clue, even without previous exposure to training examples. Plural domains could offer separate definitions for a specific class, granting more opportunities to absorb LLM knowledge. For this reason, we devise a prompting technique oriented to multi-view descriptions. Formerly collected domain labels are used to count the predominant fields in which each entity-type finds mention. For every class $\tilde{\mathbf{y}}$, we request the LLM to generate individual descriptions linked to the two leading domains (Figure 3.2). Our prompt template is **Question: In [*domain*], what is [*class*]? Answer:**. Ultimately, the outputs are concatenated following their ranking sequence. The average number of tokens per multi-view description is 45.

3.5 Student Network

Our SLM is based on pre-trained BERT-large [13]. JUICER's $\mathcal{F}$ modeling parallels the cross-attention encoder (X-ENC) presented by Aly et al. [33]. For notational convenience, we use $\mathbb{C}$ to refer to target classes, regardless of whether they come from a gold dataset or are synthetically generated. With every $\mathbf{d}_c$, X-ENC produces a vector representation $\mathbf{v}_{i,c} \in \mathbb{R}^h$ for each token t_i in sentence $\mathbf{s}$:

$$\mathbf{v}_{1,c}, \ldots, \mathbf{v}_{n,c} = \text{X-ENC}(\mathbf{s}, \mathbf{d}_c), \tag{3.1}$$

where $h = 1024$ and the input tuple $(\mathbf{s}, \mathbf{d}_c)$ is structured in the form `[CLS]` $\mathbf{s}$ `[SEP]` $\mathbf{d}_c$. The vector $\mathbf{v}_{i,c}$ is transformed through a learnable linear layer to quantify the likelihood $l_{i,c} \in \mathbb{R}$ that the token t_i belongs to the entity class c:

$$l_{i,c} = \mathbf{v}_{i,c} \cdot \omega^T + b. \tag{3.2}$$

In the quest to identify entities other than classifying them, we append the token scores with the likelihood of belonging to the negative class c_{neg}:

$$l_i = (l_{i,c_1}; \ldots; l_{i,c_{|\mathbb{C}|}}; l_{i,c_{neg}}). \tag{3.3}$$

To derive c_{neg}, we use the same class-aware encoding as in [33]. After undergoing Softmax, the top-scoring class is chosen.

3.6 Training

The JUICER distillation process unfolds in two stages, in which the model is first trained on the vast LLM-generated data and then specialized with a few supervised data. We call these steps *Knowledge Pre-training* and *Knowledge Fine-tuning*, shortened KP and KF. In both cases, we manage the label imbalance induced by c_{neg} by incorporating class weights q_c into the cross-entropy loss:

$$\mathcal{L} = -\sum_{c}^{\mathbb{C}} q_c \cdot y_{i,c} \cdot log(p(\hat{y}_{i,c})), \tag{3.4}$$

where $y_{i,c}$ is the truth label and $p(y_{i,c})$ is the Softmax probability for class c. We consistently set q to 1 for positive classes, while for the negative class, we fine-tune it as a hyperparameter, taking the ratio $\frac{\#\text{ entities}}{\#\text{ non-entity words}}$ in the training data as a reference factor. Given the small number of consecutive entities belonging to the same class ($\approx 1.22\%$ on average across all datasets), we remove all I- and B- prefixes, resulting in a single label per entity class. We employ two regularization techniques. First, during training, we selectively obscure the entire entity to be classified from the input $\mathbf{s}$ with a probability m_p tuned as a hyperparameter. Entity masking prevents X-ENC from overfitting and stimulates the model to internalize the entity context for affinities with class descriptions. Second, at each KP step, we only treat a random assortment

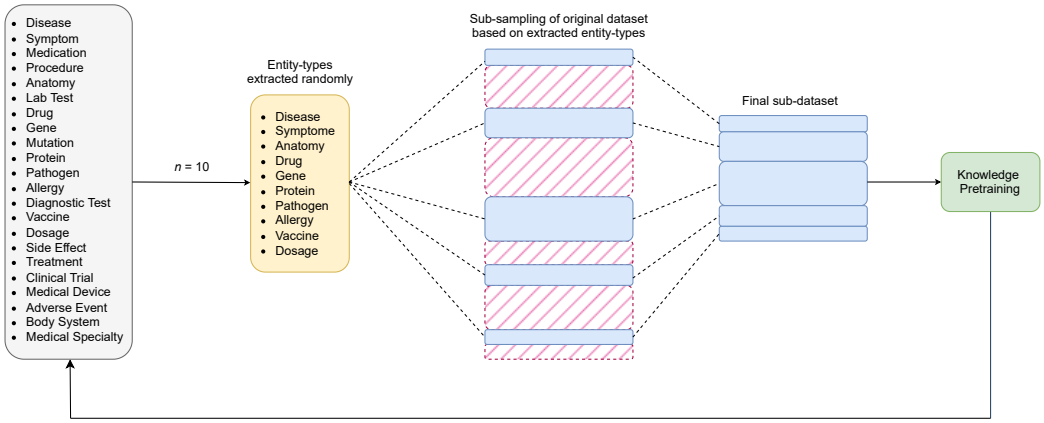


Figure 3.3: Random entity-type sampling during Knowledge Pre-training.

In each iteration of the pre-training process, a randomly selected subset of entity-types, typically ranging between 10 and 15 (here n = 10), is chosen. Subsequently, the dataset undergoes a filtering process, wherein only the data points associated with at least one entity-type from the selected subset are retained. The model is trained over the new subset and this iterative process is repeated, with each iteration involving the selection of a fresh set of entity-types. The primary objective of this approach is to train the model incrementally by focusing on smaller, manageable subsets of entity-types in each iteration. By doing so, we ensure that the model acquires expertise in a phased manner, gradually expanding its knowledge base over multiple iterations.

of entity-types, ranging between 10 and 15 (Figure 3.3). Subsequently, we filter the original dataset to retain only those data points associated with at least one of the extracted entity-types. If this results in an excess of data points, we set an upper limit of 50k samples, selecting them randomly in such cases. Any other positive tags outside the extracted entity-types are reclassified as negative ('O'). Iteratively exposing the SLM to different subsets of entity-types contributes positively to the target and domain adaptability, aligning well with the zero-shot NERC mission.

Chapter 4

Experimental Setup

4.1 Datasets

To gauge JUICER holistically, we conduct experiments on three heterogeneous English NERC datasets: MedMentions [9], OntoNotes 5.0 [10], and LegalNER [11]. MedMentions *(CC0 license)* holds 4K biomedical abstracts and a fine-grained class ontology rooted in the Unified Medical Language System (UMLS). OntoNotes 5.0 *(LDC license)* is a large-scale multigenre corpus targeting domain-general entity-types, including values. LegalNER *(MIT license)* annotates Indian court judgments published between 1950 and 2022. Building on the recommendations of Xian et al. [12], we create zero-shot adapted versions of these benchmarks (identified by the *-ZS* suffix). In pursuit of this, we adhere to the conversion protocol suggested by Aly et al. [33], alternatively leaving the rarest classes in the dev and test sets. The chosen types are guaranteed to be not trivial to recognize; we omit classes whose surface form displays regular patterns (e.g., *Percent*, *Date*). Annotations on split-disconnected classes are eliminated. An overview of the data distribution is shown in Table 4.1 and Figure 4.1. Regarding class description sources, we draw on the UMLS Metathesaurus for MedMentions and annotation guidelines for OntoNotes and LegalNER.

	MedMentions-ZS			OntoNotes-ZS			LegalNER-ZS		
	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test
# sentences	26,770	1,289	1,048	41,475	1,358	387	6,615	1,150	1,450
# words ♠	701,070	37,297	29,783	1,072,041	39,349	11,145	469,231	43,258	51,973
# entities	113,158	1,710	1,430	86,927	1,735	480	16,175	1,445	1,998
# compound entities	51,264	837	705	51,481	516	469	15,528	1,237	1,987
# consecutive entities of same class	2,787	5	14	339	11	0	154	7	6

♠ Yielded with NLTK [68].

Table 4.1: Comparative overview of zero-shot datasets.

Figure 4.1: Class occurrences in {MedMentions/OntoNotes/LegalNER}-ZS.

4.2 Metrics

We address the challenge of sample imbalance between classes by employing per-class averaged scores. Thus, our model evaluations include macro-averaged precision, recall, and F_1 metrics, assessed at both token and span levels. At the token level, we evaluate the model's ability to correctly classify individual tokens within the text, allowing us to assess fine-grained performance across all tokens. Conversely, at the span level, we assess the model's capability to correctly identify and classify entire spans or sequences of tokens that represent named entities or specific linguistic structures. Span-level evaluation provides insights into the model's ability to capture broader context and semantic meaning within the text. This dual evaluation ensures a comprehensive assessment of model performance, accounting for variations in class distribution and providing a balanced view of its effectiveness across different levels of granularity.

4.3 Baselines

We head-to-head compare JUICER with SLMs and LLMs, considering two scenarios: fine-tuned on the benchmark dataset's training set and left in a pure zero-shot configuration. We begin by comparing JUICER to SMXM [33]. SMXM is based on the BERT-large model, comparable to JUICER without KP. Due to inconsistencies, we re-compute all metrics. There are three primary motivations for this: (i) After re-building non-public SMXM -ZS datasets, we register discrepancies in entity occurrences. (ii) There is no possibility of verifying the actual descriptions used by the original work. (iii) The authors employ different hyperparameters depending on the dataset, including batch size values exceeding our hardware capabilities. This re-evaluation addresses these concerns and allows for a more accurate comparison.

On the other hand, as extensively discussed in Section 2.2, PEFT emerges as a compelling alternative to KD. Notably, recent work by Liu et al. [69] underscores the superior cost-effectiveness of PEFT compared to ICL. While PEFT incurs higher inference costs than JUICER, the potential for low-rank adaptation of a more potent LLM suggests the possibility of achieving significantly enhanced results. In a practical scenario, we contemplate the utilization

of smaller LLM versions, such as the 7B variant of BLOOM, LLaMA 2, and Falcon. This choice allows us to replicate real-world contexts while harnessing the efficiency gains offered by QLoRA. However, for the sake of a comprehensive analysis, we also consider comparing our JUICER model against the largest versions of BLOOM (176B), LLaMA 2 (70B), and Falcon (40B) in the context of one-shot ICL.[12] Table 4.2 illustrates the prompts adopted for PEFT and ICL processes with the three LLMs. During Juicer KP, we select the latest checkpoint for each *unseen* dev/test evaluation split because the absence of gold examples prevents us from measuring improvements in a specific domain.

4.4 Implementation Details

We run each experiment on a cluster of OS Linux workstations, using multiple Nvidia Tesla v100 GPUs with 32GB of dedicated memory. Juicer is developed using PyTorch 1.9. We execute 30 KP steps, 4 epochs each. A new set of target classes is extracted at the beginning of each KP step. KP completion requires 6 days of computation (100K daily processed sentences) with 8 GPU and a total of 400GB VRAM. KF is performed up to 3 epochs, with early stopping activated.[3] On average, fine-tuning a Juicer model takes 8 hours on a single GPU and 20GB VRAM occupation. In both KP and KF, we use 2 as batch size and 0.7 as m_p; we leave 42 as the default training seed.

4.5 Hyperparameters

For the sake of replicability, complete hyperparameters are provided. Table 4.3 lists all hyperparameters examined in the inference and fine-tuning phases, highlighting the final values. The parameters of the linear classification layer ω^T have been randomly initialized from a uniform distribution $U(-\sqrt{b}, \sqrt{b})$ with $b = \frac{1}{\text{in-features}}$. Span-level scores are computed with the `seqeval` library.[4]

[1] During our experiments, Falcon's 40B version was the largest available; it's worth noting that the 180B version is now the largest available.

[2] Throughout this discussion, we refer to zero-shot LLMs in the context of one-shot ICL.

[3] We find these training settings sufficient for adequately exemplifying Juicer behavior.

[4] `https://github.com/chakki-works/seqeval`

1	***Zero-Shot Large Language Model Baselines***

```
### Instruction:
From the input sentence extract instances of the following labels
[FAC, LOC, WORK_OF_ART].
Return the output in the following format:
Entity 1 (label1), entity 2 (label2) ...

### Input:  In recent years, advanced education for professionals
has become a hot topic in the business community.
### Named Entities:  recent years (DATE)

###Input:  {{sentence}}
### Named Entities:
```

2	***QLORA Fine-tuned Large Language Model Baselines*** (Train set example for MedMentions-ZS)

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

### Instruction:  From the input context below extract instances
of the following labels:  'Biologic_Function', 'Chemical',
'Health_Care_Activity', 'Anotomical_Structure', 'Finding',
'Spatial_Concept', 'Intellectual_Product', 'Research_Activity',
'Eukaryote', 'Population_Group', 'Medical_Device'

### Input:
DCTN4 as a modifier of chronic Pseudomonas aeruginosa infection in
cystic fibrosis

### Response:
DCTN4 (Chemical), chronic Pseudomonas aeruginosa infection
(Biologic_Function), cystic fibrosis (Biologic_Function)
```

Table 4.2: Prompts used in PEFT and ICL processes for 7B and largest versions of BLOOM, LLaMA 2, and Falcon, respectively.

(1) presents the prompt employed for zero-shot inference using the larger variants of LLMs. Specifically, we utilize a single example extracted from the training dataset to steer the LLM's completion. (2) illustrates the formatted prompt for fine-tuning the 7B models on each respective dataset. In this instance, we provide an example from the training set used for fine-tuning on MedMentions-ZS.

Hyperparameter	**Search space**
BLOOM-176B inference — entities and domains	
decoding_method	*sample*
max_new_tokens	{50∗, 70, 100}
min_new_tokens	1
stream	false
temperature	{0.1∗, 0.5, 0.7, 0.8}
top_k	{1, 3, 5, 7, 10∗, 20}
top_p	{0.7, 0.8, 0.9, 1∗}
BLOOM-176B inference — descriptions	
decoding_method	*sample*
max_new_tokens	{25, 50∗, 70, 100, 150}
min_new_tokens	1
stream	false
temperature	{0.1, 0.5∗, 0.6, 0.7, 0.8, 0.9}
top_k	{1, 3, 5∗, 7, 10, 20}
top_p	{0.7, 0.8, 0.9, 1∗}
BERT-large knowledge pre-training	
min_sample_class	10
max_sample_class	1
max_description_length	150
max_sequence_length	300
mask_probability	0.7
num_pretrain_steps	30
linear_dropout	0.5
linear_units_symbol	100
adam_epsilon	1e-8
max_grad_norm	1.0
default	4e-6
epochs	4
batch	2
val_steps	1
BERT-large knowledge fine-tuning	
### MedMentions-ZS and LegalNER-ZS	
max_description_length	100
max_sequence_length	200
### OntoNotes-ZS	
max_description_length	150
max_sequence_length	300
### All	
mask_probability	{0.3, 0.7∗}
linear_dropout	0.5
linear_units_symbol	100
adam_epsilon	1e-8
max_grad_norm	1.0
lr	4e-6
epochs	3
batch	2
val_steps	1

Table 4.3: Explored hyperparameters along with their empirical search grid.

∗ *marks the final picked values.*

Chapter 5

Results and Discussion

5.1 Analysis of Results

Our core results are presented in Table 5.1a and Table 5.1b, showcasing one run per experiment for the test and dev sets, respectively. To prevent redundancy, our result discussion in this section focuses primarily on the test set. However, it's worth noting that the dev set results (Table 5.1b) are fairly similar, therefore we can safely assume that our models have a reasonable generalisation power. For a more granular per-class analysis, we refer readers to Figure 5.1, which provides a detailed breakdown of the results for test set.

In the tables, the term *zero-shot* represents SML without the KF phase, while it denotes LLMs that utilize ICL at inference time instead of PEFT. For clarity, all results discussed in this section will refer to *token-level* evaluation, which is the most common approach in NER, as it provides a finer-grained analysis of model performance.

5.1.1 Juicer vs SMXM

As expected, the zero-shot SMXM, which has not undergone fine-tuning specifically for zero-shot learning, falls short due to its lack of examples for the downstream task. In contrast, the zero-shot Juicer model shines brightly, outperforming the zero-shot SMXM across all three datasets: achieving a +0.21 macro-F_1 improvement on MedMentions-ZS, +0.29 on OntoNotes-ZS,

(a) Test set results

		MedMentions-ZS ♠						OntoNotes-ZS						LegalNER-ZS						Avg		
		T			*S*			*T*			*S*			*T*			*S*			*T*	*S*	
Large Language Models	**Size** ◇	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	F_1	F_1	
Zero-shot																						
BLOOM	176B	0.28	0.26	0.14	0.11	0.26	0.11	0.29	0.23	0.22	0.16	0.18	0.14	0.13	0.32	0.17	0.09	0.27	0.13	0.18	0.13	--
FALCON	40B	0.12	0.53	0.15	0.09	**0.53**	0.12	0.12	0.37	0.17	0.07	0.19	0.09	0.03	**0.68**	0.05	0.03	**0.67**	0.04	0.12	0.08	--
LLAMA 2	70B	0.21	**0.61**	0.30	0.17	**0.53**	0.25	0.67	0.09	0.13	0.06	0.58	0.10	0.24	0.21	0.19	0.17	0.19	0.17	0.21	0.17	--
Fine-tuned (QLORA)																						
BLOOM	7B	0.42	0.27	0.30	0.22	0.35	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.24	0.04	0.07	0.04	0.25	0.07	0.12	0.11	3.12
FALCON	7B	0.41	0.21	0.25	0.16	0.34	0.20	0.00	0.00	0.00	0.00	0.00	0.00	0.30	0.02	0.03	0.02	0.29	0.03	0.09	0.08	3.13
LLAMA 2	7B	0.49	0.35	**0.39**	0.28	0.42	**0.33**	**0.97**	0.07	0.11	0.06	**0.90**	0.09	**0.51**	0.41	0.39	0.37	0.46	**0.38**	0.30	0.27	3.10
		T			*S*			*T*			*S*			*T*			*S*			*T*	*S*	
Small Language Models	**Size**	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	F_1	F_1	
Zero-shot																						
SMXM	345M	0.09	0.02	0.03	0.05	0.01	0.01	0.06	0.04	0.05	0.01	0.00	0.00	0.10	0.02	0.04	0.04	0.00	0.01	0.04	0.01	0.02
JUICER **(Ours)** · KP	345M	0.41	0.20	0.24	0.18	0.12	0.14	0.62	0.24	0.34	0.16	0.18	0.17	0.41	0.19	0.23	0.20	0.12	0.13	0.27	0.15	0.02
Fine-tuned																						
SMXM	345M	0.46	0.19	0.24	0.33	0.11	0.16	0.19	0.18	0.19	0.13	0.09	0.10	0.24	0.25	0.21	0.21	0.22	0.18	0.21	0.15	0.02
JUICER **(Ours)** · KP + KF	345M	**0.63**	0.23	0.31	**0.45**	0.16	0.23	0.62	0.37	0.46	0.49	0.25	0.33	0.37	0.45	0.38	0.30	0.32	0.31	**0.38**	**0.29**	0.02
Cross-domain fine-tuned																						
JUICER **(Ours)** · KP																						
+ KF (ONTO) + KF (MED)	345M	0.43	0.20	0.27	0.38	0.17	0.22	0.65	0.31	0.42	0.45	0.17	0.24	0.32	0.19	0.28	0.38	0.15	0.21	0.32	0.22	0.02
+ KF (ONTO) + KF (LEG)	345M	0.37	0.27	0.24	0.19	0.15	0.16	0.34	0.32	0.33	0.28	0.19	0.22	0.34	0.40	**0.43**	0.39	0.33	0.35	0.33	0.24	0.02
+ KF (MED) + KF (ONTO)	345M	0.46	0.22	0.29	0.35	0.16	0.22	0.68	0.43	**0.53**	**0.57**	0.29	**0.39**	0.50	0.23	0.31	0.39	0.21	0.27	**0.38**	**0.29**	0.02
+ KF (MED) + KF (LEG)	345M	0.18	0.17	0.13	0.13	0.10	0.10	0.43	**0.46**	0.44	0.33	0.25	0.27	0.23	0.59	0.37	0.40	0.33	0.36	0.31	0.24	0.02
+ KF (LEG) + KF (ONTO)	345M	0.42	0.26	0.23	0.18	0.12	0.14	0.49	0.30	0.37	0.42	0.22	0.28	0.32	0.20	0.29	0.41	0.20	0.27	0.30	0.23	0.02
+ KF (LEG) + KF (MED)	345M	0.29	0.16	0.20	0.26	0.13	0.16	0.52	0.33	0.40	0.37	0.17	0.23	0.26	0.30	0.31	**0.42**	0.21	0.28	0.30	0.20	0.02

(b) Dev set results

		MedMentions-ZS ♠						OntoNotes-ZS						LegalNER-ZS						Avg		
		T			*S*			*T*			*S*			*T*			*S*			*T*	*S*	
Large Language Models	**Size** ◇	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	F_1	F_1	
Zero-shot																						
BLOOM	176B	0.25	0.31	0.21	0.11	0.28	0.14	0.26	0.13	0.15	0.09	0.06	0.07	0.30	0.38	0.33	0.18	0.27	0.21	0.23	0.14	
FALCON	40B	0.06	0.36	0.08	0.03	0.27	0.05	0.13	0.31	0.18	0.05	0.12	0.07	0.08	0.33	0.13	0.05	0.25	0.09	0.13	0.07	
LLAMA 2	70B	0.29	0.51	**0.35**	0.19	**0.42**	**0.25**	0.06	0.71	0.11	0.04	0.37	0.07	0.43	0.63	0.48	0.34	0.53	0.39	0.31	0.24	
Fine-tuned (QLORA)																						
BLOOM	7B	0.17	**0.53**	0.19	0.13	0.37	0.15	0.03	**0.86**	0.05	0.03	**0.79**	0.06	0.19	0.61	0.28	0.16	0.47	0.23	0.17	0.15	
FALCON	7B	0.08	0.34	0.11	0.07	0.26	0.10	0.01	0.77	0.02	0.01	0.65	0.01	0.01	0.28	0.03	0.01	0.18	0.02	0.05	0.04	
LLAMA 2	7B	0.25	0.44	0.27	0.20	0.36	0.22	0.29	0.66	0.14	0.27	0.48	0.11	0.42	**0.71**	0.51	0.35	**0.62**	0.41	0.31	0.25	
		T			*S*			*T*			*S*			*T*			*S*			*T*	*S*	
Small Language Models	**Size**	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1	F_1	F_1	
Zero-shot																						
SMXM	345M	0.10	0.02	0.03	0.03	0.00	0.01	0.08	0.01	0.02	0.00	0.00	0.00	0.09	0.03	0.04	0.02	0.00	0.01	0.03	0.01	
JUICER **(Ours)** · KP	345M	0.48	0.17	0.24	0.22	0.10	0.13	0.48	0.15	0.20	0.22	0.09	0.11	0.57	0.19	0.24	0.30	0.08	0.12	0.23	0.12	
Fine-tuned																						
SMXM	345M	0.42	0.22	0.20	0.28	0.12	0.13	0.17	0.14	0.14	0.15	0.07	0.09	0.22	0.47	0.25	0.11	0.18	0.12	0.20	0.11	
JUICER **(Ours)** · KP + KF	345M	**0.58**	0.27	0.32	**0.41**	0.18	0.22	0.43	0.20	0.26	0.35	0.14	0.18	0.58	0.61	0.59	0.39	0.35	0.37	0.39	0.26	
Cross-domain fine-tuned																						
JUICER **(Ours)** · KP																						
+ KF (ONTO) + KF (MED)	345M	0.56	0.22	0.34	0.40	0.21	**0.25**	**0.78**	0.26	**0.42**	**0.46**	0.16	0.24	**0.79**	0.31	0.39	0.44	0.14	0.21	0.38	0.23	
+ KF (ONTO) + KF (LEG)	345M	0.30	0.25	0.26	0.20	0.17	0.17	0.26	0.17	0.34	0.27	0.23	0.25	0.78	0.58	**0.64**	**0.55**	0.39	**0.45**	**0.41**	**0.29**	
+ KF (MED) + KF (ONTO)	345M	0.50	0.27	0.32	0.36	0.18	0.21	0.56	0.32	0.40	0.44	0.21	0.28	0.77	0.35	0.47	0.34	0.14	0.20	0.40	0.23	
+ KF (MED) + KF (LEG)	345M	0.20	0.18	0.19	0.16	0.17	0.12	0.49	0.38	**0.42**	0.34	0.32	**0.32**	0.52	0.62	0.52	0.30	0.31	0.30	0.38	0.25	
+ KF (LEG) + KF (ONTO)	345M	0.38	0.19	0.24	0.23	0.11	0.15	0.38	0.16	0.31	0.34	0.15	0.21	**0.79**	0.33	0.45	0.33	0.14	0.20	0.33	0.19	
+ KF (LEG) + KF (MED)	345M	0.43	0.19	0.32	0.32	0.20	0.22	0.66	0.30	0.41	0.36	0.17	0.22	0.73	0.38	0.49	**0.55**	0.19	0.28	**0.41**	0.24	

♠ The benchmark most aligned with the transfer dataset domain.

◇ Reparameterization is applied, trained parameters for BLOOM, FALCON, LLAMA 2 are 0.06%, 0.03%, and 0.12%, respectively.

Average inference time per sample (seconds).

Table 5.1: Test set (a) and Dev set (b) performance comparison with token-level (T) and span-level (S) macro-averaged recall (R), precision (P), and F_1.

Bold *and* *underline* *denote the best and second best scores.*

and +0.19 on LegalNER-ZS. Furthermore, zero-shot JUICER even surpasses the fine-tuned SMXM version, reflecting the value of the KP stage introduced in this work, with a remarkable +0.15 macro-F_1 increase on OntoNotes-ZS, +0.02 on LegalNER-ZS and being on par on MedMentions-ZS. It's evident that the KF step provides substantial benefits to SMXM, but it's worth noting that all the fine-tuned models of JUICER consistently outperform the fine-tuned SMXMs across all three benchmarks: achieving a +0.06 macro-F_1 gain on MedMentions-ZS, +0.27 on OntoNotes-ZS, and +0.17 on LegalNER-ZS.

5.1.2 JUICER vs LLMs

Throughout our experiments, we consistently observed that LLMs tend to encounter challenges when it comes to NERC. These challenges manifest in various ways, including the discovery of unnecessary labels or the omission of essential ones. Additionally, LLMs sometimes exhibit hallucinations, introducing complexity in the post-processing phase. Consequently, they may not be the most suitable choice for NERC tasks. Despite these challenges, the largest LLM variants still manage to perform reasonably well, achieving scores that are on par with fine-tuned baseline models. Notably, the fine-tuned LLAMA 2 (7B) model stands out with an impressive macro-F_1 score of 0.39 on MedMentions-ZS. However, it's worth acknowledging that LLMs might have encountered these datasets during their intense pre-training, potentially affecting the zero-shot integrity of our experiments. On the contrary, supervised LLMs face notable difficulties when applied to OntoNotes-ZS and LegalNER-ZS datasets. Specifically, BLOOM (7B) and FALCON (7B) models tend to extract training classes during inference rather than the specific ones required for evaluation phase.

Impressively, when looking at the bigger picture, our JUICER models consistently achieve the highest overall macro-F_1 scores. In addition to their robust performance, our JUICER models offer the advantage of a compact architecture, significantly reducing inference time. In fact, our inference process is 157× faster, contributing to a more environmentally friendly and sustainable approach to AI.

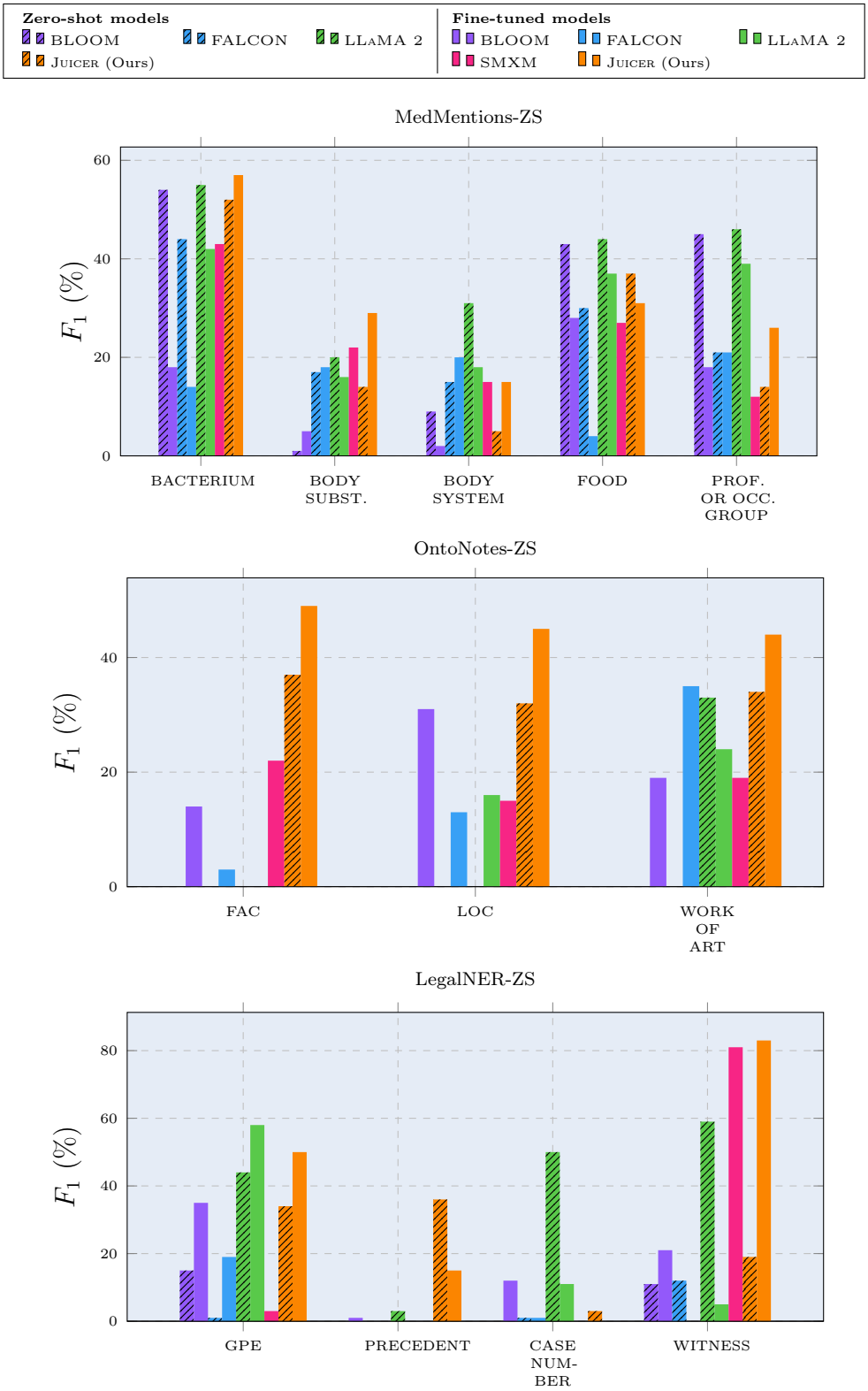


Figure 5.1: Per-class token-level macro-F_1 on the test set. Models are fine-tuned on each benchmark dataset separately.

5.1.3 Cross-domain Fine-tuning

Starting from a KP JUICER, we want to validate the influence of a greater amount of high-quality data with a two-step fine-tuning process. Our results unveiled that this extended procedure doesn't consistently enhance model performance and surprisingly, in some cases, can even produce inferior results compared to the traditional KP+KF method. The influence of this two-step fine-tuning process appears to vary depending on the dataset combinations used. For example, LEG+ONTO leads to deteriorated scores in both LegalNER-ZS and OntoNotes-ZS, while ONTO+LEG conspicuously bolsters results in LegalNER-ZS, though not for OntoNotes-ZS. Interestingly, the inclusion of the MedMentions-ZS (MED) dataset generally led to a decline in overall performance. This phenomenon might be attributed to the unique nature of the MedMentions-ZS dataset, where the SLM becomes exposed to domain-specific entities and faces challenges in leveraging distilled knowledge effectively. However, there were instances where combining the MedMentions-ZS dataset with the OntoNotes-ZS dataset (MED+ONTO) yielded impressive results, notably achieving a macro-F_1 score of 0.53 in OntoNotes-ZS. This outcome suggests a potential alignment in classes and descriptions between these datasets, resulting in improved performance. Our exploration of extended fine-tuning procedures highlights the nuanced interplay between datasets and the potential for varied outcomes. While some combinations may not yield the expected benefits, others can lead to substantial improvements in model performance, underscoring the complexity of knowledge transfer in NERC tasks.

5.1.4 KP and KF Trade-off

Figure 5.2 offers a comprehensive view of the dynamic relationship between the KF F_1 scores and the duration of the KP phase. The trends depicted in the figure provide compelling insights into how the length of the KP phase directly impacts zero-shot performance. Notably, the trends underscore a key observation: as the duration of KP extends, there is a corresponding increase in zero-shot performance. This relationship implies that a more extensive and thorough KP phase equips the model with a richer understanding of the domain,

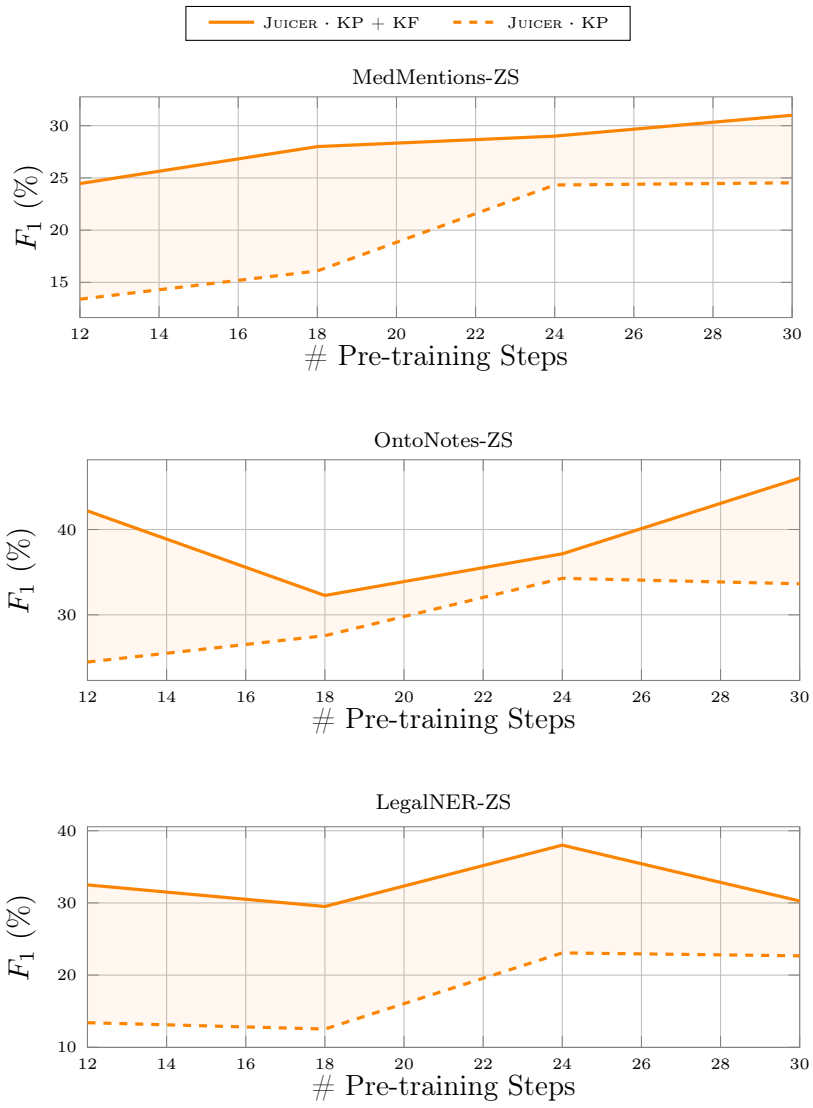


Figure 5.2: Token-level F_1 comparison between Juicer KP and Juicer KP+KF over pre-training steps.

enhancing its ability to perform effectively in zero-shot scenarios.

Furthermore, the consistent gains observed during the KF phase further validate the efficacy of incorporating a specialization window after the knowledge distillation process. This specialized phase allows the model to fine-tune its capabilities and align its knowledge with the specific requirements of the downstream task, resulting in improved overall performance. Figure 5.2 high-

lights the significance of a well-balanced KP and KF approach, emphasizing that the investment in KP can yield substantial benefits in terms of zero-shot NERC performance.

5.2 Discussion

The data's quality significantly impacts the final model's performance in KD. Methods like automatic filters, LLM ensembling, and manual curation can enhance data quality. Notably, this work's focus is on developing a *fully unsupervised and automatic approach* to improve zero-shot models, prioritizing efficiency and saving human time. While the exploration of multiple LLMs remains a potential avenue for future research, the current strategy involves the use of automatic filters and further fine-tuning of the model over human-labeled zero-shot datasets. This approach clearly demonstrates the advantages of utilizing high-quality data over distilled datasets, as evidenced by the results presented in Table 5.1.

The choice of the LLM for distillation holds a significant influence on the outcomes, necessitating the selection of the most suitable model. As the *zero-shot* setting demands substantial prior knowledge to detect unseen entities during inference, larger LLMs offer a distinct advantage. With its capacity to memorize vast amounts of data, a larger LLM like BLOOM is imbued with extensive general world knowledge. This relationship between LLM size and knowledge capacity is well-documented [66, 67], contingent upon the training data. In addressing the question of why we opted for a substantial LLM in our study, which primarily aims to reduce LLM-related costs and requirements, we conducted assessments using smaller LLMs for comparison. These assessments revealed a deterioration in the quality of the generated data when we attempted to reduce the number of model weights.

While our primary focus remains on reducing the demands associated with LLM usage, we intentionally included BLOOM in our evaluations to enable comparisons with the most proficient open-access LLM available. Importantly, it should be noted that the JUICER approach maintains model agnosticism, allowing for the distillation of smaller LLMs when circumstances warrant such

an approach. Furthermore, the results demonstrated significant improvements in OntoNotes-ZS, which belongs to a general domain, and LegalNER-ZS, which pertains to the legal field. These domains differ considerably from PubMed, which focuses on biomedicine. On the contrary, the enhancements observed in MedMentions, a closely related dataset to PubMed within the biomedical domain, were less prominent and aligned with its specific domain characteristics. This divergence in results can be attributed to two primary factors: (i) the LLM choice and (ii) the dataset complexity. First, the LLM used will extract the entities in its own way, and depending on its training, the entities extracted might be more generic (e.g., Medical Condition) or more specific (e.g., Eukaryote). Thus, depending on the alignment between the data generated by the teacher model and the final dataset used in the evaluation, the results may not improve even if the data are in the same domain. This phenomenon extends to the descriptions: if the pre-training data's descriptions are straightforward for the model to comprehend but those within the final dataset are more intricate, it can adversely impact the model's performance.

Conclusion and Future Work

The culmination of our research journey highlights the significant contributions and achievements of JUICER in the realm of zero-shot NERC. We embarked on this exploration in response to the pressing challenges of annotation scarcity and the need for models to generalize effectively to unseen entity-types. In this endeavor, embracing zero-shot learning became not just a choice but a necessity to navigate the absence of training examples. The core innovation of JUICER lies in its role as the first LLM distillation framework designed explicitly for zero-shot NERC in resource-constrained environments. We systematically transferred knowledge from LLMs to BERT-based models, enhancing their performance in zero-shot NERC. The key to this success was a preliminary fine-tuning process focused on generative data augmentation using vast pre-training corpora, followed by the injection of textual target class descriptions through cross-attention. Our extensive experiments across three zero-shot adapted BIO-format datasets, with a particular focus on the biomedical domain and assessment of adaptability to news and legal domains, yielded remarkable results. JUICER outperformed state-of-the-art baselines across all benchmarks by up to 0.27 macro-averaged F_1 points, showcasing its ability to excel across numerous observed classes. Even more impressively, when compared to zero-shot and reparameterized LLMs, JUICER achieved superior overall results with 510× fewer parameters. This reduction in model size, combined with a 157× faster inference process, not only enhances efficiency but also contributes to a more sustainable approach to AI. In our comparative analysis, we found that LLMs, while promising, face challenges in NERC tasks, such as discovering unnecessary labels, omitting essential ones, and even generating incorrect annotations. Among the largest zero-shot LLM variants, only the

fine-tuned LLaMA 2 (7B) model achieved an impressive macro-F_1 score of 0.39 on MedMentions-ZS. However, concerns may arise regarding the zero-shot constraint's integrity since we cannot verify whether the model encountered the MedMentions data during its intensive pre-training phase. Nevertheless, supervised LLMs encountered notable difficulties, in particular when applied to the OntoNotes-ZS and LegalNER-ZS datasets, potentially due to entity-types in the train set that might bear similarities with those in the test set. Consequently, the LLMs might erroneously associate test set mentions with familiar training set entity-types, leading to classification inaccuracies. A striking aspect of Juicer is its ability to provide robust performance while maintaining a compact architecture. The environmental benefits of such efficiency are not to be underestimated, marking a shift towards more sustainable AI practices. In our exploration of cross-domain fine-tuning procedures, we unveiled the intricate interplay between datasets and the potential for varied outcomes. While some combinations may not yield the expected benefits, others can lead to substantial improvements in model performance. This underscores the complexity of knowledge transfer in NERC tasks.

In conclusion, Juicer represents a step forward in the field of zero-shot NERC. It not only outperforms previous state-of-the-art zero-shot learning solutions but also excels in the challenging full zero-shot setting. Its remarkable performance surpasses even the largest and most powerful LLMs while maintaining an impressively compact architecture allowing for substantial cost reductions, faster inference times, and a reduced environmental footprint. All this reinforces the idea that efficient, lightweight neural networks can still emerge as formidable competitors for NERC tasks.

As future work, we will focus on enhancing Juicer's adaptation capabilities by leveraging a diverse range of domain-specific data sources. We will refine the pre-training process, striving for greater efficiency and effectiveness in adaptation to various domains. Moreover, we are committed to exploring ensemble approaches that harness the collective power of multiple LLMs. These ensemble techniques will not only enhance our model's performance but also enable us to extract higher-quality data during the entity extraction process, further improving the accuracy and robustness of zero-shot NERC.

Acknowledgements

The completion of this thesis represents the culmination of a rewarding journey that has been enriched by the invaluable contributions of many individuals and organizations. I am deeply grateful to all those who have played a significant role in shaping this work.

First and foremost, I would like to express my profound gratitude to my dedicated supervisor, Professor Claudio Sartori, whose guidance, expertise, and support have been the cornerstone of this research.

I extend my heartfelt thanks to my co-supervisor, Professor Gianluca Moro, and his team at DISI UniBo NLP. Their generous sharing of resources, expertise, and support have been instrumental in the successful execution of experiments and the development of my research approach. I would like to extend a special acknowledgment to Giacomo Frisoni, whose expert insights and valuable advice have significantly shaped the trajectory of this work.

I am deeply appreciative of my IBM tutor, Marcos Martínez Galindo, whose guidance and unwavering support during my internship at IBM Research Lab were instrumental in bringing this project to fruition.

I would like to express my gratitude to my IBM Manager, Vanessa Lopez, for providing me with the incredible opportunity to embark on this enriching internship. Your trust in my abilities and dedication to nurturing talent is truly commendable.

To my family, who have been my pillars of strength throughout this academic journey, I extend my deepest thanks. Your love, encouragement, and unwavering belief in me have been my constant motivation.

Last but not least, I acknowledge the collaborative nature of this endeavor, and it is important to highlight that the rights and ownership of any code and output generated as a result of this collaborative effort are attributed to IBM. This underscores the vital role of IBM in facilitating the successful completion of this research.

This thesis stands as a testament to the collective efforts of all these remarkable individuals and organizations. Thank you for being an integral part of this academic endeavor.

Alessio Cocchieri
October, 2023

Bibliography

[1] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Trans. Knowl. Data Eng.*, 34 (1):50–70, 2022.

[2] Wenxuan Zhou, Sheng Zhang, Yu Gu, Muhao Chen, et al. Universalner: Targeted distillation from large language models for open named entity recognition. *CoRR*, abs/2308.03279, 2023.

[3] Gabriele Picco, Marcos Martinez Galindo, Alberto Purpura, Leopold Fuchs, et al. Zshot: An open-source framework for zero-shot named entity recognition and relation extraction. In *ACL (Volume 3: System Demonstrations)*, pages 357–368, Toronto, Canada, July 2023. ACL. doi: 10.18653/v1/2023.acl-demo.34. URL `https://aclanthology.org/2023.acl-demo.34`.

[4] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, et al. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *ACL (Findings)*, pages 8003–8017. ACL, 2023.

[5] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, et al. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.

[6] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, et al. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[7] Lianmin Zheng, Zhuohan Li, Hao Zhang, Yonghao Zhuang, et al. Alpa:

Automating inter- and intra-operator parallelism for distributed deep learning. *CoRR*, abs/2201.12023, 2022.

[8] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, et al. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311, 2022.

[9] Sunil Mohan and Donghui Li. Medmentions: A large biomedical corpus annotated with UMLS concepts. In *AKBC*, 2019.

[10] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, et al. Towards robust linguistic analysis using ontonotes. In *CoNLL*, pages 143–152. ACL, 2013.

[11] Prathamesh Kalamkar, Astha Agarwal, Aman Tiwari, Smita Gupta, et al. Named entity recognition in Indian court judgments. In *NLLP*, pages 184–193, Abu Dhabi, United Arab Emirates (Hybrid), December 2022. ACL. doi: 10.18653/v1/2022.nllp-1.15. URL `https://aclanthology.org/2022.nllp-1.15`.

[12] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning - A comprehensive evaluation of the good, the bad and the ugly. *IEEE Trans. Pattern Anal. Mach. Intell.*, 41(9):2251–2265, 2019.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[15] Qingxiu Dong et al. A survey on in-context learning, 2023.

[16] Jason Wei et al. Finetuned language models are zero-shot learners, 2022.

[17] Long Ouyang et al. Training language models to follow instructions with human feedback, 2022.

[18] Hyung Won Chung et al. Scaling instruction-finetuned language models, 2022.

[19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[20] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018. URL `https://api.semanticscholar.org/CorpusID:49313245`.

[21] BigScience Workshop et al. Bloom: A 176b-parameter multilingual language model, 2023.

[22] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

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

[24] Hugo Touvron et. al. Llama 2: Open foundation and fine-tuned chat models, 2023.

[25] J. Simon. Large language models: A new moore's law?, October 26 2021. URL `https://huggingface.co/blog/large-language-models`. Retrieved February 10, 2023.

[26] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp, 2019.

[27] Jason Wei et al. Emergent abilities of large language models, 2022.

[28] Karl Cobbe et al. Training verifiers to solve math word problems, 2021.

[29] Jason Wei et al. Chain-of-thought prompting elicits reasoning in large language models, 2023.

[30] Karthik Valmeekam et al. Large language models still can't plan (a benchmark for llms on planning and reasoning about change), 2023.

[31] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(9):2251–2265, 2019. doi: 10.1109/TPAMI.2018.2857768.

[32] Wei Wang, Vincent W. Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2), jan 2019. ISSN 2157-6904. doi: 10.1145/3293318. URL `https://doi.org/10.1145/3293318`.

[33] Rami Aly, Andreas Vlachos, and Ryan McDonald. Leveraging type descriptions for zero-shot named entity recognition and classification. In *ACL/IJCNLP*, pages 1516–1528, Online, August 2021. ACL. doi: 10.18653/v1/2021.acl-long.120. URL `https://aclanthology.org/2021.acl-long.120`.

[34] Rasha Obeidat, Xiaoli Fern, Hamed Shahbazi, and Prasad Tadepalli. Description-based zero-shot fine-grained entity typing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 807–814, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1087. URL `https://aclanthology.org/N19-1087`.

[35] Hoang-Van Nguyen, Francesco Gelli, and Soujanya Poria. Dozen: Cross-domain zero shot named entity recognition with knowledge graph. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 1642–1646, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380379. doi: 10.1145/3404835.3463113. URL `https://doi.org/10.1145/3404835.3463113`.

[36] Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei

Zhang, Jiwei Li, and Guoyin Wang. Gpt-ner: Named entity recognition via large language models, 2023.

[37] Vladislav Lialin, Vijeta Deshpande, and Anna Rumshisky. Scaling down to scale up: A guide to parameter-efficient fine-tuning, 2023.

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

[39] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *CoRR*, abs/2305.14314, 2023.

[40] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *KDD*, pages 535–541. ACM, 2006. ISBN 1-59593-339-5. URL `http://dblp.uni-trier.de/db/conf/kdd/kdd2006.html#BucilaCN06`.

[41] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[42] Cristian Buciluundefined, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, page 535–541, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933395. doi: 10.1145/1150402.1150464. URL `https://doi.org/10.1145/1150402.1150464`.

[43] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding, 2020.

[44] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.

[45] Young Jin Kim and Hany Hassan Awadalla. Fastformers: Highly efficient transformer models for natural language understanding, 2020.

[46] Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *EMNLP*, pages 1317–1327. ACL, 2016.

[47] NLLB Team and Marta R. Costa jussà et al. No language left behind: Scaling human-centered machine translation, 2022.

[48] Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. Lamini-lm: A diverse herd of distilled models from large-scale instructions, 2023.

[49] Rohan Taori et al. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[50] Shuohang Wang, Yang Liu, Yichong Xu, Chenguang Zhu, et al. Want to reduce labeling cost? GPT-3 can help. In *EMNLP*, pages 4195–4205, Punta Cana, Dominican Republic, November 2021. ACL. doi: 10.18653/v1/2021.findings-emnlp.354. URL `https://aclanthology.org/2021.findings-emnlp.354`.

[51] Peter West, Chandra Bhagavatula, Jack Hessel, Jena Hwang, et al. Symbolic knowledge distillation: from general language models to commonsense models. In *NAACL*, pages 4602–4625, Seattle, United States, July 2022. ACL. doi: 10.18653/v1/2022.naacl-main.341. URL `https://aclanthology.org/2022.naacl-main.341`.

[52] Ryan Smith, Jason A. Fries, Braden Hancock, and Stephen H. Bach. Language models in the loop: Incorporating prompting into weak supervision. *CoRR*, abs/2205.02318, 2022.

[53] Priyanka Agrawal, Chris Alberti, Fantine Huot, Joshua Maynez, et al. Qameleon: Multilingual QA with only 5 examples. *CoRR*, abs/2211.08264, 2022.

[54] Zhen Guo, Peiqi Wang, Yanwei Wang, and Shangdi Yu. Dr. llama: Improving small language models on pubmedqa via generative data augmentation. *CoRR*, abs/2305.07804, 2023.

[55] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

[56] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL `https://api.semanticscholar.org/CorpusID:160025533`.

[57] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[58] Microsoft. Turing-nlg: A 17-billion-parameter language model, 2020. URL `https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/`. Microsoft Blog.

[59] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models, 2021.

[60] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, et al. The pile: An 800gb dataset of diverse text for language modeling. *CoRR*, abs/2101.00027, 2021.

[61] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, et al. BLOOM: A 176b-parameter open-access multilingual language model. *CoRR*, abs/2211.05100, 2022.

[62] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, et al. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416, 2022.

[63] Shayne Longpre, Le Hou, Tu Vu, Albert Webson, et al. The flan collection: Designing data and methods for effective instruction tuning. In *ICML*, volume 202 of *PMLR*, pages 22631–22648. PMLR, 2023.

[64] Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, et al. UL2: unifying language learning paradigms. In *ICLR*. OpenReview.net, 2023.

[65] Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, et al. Crosslingual generalization through multitask finetuning. In *ACL (1)*, pages 15991–16111. Association for Computational Linguistics, 2023.

[66] Kushal Tirumala, Aram H. Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models, 2022.

[67] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, et al. Quantifying memorization across neural language models. *arXiv*, abs/2202.07646, 2022. URL `https://api.semanticscholar.org/CorpusID:246863735`.

[68] Steven Bird and Edward Loper. NLTK: The natural language toolkit. In *ACL*, pages 214–217, Barcelona, Spain, July 2004. ACL. URL `https://aclanthology.org/P04-3031`.

[69] Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning, 2022.