ISSN 2499-4553

IJCoL

Italian Journal of Computational Linguistics

Rivista Italiana di Linguistica Computazionale

Volume 10, Number 1
june 2024

aA Accademia university press

Registrazione presso il Tribunale di Trento n. 14/16 del 6 luglio 2016

Rivista Semestrale dell'Associazione Italiana di Linguistica Computazionale (AILC)

Associazione Italiana di Linguistica Computazionale

direttore responsabile
Michele Arnese

isbn 9791255000983

Accademia University Press
via Carlo Alberto 55
I-10123 Torino
info@aAccademia.it
www.aAccademia.it/IJCoL_10_1

aAccademia university press

Accademia University Press è un marchio registrato di proprietà
di LEXIS Compagnia Editoriale in Torino srl

IJCoL

Volume 10, Number 1
june 2024

CONTENTS

U–DepPLLaMA: Universal Dependency Parsing via Auto-regressive Large Language Models

Claudiu Daniel Hromei*
Università di Roma, Tor Vergata

Danilo Croce**
Università di Roma, Tor Vergata

Roberto Basili†
Università di Roma, Tor Vergata

This paper investigates the rapidly advancing domain of Large Language Models (LLMs) and their growing potential in various fields. A central focus is the exploration of LLMs, e.g., LLaMA, as powerful tools for modeling and representing linguistic information, especially in the realm of syntax. We aim to evaluate the ability of these models to encode syntactic information, especially when explicitly supplied, through fine-tuning processes. Traditionally, Dependency Parsing has relied on specific techniques and dedicated architectures. Our research shifts this approach, conceptualizing it as a sequence-to-sequence task where Language Models interpret and transform syntax into bracketed structures that reflect dependency graphs. We introduce U-DepPLLaMA (Universal Dependency Parsing via auto-regressive LLMs based on LLaMA), a novel architecture optimized for multilingual, end-to-end Dependency Parsing. Our experimental evaluation, across 50 datasets in 26 languages from the Universal Dependency Treebank, shows that LLMs can be effectively trained for dependency parsing without the need for task-specific architectures. The results are on par with current state-of-the-art methods and demonstrate resilience across varying sentence complexities and lengths.

1. Introduction

In recent years, Large Language Models (LLMs) have emerged as a dominant paradigm in the field of natural language processing, revolutionizing the way linguistic data is interpreted and utilized. The increasing sophistication of these models has opened new avenues for understanding the depth and breadth of linguistic information they can capture. This paper centers on probing the abilities of LLMs, particularly focusing on their capacity to model syntactic information, which is the cornerstone of language structure. By delving into the intricacies of this syntactic modeling, our research aims to shed light on the capabilities and potential limitations of LLMs in representing the complex relationships inherent in such structures. In particular, Dependency parsing is a relevant stage in natural language processing, playing a vital role in capturing the

* Department of Enterprise Engineering - Via del Politecnico 1, 00133 Rome, Italy.
E-mail: hromei@ing.uniroma2.it
** Department of Enterprise Engineering - Via del Politecnico 1, 00133 Rome, Italy.
E-mail: croce@info.uniroma2.it
† Department of Enterprise Engineering - Via del Politecnico 1, 00133 Rome, Italy.
E-mail: basili@info.uniroma2.it

syntactic and semantic relationships within sentences (Kübler, McDonald, and Nivre 2009). Its primary goal is to establish dependencies among words, enabling humans to grasp how words interconnect and rely on one another within a sentence's structure (Tesnière 1959). This understanding is invaluable across various applications, e.g., relation extraction or machine translation. High-performance neural parsing systems can promote the explanation of emerging grammatical competence, as carried out through probing techniques (Conneau et al. 2018). One prominent parsing technique is the shift-reduce method, as exemplified in (Nivre 2008; Chen and Manning 2014). This parser processes sentences from left to right, word by word while maintaining a buffer for unprocessed words. Additionally, machine learning-based approaches, such as biaffine neural networks, as demonstrated in (Dozat and Manning 2016), have proven effective in capturing complex word dependencies. An intriguing parsing architecture is UDPipe (Straka and Straková 2017), which focuses on parsing within the Universal Dependency Framework (de Marneffe et al. 2014). UDPipe stands out because it handles dependency parsing along with essential tasks like tokenization, morphological analysis, part-of-speech tagging, and lemmatization for multiple languages, all without relying on external data. It employs a Bi-LSTM model fed with end-to-end, character-level, pre-trained, and contextualized embeddings. The model was trained on a vast dataset spanning multiple languages, effectively capturing cross-lingual relations. The architecture was later expanded as UDPipe+ (Straka, Straková, and Hajic 2019), incorporating multilingual BERT (Devlin et al. 2019) into its token representations. However, despite their state-of-the-art results in various languages, these methods are specifically tailored for solving structure prediction problems using ad-hoc techniques, neural architectures and complex optimizations.

More recently, Transformer-based models (Vaswani et al. 2017) have gained popularity, mainly because of their ability to handle many different tasks, including classification, regression, and rewriting. These models have the inherent ability to work with sequences, taking one as input and creating another as output. In particular, the appeal of sequence-to-sequence modeling for dependency parsing emerged with the pioneering work of (Aharoni and Goldberg 2017) which defined linearized trees to generate parsers with sequential decoders. Furthermore, in (Li et al. 2018) the authors presented an end-to-end seq2seq approach to dependency parsing, allowing the model to directly predict the relative position of the head of each word within a sentence. It also incorporated a beam search decoder with tree constraints and sub-root decomposition to enhance results. Moreover, in (Kondratyuk and Straka 2019b), authors experimented with a multi-task, multilingual version of BERT (Devlin et al. 2019) pre-trained on 104 languages, capable of predicting not only dependency parsing trees but also lemmas, part-of-speech tags, and more for each word in an input sentence. A notable Transformer-based architecture is LLaMA (Touvron et al. 2023a, 2023b), a large language model (LLM) with billions of parameters that generates output sequences in an autoregressive manner based on input and previously generated tokens.

In this study, we assess the capability of Large Language Models (LLMs) to encode syntactic structures, particularly focusing on their performance when enhanced with explicit syntactic data through fine-tuning. This evaluation is essential to understand how well LLMs, when equipped with targeted refinements, can adapt to and represent complex linguistic patterns. In particular, we explore the ability of Large Language Models (LLMs) to perform end-to-end parsing without the need for any task-specific architectural design and potentially enhance the ability to induce syntactic information from text data. We focus on the use of fine-tuned LLaMA-based models to predict tree-shaped representations, by turning sentence syntax into bracketed forms. These brack-

eted structures, which can be directly translated from sentence dependency graphs, were previously utilized to represent the sentence's syntactic structure in a tree format (Croce, Moschitti, and Basili 2011; Filice et al. 2018). Our goal is to train a system that can take raw sentences as input and produce these bracketed outputs. Once we have this bracketed representation, creating the dependency graph becomes straightforward. This exploration aims to provide insights into the potential of LLMs to transform the landscape of natural language processing, especially in domains where precise syntactic understanding is vital.

Experimental results on 50 treebanks in 26 different languages demonstrate that this approach is not only feasible but also capable of achieving results comparable to the state-of-the-art, all while requiring minimal training resources, such as training on a single GPU with "limited" memory. In-depth error analysis shows that a model like LLaMA generates symbols based on the entirety of the input sequence and this allows the parsing accuracy to behave independently from the sentence dependency length. This is promising as it challenges conventional notions about the pitfalls of parsing longer textual inputs.

In the following sections, we detail our approach in Section 2, present experimental results in Section 3, and conclude in Section 4.

2. Universal Dependency Parsing via Auto-regressive LLMs

The Transformer architecture (Vaswani et al. 2017) has significantly advanced Natural Language Processing (NLP). Broadly, Transformers can be classified into three types: Encoder-only models (e.g., BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019)), which generate meaningful representations; Decoder-only models (e.g., GPT (Radford et al. 2018), LLaMA (Touvron et al. 2023a, 2023b)), adept at autoregressive language generation; and Encoder-Decoder models (e.g., T5 (Raffel et al. 2020), BART (Lewis et al. 2019)), combining both encoder and decoder strengths, ideal for tasks like translation or summarization. Large Language Models (LLMs) have recently set benchmarks in various NLP tasks. Notably, ChatGPT and the updated LLaMA 2 model, especially its dialogue-optimized variant LLaMA 2-Chat, stand out. LLaMA 2 incorporates enhancements like a pretraining corpus made of 2 trillion tokens, doubled context length, and grouped-query attention (Ainslie et al. 2023).

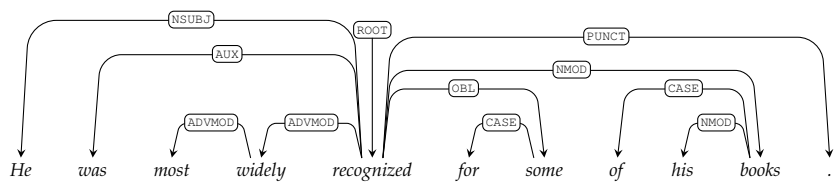


Figure 1
Example of a dependency graph associated with the sentence (1).

Modeling tasks using LLMs poses a unique challenge due to their inherent nature of taking sequences as input and generating sequences as output. To illustrate this, let's consider the following sentence:

"*He was most widely recognized for some of his books.*" (1)

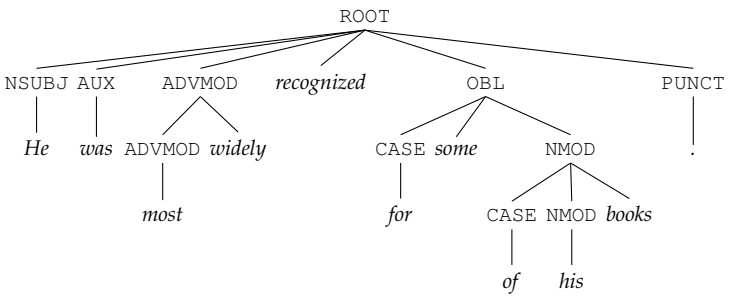


Figure 2
The syntactic parse tree associated with the dependency graph from Figure (1).

The Dependency Graph of this sentence is illustrated in Figure 1, where nodes represent words, and arcs express syntactic relationships, each labeled with a specific dependency type. A notable node is ROOT, typically representing the main verb. The same syntactic information is portrayed as a Dependency Tree in Figure 2, with the main verb as the actual tree ROOT. In this tree, non-terminal nodes reflect dependency labels, while terminal nodes depict words from the sentence. For instance, the NSUBJ arc shows *He* as the subject, and the OBL arc indicates a nominal phrase with the word *some*. The verb *recognized* is modified by *widely*, further modified by *most* via the ADVMOD relation. Both representations, the dependency graphs and trees, convey the same information, and transitioning between them preserves this information. The only potential complexity arises from the non-projectivity of the dependency graph, which might alter word order within the tree. However, this doesn't hinder syntactic analysis since realigning the words to match the original sentence is straightforward. More details about the conversion algorithm and the application to a non-projective parse tree are discussed in the Appendix A.

The plain parenthetic representation, computed as in (Croce, Moschitti, and Basili 2011), of the dependency tree in Figure 2 corresponds to:

$$\begin{aligned}&[\texttt{ROOT}[\texttt{NSUBJ}[\textit{He}]][\texttt{AUX}[\textit{was}]][\texttt{ADVMOD}[\texttt{ADVMOD}[\textit{most}]][\textit{widely}]][\textit{recognized}]\\&[\texttt{OBL}[\texttt{CASE}[\textit{for}]][\textit{some}][\texttt{NMOD}[\texttt{CASE}[\textit{of}]][\texttt{NMOD}[\textit{his}]][\textit{books}]]][\texttt{PUNCT}[.]]]\end{aligned} \tag{2}$$

The idea is thus straightforward: we aim to employ an LLM to transform the example sentence 1 into its parenthetical format 2, using solely raw text, without the aid of intermediate additional information like morphological data, part-of-speech tags, or lemmatization. In particular, this sequence-to-sequence model is obtained by fine-tuning the LLaMA2 $7B$[1] and $13B$[2] models (Touvron et al. 2023b). As of this writing and inspired by (Hromei et al. 2023) which applied a single architecture to solve a plethora of linguistic tasks, these models represent the optimal balance between model accuracy and size across a wide range of tasks (Touvron et al. 2023b).

It's essential to understand that LLaMA2 operates as an auto-regressive sequence-to-sequence model, processing input text and generating output text. Upon analyzing an input sentence, the model's first task is to identify and generate the ROOT

1 https://huggingface.co/meta-llama/Llama-2-7b-hf
2 https://huggingface.co/meta-llama/Llama-2-13b-hf

node, which typically corresponds to the primary verb or central idea in the sentence. Once established, the model's attention mechanism begins to delineate the primary nodes directly connected to the `ROOT` prefix, effectively capturing the main syntactic constituents or "branches" that sprout from this root. At this juncture, the model is expected to employ a recursive strategy. For each of these primary nodes, the model drills down to identify and generate its corresponding subtrees. As it descends deeper into these syntactic branches, the phenomena it encounters tend to be more localized to specific sentence segments and are, thus, generally more straightforward. Essentially, the model unpacks each subtree (i.e., syntactic chunks), always maintaining focus on the nodes and words that haven't been processed yet. First, "*He*" will be focused, then "*was*", then "*most widely*" and so on. This behavioral pattern should emerge as a result of the specified sentence output prefix notation, as shown in Equation 2. In this context, the `ROOT` assumes the position of the primary label, succeeded sequentially by the label and corresponding word of the initial element. This linear configuration facilitates a systematic rewriting of the sentence, affording focused attention to each subtree originating from the central `ROOT`. Throughout this parsing endeavor, the model is anticipated to exploit the attention mechanism to effectively handle short and long-range dependencies. The recursive nature of this approach ensures that each segment of the sentence, from broad constituents down to specific syntactic details, is thoroughly and systematically examined. We anticipate that the language model should proficiently handle this parenthetical form, considering its resemblance to the recursive structures prevalent in programming languages like C and Java. Many language models, including LLaMA, have been extensively pre-trained on these languages, equipping them with a foundational understanding of similar syntactic patterns, typical in the use of parentheses in functional expressions.

Furthermore, given the inherent multilingualism of syntactic information (as demonstrated by the Universal Dependency Annotation scheme of (McDonald et al. 2013)) and the capability of LLMs to handle multiple languages, we will apply this framework to a multilingual model, aggregating data across as many languages as possible to achieve a unique, language-independent architecture. It's important to note that LLaMA2 was not pre-trained on data from every existing language. Instead, it was trained on a collection of texts in 27 languages, with a vast majority of almost 90% of the sentences being in English. The remaining languages make up less than 1% each of the training data (the complete list is provided in Appendix B). To gauge the model's efficacy, we will focus solely on data from those specific languages. While this approach doesn't address challenges associated with low-resource languages (not considered in the LLaMA2 pre-training), we do not wish to attribute potential shortcomings in our initial experiments to a lack of pre-training data. Nonetheless, there are no constraints on training LLaMA2 with data from those languages, or in cross-lingual parsing scenarios, as highlighted by (Sherborne and Lapata 2022). We will thus rely on data available from the Universal Dependency Treebank[3] and refer to our model as `U-DepPLLaMA`: Universal Dependency Parsing via Auto-regressive LLMs based on LLaMA.

Efficient fine-tuning techniques. Training and fine-tuning large-scale models typically necessitate vast computational resources, often requiring an array of GPUs. An efficient solution has been proposed with the advent of Low-Rank Adaptation (LoRA, from (Hu et al. 2022)). This method predominantly freezes the pre-trained model's weights, while introducing trainable rank decomposition matrices within each Transformer layer, sig-

3 `https://universaldependencies.org/`

nificantly reducing the trainable parameters without any added inference delay. The Quantized-LoRA technique (Q-LoRA), as detailed in (Dettmers et al. 2023), further enhances this efficiency. It permits the fine-tuning of a 7-billion-parameter model (7B) on a conventional 16GB GPU, preserving full 16-bit task performance. This is achieved by propagating gradients via a 4-bit quantized frozen pre-trained model within the LoRA structure. For our purposes, we adopted the QLoRA-based method, which allows for the training of a combined architecture where only a small fraction of parameters undergoes fine-tuning, leaving the majority of LLaMA's parameters untouched during the process. After this fine-tuning phase, only the fine-tuned component remains (the adapters), which can be integrated with a model like LLaMA for dependency parsing[4].

3. Experimental Evaluation

Objectives. We primarily investigate the potential of the proposed model for multilingual dependency parsing. We emphasize that this model isn't inherently task-specific; its uniqueness stems from the distinct input and output forms. A crucial question is its robustness across languages, especially those it has been minimally exposed to during pre-training (all except English). We also question the influence of sentence complexity on parsing quality. Concurrently, we explore the benefits and limitations of employing architectures ranging from 7 billion to 13 billion parameters. Lastly, we evaluate LLaMA2's end-to-end capability, where it's tasked with autonomously deriving tokenization from 'raw' sentences and producing the resultant dependency graph.

Dataset and Evaluation metrics. In our experiments, we narrowed our focus to the Universal Dependency Parsing dataset version 2.3[5]. Our approach aligns with the state-of-the-art architecture for dependency parsing proposed by (Straka, Straková, and Hajic 2019). From the Universal Dependency Parsing dataset, our concentration was particularly on the subset whose languages were supported by LLaMA2, as indicated in their report (Touvron et al. 2023b). This subset included 27 languages, from English and Finnish to Korean (all shown in the first column of Table 1). Unfortunately, we discarded texts written in Vietnamese because we encountered problems with encoding and the LLaMA2 tokenizer. Our dataset thus contains examples written in 26 languages. For languages like Czech, Norwegian, and Russian with over $30,000$ examples, we limit the training datasets to a maximum of $30,000$ sentences per language, retaining balanced datasets across languages. Our neural architecture was trained on the aggregated dataset of all supported languages after the cutoff reduction, which consisted of $392,088$ training examples, $59,084$ development sentences, and $62,069$ testing examples. To ensure consistency with the evaluation framework presented in (Straka and Straková 2017; Straka, Straková, and Hajic 2019), we have chosen to employ two pivotal metrics for assessing the performance of the LLaMA model: Unlabeled Attachment Score (*UAS*) and Labeled Attachment Score (*LAS*). *UAS* evaluates the precision of the model's dependency tree structure, verifying the correct generation of head and dependency arcs. In contrast, *LAS* offers a more comprehensive evaluation by also gauging the accuracy of the assigned dependency labels for each dependency arc.

4 The model, along with all the necessary software for training and utilizing U-DepPLLaMA, is publicly released on a dedicated GitHub page at `https://github.com/crux82/u-deppllama` according to the LLaMA `https://ai.meta.com/llama/license/` and UD `https://github.com/UniversalDependencies/LICENSE` licenses.

5 `http://hdl.handle.net/11234/1-2895`

Experimental details. The model training process employed PyTorch and the Huggingface library, complemented by the Peft packages, for the implementation of the Q-LoRA technique. The LLaMA2 models were subjected to 3 training epochs, utilizing a learning rate of $3 \cdot 10^{-4}$ and a batch size of 16. To optimize the model's performance, a linear scheduler with warmup was employed, incorporating a warmup ratio of 0.1. Throughout the training process, Q-LoRA with 4-bit precision was utilized to enhance the transformer's W_q, W_v, W_k, W_o, W_{gate}, W_{down}, W_{up} and W_{lm_head} modules, as detailed in the work of (Dettmers et al. 2023). The LoRA matrices featured a matrix rank (R) of 8 and a parameter α of 16. The training took place on a single NVIDIA H100 GPU equipped with 80GB total memory, although only a maximum of 20GB (using Nvidia MIG) was utilized during both training and inference. Training time took about 160 hours for `U-DepPLLaMA 7B` and 190 hours for `U-DepPLLaMA 13B` on a single GPU. This is particularly noteworthy as it demonstrates the applicability of Q-LoRA even on standard architectures with the two smallest available models, having 7 and 13 billion parameters, without necessitating significant computing power[6]. All hyper-parameters were selected using a subset comprising 10% of the training and development sets. In the final generation process, we opted against sampling, as our objective was not to increase sequence variability but to identify the best sequence that explains the given sentence. Consequently, we adopted a greedy search policy. While initial experiments considered beam search to generate an optimal tree structure, preliminary results on a subset of the data indicated a potential increase of 0.2% in UAS and/or LAS. However, this came at a significant computational cost, with decoding time increasing by nearly an order of magnitude. As a result, we chose the greedy approach to prioritize efficiency over a modest performance boost.

Results Discussion. In this section, we present our experimental results, specifically focusing on the comparison between our method and the state-of-the-art linguistic analysis system, UDPipe2.0 (Straka, Straková, and Hajic 2019). The most notable distinction between our approach and UDPipe2.0 lies in their architectures. UDPipe2.0 is a task-specific architecture, weaving together recurrent networks, attention-based biaffine networks, and diverse word embedding techniques. It also tackles a plethora of tasks beyond parsing, such as POS tagging and lemmatization. Additionally, it boasts training on a vast 89 datasets across 54 languages. In contrast, our model leans heavily on LLaMA2. Its uniqueness does not stem from architectural alterations but rather from its prompting method. It was trained on a narrower scope of 50 datasets for languages supported by LLaMA. Thus, while our architecture inherently possesses more parameters, it fundamentally remains a task-agnostic language model. We investigated different output formats as in (Li et al. 2018; Kondratyuk and Straka 2019a) where sentences such as "*The dog runs*" are rewritten in pseudo-sentences, such as "`R1-det R1-nsubj ROOT`". An explicit counter of the relative distance between the head and modifier is here adopted to label the dependency relation. However, LLMs are not explicitly designed for precise numerical tasks, and accurately measuring distances between tokens can be very hard. This limitation was especially observed in labeling decisions involving long-distance dependencies, such as correctly attaching a punctuation mark to a distant ROOT. Owing to these complexities and space limitations, we have omitted these unstable results, as they do not accurately reflect the model's true performance. In Table 1, the first columns present the UAS and LAS performance metrics for UDPipe2.0 and its enhanced variant, UDPipe2.0++. The key distinction between

6 Reducing the batch size to 4, they can be trained on an Nvidia T4 too, with 16GB of memory.

Table 1
Results in terms of UAS and LAS for UDPipe 2.0, UDPipe 2.0++ with the different embeddings, U-DepPLLaMA 2 7B and 13B with Gold Standard (GS) Tokenization and the End-to-End version of U-DepPLLaMA 2 7B. These last results are in italics as they are not comparable with the others because the End-to-End model is trying to solve a much more complex task. The best result for each treebank is highlighted in bold.

Language	UDPipe 2.0		UDPipe 2.0++		U-DepPLLaMA 7B GS Tokenization		U-DepPLLaMA 13B GS Tokenization		U-DepPLLaMA 7B End-to-End	
	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
Bulgarian-BTB	93,38%	90,35%	95,34%	92,62%	96,21%	93,55%	**96,37%**	**93,77%**	*95,82%*	*93,27%*
Catalan-AnCora	93,22%	91,06%	94,49%	92,74%	94,28%	92,60%	**94,53%**	**92,86%**	*94,22%*	*92,47%*
Chinese-GSD	84,64%	80,50%	**90,13%**	**86,74%**	85,35%	81,09%	86,20%	82,21%	*83,40%*	*78,69%*
Croatian-SET	91,10%	86,78%	93,20%	89,35%	93,09%	88,47%	**93,71%**	**89,51%**	*93,07%*	*88,60%*
Czech-CAC	92,99%	90,71%	93,59%	91,50%	93,39%	91,19%	**94,48%**	**92,71%**	*93,50%*	*91,35%*
Czech-CLTT	86,90%	84,03%	89,59%	87,01%	93,14%	90,84%	91,82%	89,55%	***94,38%***	***92,28%***
Czech-FicTree	92,91%	89,75%	94,34%	91,87%	93,58%	90,94%	**95,30%**	**93,44%**	*93,74%*	*91,05%*
Czech-PDT	93,33%	91,31%	94,43%	92,56%	93,51%	91,44%	**94,85%**	**93,22%**	*93,45%*	*91,41%*
Danish-DDT	86,88%	84,31%	**89,32%**	**87,24%**	87,29%	84,63%	88,04%	85,55%	*87,77%*	*84,96%*
Dutch-Alpino	91,37%	88,38%	94,12%	91,78%	94,07%	91,78%	**94,32%**	**91,93%**	*92,94%*	*90,24%*
Dutch-LassySmall	90,20%	86,39%	93,07%	89,88%	**94,33%**	91,13%	94,19%	91,07%	*94,19%*	***91,29%***
English-EWT	89,63%	86,97%	**92,50%**	**90,40%**	92,19%	89,91%	92,32%	90,05%	*91,85%*	*89,44%*
English-GUM	87,27%	84,12%	**91,47%**	**88,80%**	90,84%	87,92%	90,42%	87,47%	*90,71%*	*87,88%*
English-LinES	84,15%	79,71%	87,28%	83,48%	**88,25%**	84,94%	88,23%	**84,97%**	*88,21%*	*84,71%*
English-ParTUT	90,29%	87,27%	**93,75%**	**91,12%**	93,05%	90,41%	92,96%	90,29%	*91,43%*	*88,85%*
Finnish-FTB	90,68%	87,89%	**91,68%**	**89,02%**	85,51%	80,60%	86,74%	81,95%	*91,31%*	*88,53%*
Finnish-TDT	89,88%	87,46%	**91,66%**	**89,49%**	85,96%	81,69%	86,90%	83,01%	*91,33%*	*88,74%*
French-GSD	90,65%	88,06%	92,55%	90,31%	93,51%	91,21%	**94,00%**	**91,90%**	*93,80%*	*91,51%*
French-ParTUT	92,17%	89,63%	**94,51%**	**92,47%**	92,03%	89,42%	91,36%	88,75%	*91,32%*	*88,34%*
French-Sequoia	92,37%	90,73%	**94,88%**	**93,81%**	92,02%	89,89%	91,92%	89,47%	*92,12%*	*89,54%*
French-Spoken	82,90%	77,53%	**86,27%**	**81,40%**	85,66%	80,24%	84,77%	78,92%	*85,30%*	*80,29%*
German-GSD	85,53%	81,07%	88,11%	84,06%	**88,34%**	**84,16%**	88,22%	84,14%	*87,82%*	*83,65%*
Hungarian-Szeged	84,04%	79,73%	**88,76%**	**85,12%**	85,42%	80,59%	86,87%	82,03%	*86,01%*	*81,24%*
Indonesian-GSD	85,31%	78,99%	86,47%	**80,40%**	86,43%	80,05%	86,09%	80,08%	***86,49%***	*80,32%*
Italian-ISDT	93,49%	91,54%	94,97%	93,38%	95,20%	93,61%	**95,65%**	**94,09%**	*95,02%*	*93,39%*
Italian-ParTUT	92,64%	90,47%	95,36%	93,38%	**96,65%**	**94,59%**	96,62%	94,42%	*96,20%*	*93,92%*
Italian-PoSTWITA	86,03%	81,78%	87,25%	83,07%	87,24%	83,40%	**87,65%**	**83,87%**	*87,33%*	*83,35%*
Japanese-GSD	95,06%	93,73%	**95,55%**	**94,27%**	93,32%	90,83%	92,53%	89,85%	*87,18%*	*84,14%*
Korean-GSD	87,70%	84,24%	**89,38%**	**86,05%**	79,81%	70,33%	80,21%	70,56%	*85,66%*	*80,88%*
Korean-Kaist	88,42%	86,48%	**89,35%**	**87,54%**	86,00%	82,63%	87,15%	83,79%	*87,17%*	*84,82%*
Norw.-Bokmaal	92,39%	90,49%	93,78%	92,19%	93,81%	92,20%	94,28%	**92,78%**	***94,34%***	*92,70%*
Norw.-Nynorsk	80,09%	75,04%	82,64%	78,08%	86,55%	83,16%	**87,02%**	**83,68%**	*86,57%*	*83,14%*
Norw.-NynorskLIA	68,08%	60,07%	**71,42%**	64,12%	70,87%	**64,41%**	70,75%	64,23%	*70,95%*	*64,24%*
Polish-LFG	96,58%	94,76%	**97,44%**	**96,03%**	96,22%	94,11%	96,38%	94,47%	*96,28%*	*94,05%*
Polish-SZ	93,39%	91,24%	**95,73%**	**94,25%**	94,11%	89,67%	94,18%	89,63%	*93,61%*	*89,18%*
Portuguese-Bosque	91,36%	89,04%	**92,69%**	**90,70%**	91,22%	87,66%	92,12%	88,64%	*89,77%*	*84,83%*
Portuguese-GSD	93,01%	91,63%	**94,74%**	**93,71%**	94,44%	92,92%	94,54%	93,13%	*94,18%*	*92,35%*
Romanian-Nonst.	89,12%	84,20%	89,61%	84,78%	88,71%	83,73%	**89,88%**	**85,32%**	*89,21%*	*83,28%*
Romanian-RRT	91,31%	86,74%	92,41%	88,05%	92,47%	88,18%	**92,72%**	**88,46%**	*92,68%*	*88,27%*
Russian-GSD	88,15%	84,37%	90,74%	**87,51%**	90,10%	85,56%	**91,16%**	87,03%	*90,76%*	*86,38%*
Russian-SynTagRus	93,80%	92,32%	94,92%	**93,68%**	94,45%	92,68%	**95,21%**	93,62%	*94,41%*	*92,65%*
Russian-Taiga	75,45%	69,11%	80,74%	75,65%	82,94%	77,08%	**84,91%**	**79,27%**	*83,42%*	*77,34%*
Serbian-SET	92,70%	89,27%	94,57%	91,65%	95,24%	91,85%	**96,14%**	**92,52%**	*95,87%*	*92,49%*
Slovenian-SSJ	92,96%	91,16%	**94,81%**	**93,49%**	94,14%	92,55%	94,43%	93,14%	*94,04%*	*92,51%*
Slovenian-SST	73,51%	67,51%	**77,23%**	**71,79%**	77,12%	71,06%	76,65%	71,06%	*76,78%*	*70,67%*
Spanish-AnCora	92,34%	90,26%	**93,75%**	**92,03%**	93,31%	91,21%	93,41%	91,38%	*93,51%*	*91,44%*
Spanish-GSD	90,71%	88,03%	**92,32%**	**90,11%**	91,26%	87,76%	90,96%	87,65%	*89,37%*	*85,17%*
Swedish-LinES	86,07%	81,86%	88,16%	84,55%	89,43%	85,98%	**90,05%**	**86,91%**	*89,63%*	*86,21%*
Swedish-Talbanken	89,63%	86,61%	92,42%	90,16%	92,92%	90,64%	**93,03%**	**90,90%**	*92,74%*	*90,40%*
Ukrainian-IU	88,29%	85,25%	91,65%	89,36%	93,40%	91,37%	**93,89%**	**91,89%**	*93,35%*	*91,30%*
Average	88.88%	85.60%	**91.10%**	**88.26%**	90.37%	86.96%	90.72%	87.42%	*90.48%*	*87.16%*

the two lies in UDPipe2.0++'s integration of BERT transformer and other word/character embeddings for input sentence representation, resulting in a noticeable boost in efficacy across all treebanks. It's worth noting that both methods rely on gold-standard tokenization. Therefore, our initial experiments also adopted this simplification for a consistent comparison. Then, we report our architectures of `U-DepPLLaMA` with 7 and 13 billion parameters, respectively, both based on a Gold Standard tokenization of the

input text. The $7B$ version shows improvements for some treebanks, such as *Czech-CLTT*, *Dutch-LassySmall*, *German-GSD*, and *Italian-PoSTWITA*, with on-par performance for other Romance languages but drastic drops for languages that are significantly different from English and its derivatives. The $13B$ version shows better results as the size of parameters increases, reaching top performance for Bulgarian, Croatian, Czech, *French-GSD*, Italian, Norwegian, Romanian, Russian, Serbian, Swedish, and Ukrainian. The performance drop is especially pronounced in languages like Korean, Chinese, and Finnish, which differ greatly from English, LLaMA's primary training language. While UDPipe++ remains consistently effective across most languages, in *Chinese-GSD*, `U-DepPLLaMA` lags behind UDPipe++ by over 4%, and nearly 10% for *Korean-GSD*. Exploring models with balanced linguistic exposure during pre-training might be insightful, but their effectiveness, even with limited language-specific data, is notable. Finally, we report the performance of our `U-DepPLLaMA 7B End-to-End`, which solves the task of Dependency Parsing in an end-to-end manner: that is, no Gold Standard tokenization is required. It simply takes the sentence as input and produces the bracketed output directly, ready to be transformed into a Dependency Tree. This process is computed all at once during the inference of the unique model for all the languages we trained it on. Its results are reported in *italics* as this model is solving a more complex task since it does not rely on GS tokenization. The model without gold standard tokenization not only matches but even surpasses the results of the 7B model with gold standard tokenization. This suggests that the language model might be more comfortable with unaltered real-world text, free from artificial tokenization adjustments. Additionally, the added complexity seems to mitigate potential overfitting effects, prompting the model to perform better.

In the last row of Table 1, we showcase mean UAS and LAS scores for each model. While UDPipe2.0++ remains the leading solution, our models closely compete. We contrast UDPipe's task-specific, feature-engineered approach with our universal LLaMA Language Model that generates DP Trees consistently across languages without architectural adjustments. In an evaluation across 50 treebanks, both architectures topped in 25 cases, revealing the potential of embedding expertise into a Language Model for synthetic outputs. Yet, the computational demands of the LLaMA models, both in training and inference, are considerably higher than UDPipe, given the higher number of involved parameters. Despite the obtained UAS/LAS, to make `U-DepPLLaMA` applicable, the tree output by the model must be a well-structured tree structure. In 0.45% of the sentences generated by the 7B parameter model, this isn't the case, leading to those sentences being discarded entirely. This drops to 0.31% for the 13B model. Without employing gold standard tokenization, the percentage of incorrect sentences for the 7B model rises from 0.45% to 1.44%. We anticipate larger models might mitigate this issue, though further experiments are needed. Occasionally, the model experienced hallucination issues. While it doesn't add new terms, it misses some or switches the word order, particularly in non-projective tree cases (as detailed in Appendix A). Correcting this typically involves realigning words to the original sentence order, but retaining word connections and relationships produced by the `U-DepPLLaMA`. For missing words, connecting the term to the tree's root (always present) suffices. However, such recovery measures are needed in under 7% of instances. Still, more in-depth manual analysis is essential to determine whether the primary cause of this swap is consistently due to non-projectivity or not.

Error Analysis. Initially, we seek to understand the system's error magnitude to gauge its potential influence on analyses or systems leveraging syntactic information. With the

$13B$ U-DepPLLaMA model, 60% of sentences display no UAS errors, and about 50% are error-free in LAS metrics. Considering that sentences typically contain an average of 17 words, it's notable that nearly 80% of sentences have at most one error, emphasizing the model's resilience. Conversely, sentences with more than five incorrectly linked words represent roughly 6%. For a comprehensive breakdown, please consult Appendix C.

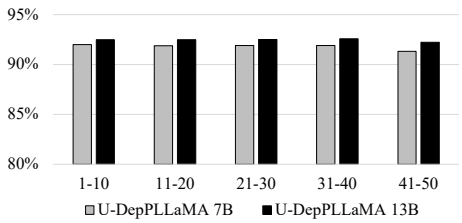


Figure 3
UAS vs Sentence Length

Inspired by studies such as (McDonald and Nivre 2007), we delved into assessing the system's performance based on sentence complexity, starting with an analysis centered on sentence length. In Figure 3, we display the UAS scores for our $7B$ and $13B$ parameter models. Sentences are categorized into length-based bins, ranging from 1 to 10 words and expanding in increments of ten, encompassing over 99% of the dataset up to the bin containing sentences with $50 - 60$ words. Interestingly, the UAS remains fairly stable across sentence lengths, averaging about 91% for the $7B$ model and 92% for the $13B$ model[7]. This consistency indicates that our models maintain robust parsing performance, even for longer, structurally complex sentences. The LLaMA's capacity to sustain these UAS scores across varied sentence lengths, being an Autoregressive Generative Model that processes sentences in their entirety, underlines its comprehensive and robust processing abilities.

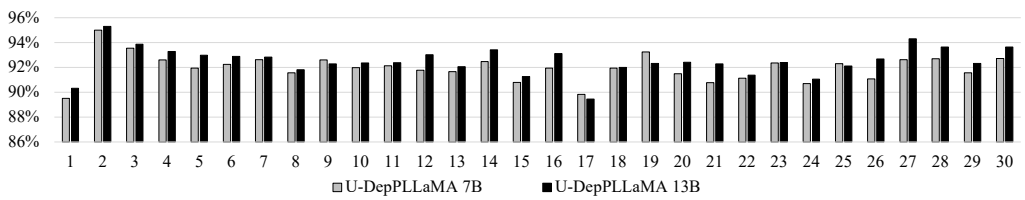


Figure 4
UAS versus Length of the Dependency Arc

Another pivotal aspect concerning complexity is the system's quality relative to the length of syntactic relations. This becomes crucial in assessing if a LLaMA-based model can effectively manage long-range dependencies, leveraging an attention mechanism spanning the entire sentence. As illustrated in Figure 4, while there's a slight performance drop for relations involving immediate adjacent words, the model displays a remarkably consistent performance, virtually always hovering between 91% and 93%, regardless of the relation distance. This consistency underscores the model's adeptness at managing varying dependency lengths. The trend for LAS closely mirrors this, as

7 The average UAS in Table 1 reflects the mean accuracy over the 50 datasets, while Figure 3 shows an average based on all sentences, not adjusted for the varying sizes of individual datasets.

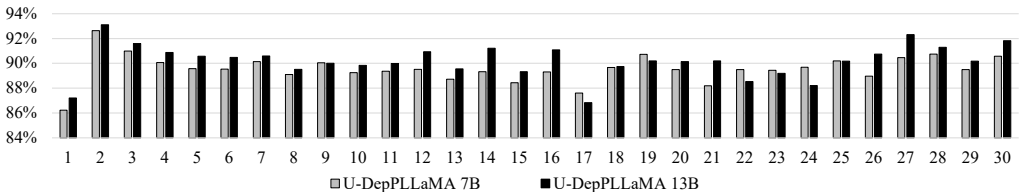


Figure 5
LAS versus Length of the Dependency Arc

depicted in Figure 5. However, a marginally steeper decline is evident, hinting that the model, while apt at establishing the syntactic connection, occasionally misinterprets the syntactic purpose of a word. Intriguingly, the 13B model shines particularly in longer distances, as evidenced by the minimal difference between the two models for short relations and a more pronounced divergence for extended ones.

Linguistic Analysis. In Table 2 we show a quantitative analysis concerning the types of relations that need to be predicted, along with the number and percentage of errors generated by our models. First and foremost, we encounter the PUNCT relation, accounting for 24% and 25% of the errors. This suggests issues with correctly associating punctuation marks like periods and commas with the words they depend on. It is worth noting that such errors may be considered marginal and could be reasonably overlooked. Of greater significance is the misprediction of the ROOT relation, occurring in 6% of cases. These errors entail that the entire sentence may be deemed incorrect, as the ROOT serves as the pivotal word from which the entire sentence's meaning derives.

Table 2
Top errors (plain numbers and percentages) concerning the relation for the two sizes of the U-DepPLLaMA models, cut above 5%. These constitute almost 75% of the errors from our models.

Relation	U-DepPLLaMA-7B		U-DepPLLaMA-13B	
	#errors	percentage	#errors	percentage
PUNCT	17,345	24%	16,538	25%
NMOD	7,980	11%	7,371	11%
ADVMOD	6,970	10%	6,579	10%
CONJ	6,150	9%	5,476	8%
OBL	5,954	8%	5,469	8%
NSUBJ	4,557	6%	4,229	6%
ROOT	3,952	6%	3,717	6%

In our qualitative analysis of the 13B model's LAS evaluation, we found that of about 80,000 misassigned words, 55% had the correct label. Notably, this accuracy is achieved without using specialized Part-of-Speech features or grammatical generalizations, which are used in many parsers such as UDPipe. These capabilities are implicitly embodied by the language model and enhanced during fine-tuning. Concerning the real Part-of-Speech (PoS) tags of these words, the most prevalent errors (occurring in over 5% of words) are found in nouns (23.4%), punctuation symbols (19%), verbs (12.8%), and adverbs (8.1%). For nouns, out of the 18,700 noun-related errors, 40% have the correct label but are incorrectly attached to a different word. These attachment errors exhibit irregular patterns in terms of the types of incorrect relations. The most common mistake, at 15%, is caused by the erroneous attachment of nouns and verbs, resulting in

the interchange of "OBL" with "NMOD". This is primarily attributed to arc introduction errors, although the label is coherent with respect to the incorrect attachment. In the case of verbs, nearly 40% of errors involve incorrect attachments while the label remains correct. In the remaining cases, there is little regularity in the errors: in 5% of cases, a main verb (ROOT) is swapped with a subordinate verb, resulting in an attachment error, with the "ROOT" label being replaced by "CONJ", and in 3.9% of cases, an error occurs between "ROOT" and "PARATAXIS". For adverbs, in 63% of cases, the label is correct, but the attachment is not. The remaining cases exhibit varying irregularities, with the most common being the interchange between "ROOT" and "ADVMOD", although this only occurs in 3% of cases.

4. Conclusions

In this investigation, we introduce U-DepPLLaMA, a framework for employing Large Language Models (LLMs) in the domain of syntactic analysis through Dependency Parsing. By conceptualizing parsing as a sequence-to-sequence task, our model transforms input sentences into bracketed forms, subsequently mapped onto comprehensive dependency parse trees. Evaluated across 50 datasets spanning 26 languages from the Universal Dependency Treebank, our approach consistently demonstrates competitive performance, underscoring its efficacy without necessitating task-specific configurations. The model demonstrates robustness against varying sentence complexities and lengths, indicating a strong capacity for understanding sentence relationships. While our results validate the model's adeptness in syntactic processing, there remains a need for deeper investigation into the latent syntactic knowledge these models harbor, as suggested by studies like (Hewitt and Manning 2019).

Future research directions include probing the benefits of integrating explicit syntactic information during LLM pre-training or fine-tuning. Such exploration could further refine their interpretative abilities, particularly in tasks requiring a nuanced understanding of language structures. Moreover, examining the impact of syntactic information in multimodal models that involve semantic understanding could significantly enhance the robustness and versatility of LLMs, as demonstrated by (Hromei et al. 2024). Another interesting direction should focus on adapting U-DepPLLaMA for low-resource languages and exploring cross-lingual learning techniques. Investigating different attention mechanisms within LLMs could also improve the parsing of complex or lengthy sentences. Applying these models to tasks like semantic analysis or text summarization offers a promising direction, leveraging their syntactic understanding.

Acknowledgments

We would like to thank the "Istituto di Analisi dei Sistemi ed Informatica - Antonio Ruberti" (IASI) for supporting the experimentations through access to dedicated computing resources. Claudiu Daniel Hromei is a Ph.D. student enrolled in the National Ph.D. in Artificial Intelligence, XXXVII cycle, course on *Health and life sciences*, organized by the Università Campus Bio-Medico di Roma. We acknowledge financial support from the PNRR MUR project PE0000013-FAIR.

References

Aharoni, Roee and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140, Vancouver, Canada, July. Association for Computational Linguistics.

Ainslie, Joshua, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: training generalized multi-query transformer models from

multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4895–4901. Association for Computational Linguistics.

Chen, Danqi and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

Conneau, Alexis, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia, July. Association for Computational Linguistics.

Croce, Danilo, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

de Marneffe, Marie-Catherine, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland, May. European Language Resources Association (ELRA).

Dettmers, Tim, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

Dozat, Timothy and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Filice, Simone, Giuseppe Castellucci, Giovanni Da San Martino, Aless, ro Moschitti, Danilo Croce, and Roberto Basili. 2018. Kelp: a kernel-based learning platform. *Journal of Machine Learning Research*, 18(191):1–5.

Hewitt, John and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. 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 4129–4138, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Hromei, Claudiu Daniel, Danilo Croce, Valerio Basile, and Roberto Basili. 2023. ExtremITA at EVALITA 2023: Multi-task sustainable scaling to large language models at its extreme. In *Proceedings of the Eighth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2023)*, Parma, Italy, September. CEUR.org.

Hromei, Claudiu Daniel, Daniele Margiotta, Danilo Croce, and Roberto Basili. 2024. MM-IGLU: Multi-modal interactive grounded language understanding. In Nicoletta Calzolari, Min-Yen Kan, Veronique Hoste, Alessandro Lenci, Sakriani Sakti, and Nianwen Xue, editors, *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 11440–11451, Torino, Italia, May. ELRA and ICCL.

Hu, Edward J., Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Kondratyuk, Dan and Milan Straka. 2019a. 75 languages, 1 model: Parsing Universal Dependencies universally. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China, November. Association for Computational Linguistics.

Kondratyuk, Daniel and Milan Straka. 2019b. 75 languages, 1 model: Parsing universal dependencies universally. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2779–2795. Association for Computational Linguistics.

Kübler, Sandra, Ryan T. McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Li, Zuchao, Jiaxun Cai, Shexia He, and Hai Zhao. 2018. Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA, August. Association for Computational Linguistics.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

McDonald, Ryan and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 122–131, Prague, Czech Republic, June. Association for Computational Linguistics.

McDonald, Ryan, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal Dependency annotation for multilingual parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 92–97, Sofia, Bulgaria, August. Association for Computational Linguistics.

Nivre, Joakim. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Radford, Alec, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:140:1–140:67.

Sherborne, Tom and Mirella Lapata. 2022. Zero-shot cross-lingual semantic parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4134–4153, Dublin, Ireland, May. Association for Computational Linguistics.

Straka, Milan and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada, August. Association for Computational Linguistics.

Straka, Milan, Jana Straková, and Jan Hajic. 2019. Evaluating contextualized embeddings on 54 languages in POS tagging, lemmatization and dependency parsing. *CoRR*, abs/1908.07448.

Tesnière, Lucien. 1959. *Éléments de syntaxe structurale*. Klincksieck, Paris.

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models.

Touvron, Hugo, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu,

Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models.
Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Appendix A: Generating dependency trees from dependency graphs

The recursive pseudo-code utilized to derive the parenthetic form from a dependency graph is detailed in Table 1. It assumes that dependency graphs are loaded using the conllu[8] python library.

The code defines two methods:

1. `tree2string_grct`: This function accepts a dependency tree (for instance, loaded using the `collu` library) and yields the parenthetic form we have adopted in this study.
2. `tree2list_grct`: This function recursively constructs a string in parenthetic form for a given subtree rooted at 'node'. It begins by adding the dependency relation. The boolean flag `is_deprel_written` ensures that the current node's token form (`node.token["form"]`) is inserted in the correct position and only once. For nodes without children, the lexical form is directly appended. For nodes with children, the function checks the order of tokens and ensures they are appropriately nested, with recursive calls to process the entire tree. Finally, it concludes the string representation for the subtree.

While the aforementioned function handles projective graphs seamlessly, it may face challenges with non-projective ones. Figure 1 showcases the non-projective dependency graph for the sentence "*I guess you get what you pay for*." However, as illustrated by the tree in 2, the tree can be generated straightforwardly. It's noteworthy that the phrase "*what you pay for*" was reordered to "*what for you pay*". During fine-tuning the model, such derived non-projective trees are simply provided to the model (though they represent a minor fraction of the entire Universal Dependency set). During tagging, realigning the sentences to their original order while retaining the dependency relations will be sufficient.

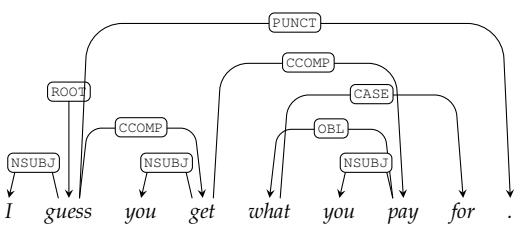


Figure 1
Example of a non-projective dependency graph associated with the sentence "*I guess you get what you pay for.*".

8 https://pypi.org/project/conllu/

Table 1
Pseudocode in python to generate the parenthetic form

```
# This method takes into input a dependency graph, e.g., loaded with the collu
                                        library, and generates the parenthetic
                                        form
def tree2string_grct(node):
  my_list = []
  tree2list_grct(node, my_list)
  return " ".join(my_list)

# This method takes in input a subtree rooted at 'node' and generates a string
                                        in a parenthetic form
def tree2list_grct(node, mylist = []):
  # First the dependency relation is added
  mylist.append("[" + node["deprel"])
  # Boolean flag used to ensure that the current node's token form (node.token
                                          ["form"]) is added once in the
                                          correct position
  is_deprel_written = False
  # No children, so the lexical node can be written
  if len(node.children) == 0:
      mylist.append("[" + node["form"] + "]")

  # For each child, write the node's token form if needed or recursively call
                                          the tree2list_grct method
  for i, child in enumerate(node.children):
      if child.token["id"] > node.token["id"] and not is_deprel_written:
          mylist.append("[" + node.token["form"] + "]")
          is_deprel_written = True

      # Recursive call
      tree2list_grct(child, mylist=mylist)

      if i == len(node.children) - 1 and not is_deprel_written:
          mylist.append("[" + node.token["form"] + "]")
          is_deprel_written = True
  # Close the string associated with the subtree
  mylist.append("]")
```

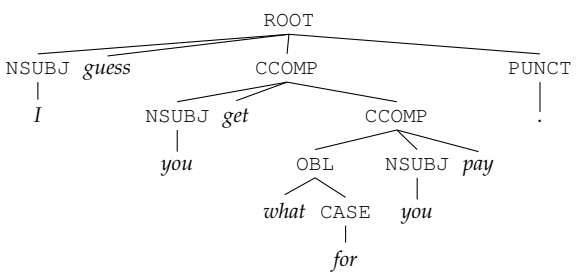


Figure 2
The syntactic parse tree associated with the dependency graph from Figure (1).

Appendix B: Distribution of the languages for pre-training LLaMA

In assessing the data distribution disparities between the pre-training phase of LLaMA and our fine-tuning datasets (Train30k, Dev, and Test sets), we provide an illustrative breakdown in Table 2.

Table 2
Data distribution

Code	Language	LLaMA	Train30k	Dev	Test
en	English	**89,70%**	5,09%	6,39%	6,32%
unk	unknown	8,38%	-	-	-
de	German	0,17%	3,52%	1,35%	1,57%
fr	French	0,16%	4,75%	4,91%	2,75%
sv	Swedish	0,15%	1,82%	2,54%	3,49%
zh	Chinese	0,13%	1,02%	0,85%	0,81%
ru	Russian	0,13%	7,45%	12,12%	12,85%
es	Spanish	0,13%	7,27%	5,17%	3,46%
nl	Dutch	0,12%	4,61%	2,36%	2,37%
it	Italian	0,11%	5,17%	2,35%	2,11%
ja	Japanese	0,10%	1,82%	0,86%	0,89%
pl	Polish	0,09%	5,07%	4,69%	4,55%
pt	Portuguese	0,09%	4,59%	3,00%	2,71%
vi	Vietnamese	0,08%	0,36%	1,35%	1,29%
uk	Ukrainian	0,07%	1,35%	1,10%	1,39%
ko	Korean	0,06%	6,99%	5,10%	5,28%
ca	Catalan	0,04%	3,35%	2,89%	2,97%
sr	Serbian	0,04%	0,75%	0,79%	0,79%
cs	Czech	0,03%	**7,65%**	19,14%	19,66%
fi	Finnish	0,03%	6,94%	5,48%	5,51%
hu	Hungarian	0,03%	0,23%	0,75%	0,72%
id	Indonesian	0,03%	1,14%	0,95%	0,90%
no	Norwegian	0,03%	7,63%	7,28%	7,26%
ro	Romanian	0,03%	4,08%	3,05%	2,87%
bg	Bulgarian	0,02%	2,27%	1,89%	1,80%
da	Danish	0,02%	1,12%	0,95%	0,91%
hr	Croatian	0,01%	1,78%	1,44%	1,70%
sl	Slovenian	0,01%	2,18%	1,24%	3,06%

During the pre-training phase of LLaMA2, English dominates, accounting for nearly 90% of the data. This stark bias towards English is contrasted by the top language in our Train30k set, which is Czech, comprising only 7.65%.

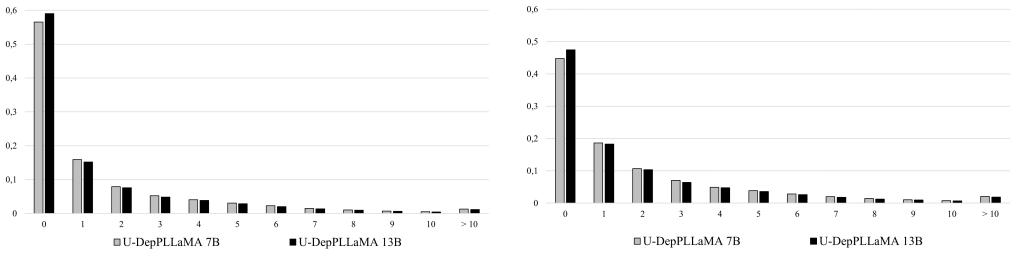
Each row in Table 2 presents a language considered in this study. The second column highlights the proportion of that language's representation in LLaMA2's pre-training data. The subsequent columns provide the percentages of examples for that particular language in our Train30k, Dev, and Test datasets.

It's important to clarify the naming convention of Train30k. Datasets for languages with more than 30,000 examples were uniformly sampled across all available datasets. The goal was to ensure a maximum of 30,000 examples per language, with each source dataset being equally represented. This strategy addresses potential biases and ensures a balanced representation across various datasets.

Appendix C: Detailed Analysis of Sentence-level Errors in UAS and LAS Metrics

In Figure 1, we present a comprehensive visualization of the percentage of sentences in our test set of 62,069 sentences, which exhibit specific counts of errors. This distribution allows for a granular understanding of the model's syntactic accuracy.

Panel a) focuses on the Unlabeled Attachment Score (UAS) metric. Here, it's remarkable to observe that sentences with no UAS errors comprise a significant 60% of the test set when the $13B$ size `U-DepPLLaMA` model is employed. Moreover, when



(a) (b)

Figure 1
UAS (a) and LAS (b) for sentences with an increasing number of errors

aggregating sentences that exhibit 0 or 1 error, this rises to nearly 80%, an impressive feature, especially considering the average sentence length stands at 17 words.

Subsequently, panel b) provides insights into the Labeled Attachment Score (LAS) metric. A direct comparison between the two panels readily highlights the superior performance of the model with 13 billion parameters. Furthermore, as expected, the LAS figures are generally lower, a direct result of its being a subset of the instances where the attachment is incorrect (captured under UAS) and additionally carries an erroneous syntactic relation label.