

Arabic Morphosyntactic Tagging and Dependency Parsing with Large Language Models

Mohamed Adel¹ Bashar Alhafni² Nizar Habash¹

Computational Approaches to Modeling Language Lab

¹New York University Abu Dhabi

²Mohamed bin Zayed University of Artificial Intelligence

{mohamed.adel,nizar.habash}@nyu.edu, bashar.alhafni@mbzuai.ac.ae

Abstract

LLMs perform strongly across NLP, but their ability to produce explicit grammatical analyses remains unclear. Arabic provides a challenging testbed due to its rich morphology and orthographic ambiguity, which create strong morphology–syntax interactions. We present a unified evaluation of LLMs on Arabic morphosyntactic tagging and dependency parsing, covering pre-tokenized, raw-text, and cascaded settings. We compare zero-shot prompting with retrieval-based in-context learning. Relevant demonstrations substantially improve performance. The strongest LLMs approach supervised tagging and parsing systems; however, they require substantial annotated data for demonstration retrieval and considerable computational resources. We make all code and data used in this paper publicly available.¹

1 Introduction

LLMs achieve strong performance across many NLP tasks (OpenAI et al., 2024; Gemini Team et al., 2025; Grattafiori et al., 2024; Yang et al., 2025). However, it remains unclear whether they can reliably produce structured linguistic analyses such as morphosyntactic features and dependency trees. Prior work shows that LLMs can recover some linguistic patterns when appropriately prompted, but performance remains uneven when tasks require explicit morphosyntactic constraints or precise structured predictions such as parsing (Blevins et al., 2023; Anh et al., 2024; Miaschi et al., 2024; Tian et al., 2024; Ezquerro et al., 2025).

Morphologically rich languages provide a particularly challenging testbed for evaluating the structural competence of LLMs (Anh et al., 2024; Ismayilzada et al., 2025). Although recent work has increasingly examined the linguistic capabilities of LLMs, morphosyntactic tagging and dependency

parsing are often evaluated separately, despite evidence that morphological and POS information substantially affect parsing and that joint evaluation can reveal weaknesses hidden in isolated or gold-preprocessed settings (Tsarfaty et al., 2012; Zhang et al., 2015; Lu et al., 2019; More et al., 2019; Temesgen et al., 2025; Matsuda et al., 2025).

In this paper, we evaluate whether instruction-tuned LLMs can perform morphosyntactic and syntactic analysis for Standard Arabic, a language whose rich inflectional morphology, cliticization, and tokenization ambiguities create strong morphology-syntax interactions (Habash and Rambow, 2005; Green and Manning, 2010; Marton et al., 2013). We evaluate instruction-tuned LLMs on two tasks: **(i) morphosyntactic tagging** with 14 morphological features and **(ii) labeled dependency parsing**, comparing zero-shot prompting with retrieval-based in-context learning (ICL) (Brown et al., 2020; Liu et al., 2022; Min et al., 2022). We also examine whether model-predicted morphology can improve dependency parsing. Results show that prompt design and example retrieval strongly affect performance: open-weight models struggle in zero-shot but improve with retrieved demonstrations, while proprietary LLMs approach specialized supervised taggers and parsers.

Our contributions are as follows:

- A unified evaluation of morphosyntactic tagging and dependency parsing across pre-tokenized, raw-text, and cascaded settings.
- An analysis of prompt design and retrieval-based ICL for structured linguistic prediction.
- A detailed error analysis identifying which aspects of Arabic tagging and parsing LLMs capture reliably and which remain difficult.
- A SOTA extension of CamelParser, a supervised Arabic dependency parser (Elshabrawy et al., 2023).²

¹<https://github.com/CAMEL-Lab/arabic-morphosyntax-llm>

²https://github.com/CAMEL-Lab/camel_parser

2 Related Work

2.1 Arabic Tagging and Parsing

Arabic Morphosyntactic Tagging Arabic morphological modeling has proven useful for various downstream tasks, including machine translation (Sadat and Habash, 2006; El Kholy and Habash, 2010), grammatical error correction (Alhafni et al., 2023), and text rewriting (Alhafni et al., 2022). A wide range of approaches have been proposed for Arabic morphosyntactic tagging (Diab et al., 2004; Habash and Rambow, 2005; Pasha et al., 2014; Abdelali et al., 2016; Zalmout and Habash, 2017; Inoue et al., 2017; Zalmout et al., 2018; Zalmout and Habash, 2019, 2020; Khalifa et al., 2020). More recently, Inoue et al. (2022) introduced BERT-based models that achieve state-of-the-art (SOTA) performance on MSA and multiple Arabic dialects, and were later integrated into CAMEL Tools (Obeid et al., 2020). Much of this work relies on the Penn Arabic Treebank (PATB), which serves as a central benchmark for Arabic morphological and syntactic annotation (Maamouri et al., 2004). To the best of our knowledge, the use of LLMs for Arabic morphosyntactic tagging has not been systematically investigated. In this work, we evaluate LLMs on PATB and compare them against CAMEL Tools as a strong benchmark system.

Arabic Dependency Parsing Arabic dependency parsing has benefited from the development of several treebanks (Smrž et al., 2002; Maamouri et al., 2004; Habash and Roth, 2009; Dukes and Buckwalter, 2010; Taji et al., 2017; Nivre et al., 2020; Al-Ghamdi et al., 2021). More recently, Habash et al. (2022) introduced the CAMEL Treebank (CAMELTB), a multi-genre Arabic dependency treebank created according to the Columbia Arabic Treebank (CATiB) formalism (Habash and Roth, 2009). A wide range of parsing approaches have been proposed for Arabic (Green and Manning, 2010; Marton et al., 2013; Shahrour et al., 2016; Straka, 2018; Kankanampati et al., 2020; Al-Ghamdi et al., 2023; Elshabrawy et al., 2023). Among these, Elshabrawy et al. (2023) introduced CAMELParser2.0, a BERT-based parser that currently achieves state-of-the-art performance on CAMELTB. In this work, we benchmark LLMs on CAMELTB and compare their performance against CAMELParser2.0.

Recently, Al-Sabahi et al. (2026) explored the use of LLMs for Arabic syntactic analysis. They

evaluate LLMs on a carefully curated dataset targeting specific grammatical phenomena and frame the task as morphosyntactic annotation over gold-tokenized text. Our work differs in several ways. First, while their task predicts a morphosyntactic label for each token, we use a 14-dimensional morphosyntactic representation and extend the task to full dependency parsing with labeled relations. Second, their dataset assumes gold tokenization, whereas we evaluate parsing with both gold and model-predicted tokenization. Finally, their dataset targets specific constructions, while our experiments use multi-genre treebank data reflecting naturally occurring Arabic text. We also evaluate a wider range of prompting strategies and compare LLMs against a SOTA parsing system.

2.2 LLMs and Linguistic Knowledge

Recent work has examined whether LLMs encode linguistic knowledge that can be explicitly elicited through prompting (Blevins et al., 2023; Miaschi et al., 2024). A parallel line of work evaluates LLMs on morphological generalization, showing mixed results, particularly for morphologically rich languages (Weissweiler et al., 2023; Anh et al., 2024; Weller-Di Marco and Fraser, 2024; Waldis et al., 2024; Ismayilzada et al., 2025; Temesgen et al., 2025). Other studies investigate syntactic structure and parsing, finding that while LLMs capture aspects of shallow syntax, full dependency parsing remains challenging (Tian et al., 2024; Ezquerro et al., 2025; Zhang et al., 2025).

ICL enables LLMs to perform new tasks from a small set of demonstrations without parameter updates (Brown et al., 2020). Prior work shows that its effectiveness depends strongly on the choice and ordering of examples, with retrieval-based selection often outperforming random sampling and demonstrations helping models infer both the task and the expected output format (Liu et al., 2022; Min et al., 2022; Rubin et al., 2022). Motivated by this line of work, we evaluate whether instruction-tuned LLMs can produce explicit morphosyntactic analyses and labeled dependency trees for Arabic under both zero-shot and in-context settings, and how different example-selection strategies affect performance. In particular, we study how prompting and example selection affect the ability of LLMs to generate structured linguistic outputs for Arabic morphosyntactic tagging and dependency parsing.

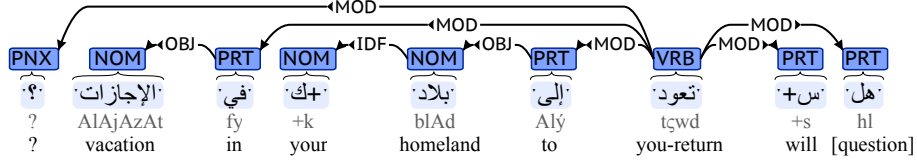


Figure 1: Example **dependency parsing** output for “will you return to your homeland during vacation?”. The model predicts a syntactic head (0 = root) and CATiB dependency relation for each token.

Words	؟	الإجازات	في	بلادك	إلى	ستعود	هل
Transliteration	?	AlĀjAzAt	fy	blAdk	Āly	stçwd	hl
Gloss	?	vacation	in	your-homeland	to	will-you-return	[question]
pos	punc	noun	prep	noun	prep	verb	part_interrog
prc3	na	0	0	0	0	0	0
prc2	na	0	0	0	0	0	0
prc1	na	0	0	0	0	0	0
prc0	na	Al_det	na	0	na	fut_part	na
asp	na	na	na	na	na	i	na
vox	na	na	na	na	na	a	na
mod	na	na	na	na	na	i	na
gen	na	f	na	m	na	f	na
num	na	p	na	p	na	s	na
stt	na	d	na	c	na	na	na
cas	na	g	na	g	na	na	na
per	na	na	na	na	na	3	na
enc0	na	0	0	poss_2ms	0	0	0

Table 1: Example **morphosyntactic tagging** output for “will you return to your homeland during vacation?”. The model predicts a 14-feature morphosyntactic analysis for each token.

3 Tasks

3.1 Task 1: Morphosyntactic Tagging

Task Definition In morphosyntactic tagging, the model receives an Arabic sentence as a fixed sequence of tokens and outputs a complete morphosyntactic analysis for each token. Specifically, it predicts values for 14 categorical features: part-of-speech (pos), aspect (asp), mood (mod), voice (vox), person (per), form gender (gen), form number (num), case (cas), state (stt), proclitics (prc3, prc2, prc1, prc0) and the pronominal enclitic (enc0). See example in Table 1. This feature inventory follows common Arabic morphological disambiguation practice (Pasha et al., 2014; Obeid et al., 2020).

Prompt Formulation The prompt (Appendix A.1) provides the sentence as a fixed list of whitespace/punctuation tokens and supplies the target feature inventory, with brief descriptions and allowed values. The model returns a complete morphosyntactic analysis for each token without changing the given list or its order.

3.2 Task 2: Dependency Parsing

Task Definition In dependency parsing, the model predicts a labeled dependency tree over a sentence’s tokens. For each token, it identifies the syntactic head, using index 0 for the root, and assigns a dependency relation label. We adopt the CATiB representation (Habash and Roth, 2009; Habash et al., 2022). CATiB follows the ATB tokenization standard, which separates all clitics except the definite article (Figure 1). We evaluate parsing under two input conditions: gold tokenization and raw text (end-to-end).

Prompting Formulation The parsing prompts (Appendix A.2) provide the input sentence, the CATiB dependency-label inventory, and annotation guidelines. The model either preserves the supplied gold sequence of morphological tokens or derives the morphological tokenization from raw text according to CATiB conventions before producing the dependency analysis.

3.3 Task Integration

We connect the two tasks on the same PATB sentences and their converted CATiB dependency trees.

In the independent condition, morphology and raw-text parsing are predicted separately; in the cascaded condition, the same model’s morphology predictions for the input tokens are also provided to the parser as auxiliary evidence. The cascaded prompt reuses the two task prompts (Appendix A.3). We also evaluate a unified NUDAR/UD formulation for UPOS, morphology, and dependencies (Appendix C.5), but center PATB/CATiB because they are linguistically motivated for Arabic, supported by manually annotated treebanks, and widely used in Arabic NLP.

4 Methodology

We evaluate LLMs on two structured tasks: **morphosyntactic tagging** and **dependency parsing** (§3), comparing zero-shot prompting with ICL and varying example selection strategies.

In-Context Learning Setup We evaluate ICL by varying both the number of examples (shots) and the strategy used to select them. For each test instance x (from the dev or test splits), we retrieve examples from a fixed candidate pool $\mathcal{D}_{\text{train}} = \{(x_i, y_i)\}$ consisting only of training data (§5). Selection relies only on the input side x_i (raw text or token sequence, depending on the task and setting), ensuring that no dev or test labels influence retrieval. Formally, let k denote the number of examples prepended to x . We vary two factors: the **shot count** $k \in \{0, 1, 3, 5, 10\}$, where $k = 0$ corresponds to zero-shot prompting, and the **selection method** m , which determines which k examples from $\mathcal{D}_{\text{train}}$ are included in the prompt. Each configuration (k, m) therefore defines a prompt $P(x; k, m)$ for each test instance x .

Selection Methods We explore two example-selection methods based on surface- and semantic-level similarity. **Surface similarity** retrieves inputs that are close at the character or subword level, which can capture shared orthographic patterns and morphological features in Arabic even when exact word overlap is limited. We implement this using chrF++ (Popović, 2015, 2017), following Ginn et al. (2024), who showed that substring-based similarity better reflects morphological relatedness than word-level overlap. **Semantic similarity** retrieves inputs that are close in meaning even when surface forms differ. We implement this by embedding each input using CAMELBERT-MSA (Inoue et al., 2021) and ranking candidates by cosine similarity.

After computing similarity scores, we consider two retrieval regimes: **high-similarity** (near-neighbor) and **low-similarity** (far-neighbor). This allows us to distinguish improvements driven by content similarity from those resulting from general format calibration.

In addition, we evaluate **random selection**, where k examples are sampled randomly from $\mathcal{D}_{\text{train}}$. Overall, this yields five example-selection methods: random sampling, and for each similarity measure (surface and semantic) we evaluate both high- and low-similarity retrieval.

5 Experimental Setup

This section describes the experimental setup used to evaluate LLMs on Arabic morphosyntactic tagging and dependency parsing. We present the datasets used in our experiments, the evaluated systems, and the evaluation metrics for each task.

5.1 Data

Morphosyntactic Tagging We evaluate morphosyntactic tagging on the PATB (MSA, newswire) (Maamouri et al., 2004) following the morphological tag representations of Inoue et al. (2022), and Train/Dev/Test splits based on Diab et al. (2013). Summary statistics are in Table 3.

Dependency Parsing We evaluate dependency parsing in two data configurations. For the gold-tokenized and raw-text parsing experiments, we use CAMELTB (Habash et al., 2022), which is annotated in the CATiB dependency representation (Habash and Roth, 2009). We adopt the Train/Dev/Test splits following Habash et al. (2022). To compare independent and cascaded processing, we use CATiB dependency trees converted from the PATB constituency trees (Taji et al., 2017). These trees correspond to the original PATB Test sentences and are therefore aligned with their PATB morphosyntactic annotations. We refer to this paired evaluation data as PATB–CATiB. Summary statistics are in Table 3, and Table 2 summarizes the task coverage of each data configuration.

5.2 Systems

LLMs We benchmark multiple state-of-the-art LLMs with strong multilingual capabilities, including Arabic. Our evaluation includes two open-weight models, Llama 4 Scout (Meta, 2025) (Llama4) and Qwen3-Next-80B-A3B-Thinking (Yang et al., 2025) (Qwen3), as well

Data	Morphosyntactic Tagging	Dependency Parsing
PATB	Yes	–
CAMeL TB	–	Yes
PATB–CATiB	Yes	Yes

Table 2: Task coverage of PATB, CAMeL TB, and PATB–CATiB. In PATB–CATiB, morphosyntactic tagging is assessed on PATB words and parsing on CATiB tree tokens converted from the same PATB sentences.

Dataset	Sentences	Words
PATB Train	15,000	477,512
PATB Dev	1,986	63,137
PATB Test	1,963	63,172
CAMeL TB Train	9,397	133,459
CAMeL TB Dev	2,022	27,464
CAMeL TB Test	1,918	26,961

Table 3: Dataset statistics for the morphosyntactic tagging and dependency parsing benchmarks.

as two proprietary models, Gemini 3 Flash (Gemini Team et al., 2025) (**Gemini3**) and GPT-5.2 (OpenAI, 2026) (**GPT5.2**). Arabic-centric models (AL-LaM, Fanar, Jais) were not included because their available context lengths were less suitable for our long-context ICL setup (Bari et al., 2025; Fanar Team et al., 2025; Sengupta et al., 2023).

Supervised Baselines For morphosyntactic tagging, we use **CAMeL Tools** (Obeid et al., 2020), which was trained on PATB. For dependency parsing, we evaluate the released **CamelParser2.0**, trained on PATB and CAMeL TB (Elshabrawy et al., 2023), and **CamelParser2.1**, our decoding extension of the same checkpoint. CamelParser2.0 enforces exactly one root per input, whereas CamelParser2.1 uses projective multiroot decoding while continuing to require at least one root and to prohibit cycles and self-loops.

5.3 ICL Configuration

We use the Dev sets to select the best ICL configuration. First, all candidate LLMs are evaluated in the zero-shot setting on the Dev set for each task, alongside the supervised baselines. The strongest open-weight LLM under zero-shot performance is chosen as the ICL exploration model. Using this model, we explore the ICL configuration space by varying both the shot count k and the example-selection method described in §4. The best-performing configuration (k^*, m^*) on the Dev set is fixed and used to benchmark on the test sets.

5.4 Evaluation Metrics

Morphosyntactic Tagging Our primary tagging metric is **All Tags**, a strict token-level exact match score that counts a token as correct only if all 14 morphosyntactic features are predicted correctly. We use **All Tags** as the main score because it best reflects end-to-end correctness of the full feature structure. We also report **Tag F₁**, which captures partial correctness by aggregating matches over individual feature values across the 14-feature inventory (macro average).

Dependency Parsing Our primary parsing metric is **LAS** (Labeled Attachment Score), which measures the percentage of tokens whose predicted head and dependency relation label match the gold parse. We additionally report **UAS** (Unlabeled Attachment Score), which evaluates head correctness irrespective of the label, and **LS** (Label Score), which evaluates relation label correctness irrespective of the head. In the raw-text parsing condition, we also report **Tok F₁** (Tokenization F₁), which compares the aligned predicted and gold token sequences: matching aligned token forms indicate correct boundaries, while precision and recall penalize extra and missing splits. We use the CAMeL CoNLL evaluator.³

Joint PATB–CATiB Evaluation Here, joint means that morphology and parsing are evaluated on the same aligned PATB sentences, not that they are jointly trained or predicted simultaneously. Morphology is scored on the original PATB tokens using **All Tags** and **Tag F₁**, whereas tokenization and parsing are scored on the predicted and gold CATiB token sequences using **Tok F₁**, **UAS**, **LS**, and **LAS**. This keeps each task on its appropriate annotation unit.

6 Results

We first report the two tasks introduced in Section 4. For each task, we present Dev results used for model and ICL configuration selection, followed by Test results under zero-shot and best-ICL. We then connect the tasks in the joint PATB–CATiB evaluation.

6.1 Morphosyntactic Tagging

Baseline Table 4 reports zero-shot dev results. The supervised baseline (CAMeL Tools) is near ceiling and substantially outperforms all LLMs,

³https://github.com/CAMeL-Lab/camel_conll

Model	All Tags	Tag F ₁
Llama4	8.9	81.9
Qwen3	14.1	82.8
GPT5.2	39.9	89.7
Gemini3	48.4	92.6
CAMeL Tools	96.2	99.6

Table 4: **Morphosyntactic Tagging** results on the PATB Dev set (zero shot).

underscoring the difficulty of predicting complete 14-feature bundles for each token. Among LLMs, Gemini3 performs best, followed by GPT5.2. Although some open-weight models obtain relatively higher feature-level Tag F₁, their All Tags scores remain much lower, showing how small feature errors compound under exact bundle matching. We therefore select Qwen3, the strongest open-weight model on dev, for ICL analysis.

ICL Table 5 shows that ICL yields large gains even with very few demonstrations. With random 1-shot prompting, All Tags improves by 30.9 points over zero-shot. Retrieval quality matters at least as much as the number of shots: 1-shot highest chrF++ improves All Tags by 45.4 points, essentially matching 5-shot Random. More generally, high-similarity retrieval consistently outperforms low-similarity selection, and this gap persists as k increases. The best dev configuration is highest chrF++ with $k=10$, which improves All Tags by 63.9 points over zero-shot. We therefore use $(k^*, m^*) = (10, \text{highest chrF++})$ as the *best-ICL* setting for test evaluation. .

Test Results Table 6 shows that the selected ICL configuration produces large gains on All Tags for all LLMs. The largest absolute improvements are observed for the open-weight models, with gains of 62.6 points for Qwen3 and 49.7 points for Llama4, while the proprietary models improve by 46.5 and 37.8 points for GPT5.2 and Gemini3, respectively. These gains substantially narrow the gap to the supervised baseline in Tag F₁, but All Tags remains lower than CAMeL Tools, confirming that exact full-bundle prediction is considerably harder for LLMs than feature-level accuracy alone.

6.2 Dependency Parsing

Baseline Table 7 reports zero-shot dev results under gold tokenization. The supervised baselines remain strongest, with CamelParser2.1 improving over CamelParser2.0 through multiroot decoding.

Gemini3 is the best zero-shot LLM and Qwen3 is the strongest open-weight model. The large gap between proprietary and open-weight models in zero-shot highlights the difficulty of structured Arabic parsing without task-specific supervision. We therefore select Qwen3 for ICL analysis.

ICL Table 8 shows that ICL substantially improves parsing performance, but gains depend strongly on demonstration selection. With random 1-shot prompting, LAS improves by 17.4 points over zero-shot, indicating that even a single example helps adapt the model to the task. However, retrieval quality is crucial: at $k=1$, highest chrF++ improves LAS by 32.6 points, far exceeding low-similarity retrieval, which yields only marginal gains. This advantage persists as k increases, and highest chrF++ remains the strongest method throughout, reaching a total LAS improvement of 45.1 points at $k=10$. We therefore select $(k^*, m^*) = (10, \text{highest chrF++})$ as the *best-ICL* setting for test evaluation.

Gold Tokenization Test Results Under gold tokenization (Table 9), best-ICL yields large gains for all LLMs, with LAS improvements ranging from 9.9 to 46.9 points. The largest gains are observed for the open-weight models, especially Qwen3 (+46.9) and Llama4 (+39.1), while the proprietary models improve more modestly from stronger zero-shot baselines. Gemini3 under best-ICL slightly surpasses CamelParser2.0 (88.2 vs. 87.5 LAS), but CamelParser2.1 is stronger at 89.3 LAS.

Raw Text Test Results In the raw-text setting (Table 10), tokenization is a major source of zero-shot degradation, especially for open-weight models. Best-ICL substantially improves both tokenization and parsing, yielding Tok-F₁ gains of 33.7 and 38.3 points and LAS gains of 44.7 and 42.2 points for Qwen3 and Llama4, respectively. The proprietary models are considerably more robust in zero-shot, but still benefit from ICL, with smaller tokenization gains and LAS improvements of 24.0 points for GPT5.2 and 16.9 for Gemini3. Multiroot decoding raises the supervised raw-text LAS from 84.8 for CamelParser2.0 to 86.0 for CamelParser2.1. Under best-ICL, Gemini3 surpasses CamelParser2.0 on LS, UAS, and LAS, but remains below CamelParser2.1; GPT5.2 also remains competitive with the supervised parsers. Overall, these results show that ICL improves not only dependency prediction but also tokenization decisions.

Number of Shots	0		1		3		5		10	
Selection Method	All Tags	Tag F ₁	All Tags	Tag F ₁	All Tags	Tag F ₁	All Tags	Tag F ₁	All Tags	Tag F ₁
Lowest Cosine Similarity	–	–	16.5	83.5	20.1	84.9	23.3	86.1	29.0	87.8
Lowest chrF++	–	–	21.4	82.1	23.6	85.5	24.5	86.2	23.9	86.2
Random	14.1	82.8	45.0	90.8	56.7	93.1	59.4	92.7	65.9	95.0
Highest Cosine Similarity	–	–	59.2	93.5	68.4	95.3	72.0	96.0	76.0	96.8
Highest chrF++	–	–	59.5	92.6	70.6	95.7	74.1	96.4	78.0	97.2

Table 5: **Morphosyntactic Tagging** performance comparison of shot selection methods across different numbers of shots for **Qwen3** on the PATB Dev set. The $k=0$ (zero-shot) results correspond to the *Random* row, since no retrieval is performed in that setting.

Model	Zero-shot		Best-ICL	
	All Tags	Tag F ₁	All Tags	Tag F ₁
Llama4	7.7	78.9	57.4	90.1
Qwen3	13.9	83.1	76.5	97.0
GPT5.2	39.7	89.9	86.2	98.4
Gemini3	48.8	93.0	86.6	98.5
CAMEL Tools	96.3	99.6	–	–

Table 6: **Morphosyntactic Tagging** results on PATB Test. Best-ICL uses the Dev-selected configuration ($k=10$; highest chrF++ retrieval).

Model	LS	UAS	LAS
Llama4	62.1	41.8	28.4
Qwen3	62.2	42.5	28.9
GPT5.2	82.1	84.1	73.1
Gemini3	87.0	85.2	78.8
CamelParser2.0	93.6	89.7	87.5
CamelParser2.1	94.4	90.6	88.4

Table 7: **Dependency Parsing** results on CAMEL TB Dev under gold tokenization (zero-shot).

6.3 Joint PATB–CATiB Evaluation

Table 11 tests whether supplying a model’s predicted morphology improves its raw-text parsing relative to performing the two tasks independently.

The row-wise comparison shows that the two strongest LLMs, GPT5.2 and Gemini3, consistently achieve higher LAS when their predicted morphology is supplied to the parser, in both zero-shot and Best-ICL settings. Qwen3 also favors cascading under Best-ICL, whereas Llama4 favors independent parsing. Thus, predicted morphology is most useful for the stronger parsers, although its benefit is not universal.

Retrieval-based ICL provides the more consistent gain: every LLM improves from zero-shot to Best-ICL under both independent and cascaded parsing. Gemini3 is the strongest cascaded LLM under Best-ICL, while CamelParser2.1 remains the

strongest system overall. Together, these results show that retrieved demonstrations substantially strengthen parsing and that cascading morphology gives the best-performing LLMs an additional LAS improvement.

7 Analysis and Discussion

7.1 Cost Comparison: Time, Money, and Data

While Gemini3 achieves competitive performance, it incurs substantially higher costs than Camel tools and CamelParser. It is $1,000\times$ – $8,000\times$ slower⁴ and costs over \$200 in prompting on the test sets alone, whereas Camel tools are free and significantly faster (Appendix C.2). Moreover, retrieval-based ICL depends on extensive annotated data: the best tagging setup covers 51% of PATB sentences (61% of tokens), and parsing requires 67% (71% of tokens) (Appendix C.3). Overall, LLM-based gains come at significant cost in time, money, and reliance on pre-annotated data.

7.2 Morphosyntactic Tagging

On the PATB Test split, we compare per-feature accuracy between CAMEL Tools and Gemini3 under Best-ICL (Appendix C.4). Most features show comparable performance, with notable differences in form gender, form number, and case: gen (3.4%), num (3.9%), and cas (3.1%).

To better understand these gaps, we analyze representative examples. For form gender and form number, most discrepancies arise from mismatches between surface-form and functional features in Arabic (Alkuhlani and Habash, 2011). For instance, broken plurals such as *طالبة* *Talabaḥ* ‘students’ are feminine singular in form but functionally masculine plural. Gemini3 often predicts the correct functional interpretation, but is penalized under the form-feature annotation scheme. While this

⁴All open-weight and supervised experiments were run on a single node with 4 A100 GPUs.

Number of Shots	0			1			3			5			10		
Selection Method	LS	UAS	LAS	LS	UAS	LAS	LS	UAS	LAS	LS	UAS	LAS	LS	UAS	LAS
Lowest Cosine Similarity	–	–	–	63.1	43.7	30.5	64.9	45.6	33.3	65.7	46.2	34.6	66.6	47.7	36.4
Lowest chrF++	–	–	–	62.6	43.2	29.7	64.9	43.9	32.5	65.4	44.6	33.4	65.5	45.4	34.2
Random	62.2	42.5	28.9	71.8	55.7	46.3	76.1	62.6	54.7	77.4	65.5	58.0	80.0	69.0	62.2
Highest Cosine Similarity	–	–	–	78.6	66.7	59.2	82.6	73.0	66.7	84.5	75.3	69.9	85.8	77.6	72.5
Highest chrF++	–	–	–	79.8	68.8	61.5	84.2	74.5	69.0	85.4	76.5	71.4	86.9	78.5	74.0

Table 8: **Dependency Parsing** performance comparison of shot selection methods across different numbers of shots for **Qwen3** on CAMELTB Dev under gold tokenization. The $k=0$ (zero-shot) results correspond to the *Random* row, since no retrieval is performed in that setting.

Model	Zero-shot				Best-ICL			
	Tok F ₁	LS	UAS	LAS	Tok F ₁	LS	UAS	LAS
Llama4	100.0	60.8	43.0	28.3	100.0	82.9	73.7	67.4
Qwen3	100.0	60.6	40.8	26.9	100.0	85.7	79.3	73.8
GPT5.2	100.0	81.8	84.1	72.8	100.0	91.8	88.6	85.0
Gemini3	100.0	86.5	85.1	78.3	100.0	94.0	90.7	88.2
CamelParser2.0	100.0	93.1	89.8	87.5	–	–	–	–
CamelParser2.1	100.0	94.6	91.7	89.3	–	–	–	–

Table 9: **Dependency Parsing** results on CAMELTB Test (gold tokenization). Best-ICL uses the Dev-selected configuration ($k=10$; highest chrF++ retrieval).

Model	Zero-shot				Best-ICL			
	Tok F ₁	LS	UAS	LAS	Tok F ₁	LS	UAS	LAS
Llama4	49.5	42.7	37.6	23.8	87.8	78.6	71.9	66.0
Qwen3	51.2	49.1	37.4	24.1	84.9	80.6	74.1	68.8
GPT5.2	93.7	70.5	69.7	60.1	98.2	90.9	87.4	84.1
Gemini3	92.8	78.1	75.0	68.4	96.2	91.9	88.1	85.3
CamelParser2.0	98.0	91.5	87.3	84.8	–	–	–	–
CamelParser2.1	98.0	92.5	88.5	86.0	–	–	–	–

Table 10: **Dependency Parsing** results on CAMELTB Test (raw text). Best-ICL uses the Dev-selected configuration ($k=10$; highest chrF++ retrieval).

System	Setting	All Tags	Tag F ₁	Independent parsing				Cascaded parsing			
				Tok F ₁	LS	UAS	LAS	Tok F ₁	LS	UAS	LAS
Llama4	Zero-shot	7.7	78.9	70.9	44.4	41.1	<u>24.9</u>	73.4	43.8	40.2	24.4
Llama4	Best-ICL	57.4	90.1	86.2	74.2	69.1	<u>60.8</u>	85.7	73.4	66.4	58.6
Qwen3	Zero-shot	13.9	83.1	73.2	50.6	42.4	<u>27.7</u>	74.2	46.2	35.6	23.7
Qwen3	Best-ICL	76.5	97.0	86.2	79.2	73.5	67.8	88.1	80.0	73.8	<u>68.1</u>
GPT5.2	Zero-shot	39.7	89.9	90.1	75.0	59.9	52.3	93.1	77.6	64.4	<u>57.4</u>
GPT5.2	Best-ICL	86.2	98.4	98.0	90.4	82.5	78.4	98.4	92.0	83.7	<u>80.7</u>
Gemini3	Zero-shot	48.8	93.0	95.7	83.6	79.1	72.1	94.0	84.2	80.0	<u>72.7</u>
Gemini3	Best-ICL	86.6	98.5	96.0	93.7	88.1	85.2	95.4	94.1	88.4	<u>85.8</u>
CamelParser2.0	Supervised	96.3	99.6	98.7	96.7	92.1	<u>90.9</u>	98.7	96.7	92.1	<u>90.9</u>
CamelParser2.1	Supervised	96.3	99.6	98.7	97.2	92.6	<u>91.5</u>	98.7	97.2	92.6	<u>91.5</u>

Table 11: **Joint PATB–CATiB results**. LLM parsing is performed independently or with the same model’s predicted morphology. Morphology scores are shared across conditions; CamelParser scores are repeated because CamelParser already consumes the morphological analyses produced by CAMEL Tools. Bold indicates the best score in each column. Within each row, the higher LAS between independent and cascaded parsing is underlined.

reflects a limitation of the evaluation setup, it is consistent with prior work (Zalmout et al., 2018; Inoue et al., 2022). For case (cas), the main difficulty lies in the u (undefined) category, which

appears in the reference for many proper nouns and indeclinable nouns and adjectives lacking overt case marking, e.g. أخرى *Āuxray* ‘other’. Gemini3 achieves 72.6% accuracy on *cas=u*, compared to 94.4% for CAMEL Tools. Excluding *cas=u*, performance becomes nearly identical (98.6% vs. 98.4%), indicating that most of the overall difference is concentrated in this category.

These observations suggest that differences between CAMEL Tools and Gemini3 are smaller than they initially appear and largely reflect specifics of the reference annotation and task definition.

7.3 Dependency Parsing

Error Patterns On the CAMELTB Test set with gold tokenization, root attachment explains the ordering between Gemini3 and CamelParser2.0. The gold trees average 1.41 roots per input, with 545 of 1,918 inputs containing multiple roots, which CamelParser2.0’s exactly-one-root decoder cannot reproduce. Gemini3 consequently has much higher head accuracy on tokens attached to the gold root (85.5% vs. 66.1%). CamelParser2.1 keeps the same checkpoint and scoring model but uses projective multiroot decoding, raising gold-root head accuracy to 89.5% and overall LAS to 89.3% while leaving non-root LAS unchanged at 89.3%.

Differences by Genre and Length The genre breakdown in Appendix C.6 shows that much of the variation follows root structure. CamelParser2.0 outperforms Gemini3 on 6 of 13 CAMELTB genres, whereas CamelParser2.1 does so on 10; multiroot decoding improves 11 genres, with the largest gains on Quran, OT, NT, and Sara. Importantly, Gemini3 is strongest on Classical Arabic overall, leading on Odes, Quran, and Hayy, whereas CamelParser2.1 is strongest on MSA overall. By length, Gemini3 leads on short inputs, while CamelParser2.1 leads on mid-length and long inputs. Thus, multiroot decoding explains much of the cross-genre variation, but the results also reveal a broader contrast between Gemini3’s strength on Classical Arabic and the supervised parser’s strength on MSA.

7.4 Tokenization Error Analysis

To analyze tokenization on the CAMELTB Test split, we manually examined a 500-sentence sample (6,297 gold tokens) from the raw-text setting, aligning CAMEL Tools and Gemini3 Best-ICL outputs with the gold token sequence. Unlike earlier evaluations that ignored normalizable differ-

ences, this analysis accounts for all discrepancies, explaining Gemini3’s lower tokenization scores. As shown in Appendix C.1 Table 12, the systems exhibit distinct error profiles. CamelParser uses CAMEL Tools for tokenization, which is trained on PATB (MSA newswire) and have limited support for Classical Arabic and other CAMELTB genres. The CAMEL Tools errors are dominated by punctuation normalization (55%), e.g., Western vs. Arabic punctuation (“?” vs. ؟), which rarely affect token boundaries, along with moderate under-tokenization (11%) and normalization differences (15%). In contrast, Gemini3’s errors are primarily hallucinations (56%), generating tokens not present in the input (e.g., English words or unrelated tokens such as *plus*, *index*, *standard*, *village*, and *Current*, as well as occasional non-Arabic characters such as 突破 *tūpò* ‘breakthrough’). Excluding hallucinations, Gemini3 shows relatively few tokenization errors. Both systems also exhibit minor discrepancies due to gold annotation issues and normalization variation. Overall, CamelParser errors are largely superficial, while Gemini3’s lower performance is mainly driven by hallucinated tokens rather than systematic tokenization errors. This analysis highlights some of the weaknesses of the baseline models as well as the LLM models, all of which open interesting pathways for future research.

8 Conclusion and Future Work

We evaluated instruction-tuned LLMs on Arabic morphosyntactic tagging and dependency parsing, both independently and in a cascade, under zero-shot and retrieval-based ICL. Relevant demonstrations substantially improve both tasks, especially for open-weight models. Strong LLMs approach supervised systems on individual morphological features, although exact feature bundles remain difficult, and become competitive with specialized dependency parsers. The strongest LLM configuration reaches 85.8 LAS, but raw-text tokenization remains an important source of error. Overall, LLMs can produce useful explicit analyses of Arabic morphology and syntax, although competitive performance relies on substantial annotated data and computation. Future work should explore joint modeling of tokenization, morphology, and syntax and evaluate these methods on other Arabic dialects.

Limitations

Our study has three main limitations. First, the evaluation focuses on Standard Arabic and does not cover dialectal varieties, so the findings may not generalize across Arabic. Second, retrieval-based ICL is computationally and financially costly, and its long prompts may be impractical for models with smaller context windows. Finally, although our cascade connects morphology and parsing, it does not jointly model tokenization, morphology, and syntax. The independent PATB–CATiB comparison is therefore a pipeline-level ablation: morphology results are shared across conditions, which differ only in whether those predictions are supplied to the parser.

Ethics Statement

This work evaluates large language models (LLMs) on structured linguistic analysis tasks for Standard Arabic. We use publicly available Arabic treebanks and do not introduce new annotated data involving human subjects. As such, risks related to privacy, consent, or personal data are minimal.

However, LLMs may reflect biases present in their training data, including dialectal, regional, or stylistic biases in Arabic. Our evaluation focuses on Standard Arabic and may not generalize to dialectal varieties, potentially reinforcing disparities in language technology coverage. In addition, reliance on proprietary models raises concerns about reproducibility and transparency.

We emphasize that our findings are diagnostic and should not be interpreted as evidence of robust linguistic competence. Misinterpretation of model outputs in downstream applications—especially in educational or linguistic contexts—may lead to incorrect analyses. We encourage careful use of LLM-generated linguistic annotations and further research on fairness and robustness across Arabic varieties.

We used AI writing assistance within the scope of “Assistance purely with the language of the paper” described in the ACL Policy on Publication Ethics.

Acknowledgments

We thank Go Inoue and Muhammed AbuOdeh for insightful discussions and feedback. We gratefully acknowledge the support and computational resources provided by the High Performance Computing Center at New York University Abu Dhabi,

which supported the model training and experiments in this work. This research is in part with support from Google.org and the Google Cloud Research Credits program for the Gemini Academic Program.

References

- Ahmed Abdelali, Kareem Darwish, Nadir Durrani, and Hamdy Mubarak. 2016. [Farasa: A fast and furious segmenter for Arabic](#). In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 11–16, San Diego, California. Association for Computational Linguistics.
- Sharefah Al-Ghamdi, Hend Al-Khalifa, and Abdulmalik Al-Salman. 2021. [A dependency treebank for classical Arabic poetry](#). In *Proceedings of the Sixth International Conference on Dependency Linguistics (Depling, SyntaxFest 2021)*, pages 1–9, Sofia, Bulgaria. Association for Computational Linguistics.
- Sharefah Al-Ghamdi, Hend Al-Khalifa, and Abdulmalik Al-Salman. 2023. [Fine-Tuning Bert-Based Pre-Trained Models for Arabic Dependency Parsing](#). *Applied Sciences*, 13(7).
- Kamal Al-Sabahi, Amer Fael Mohammed Balhaf, Mohamed Moheb Zaghloul Al Shamy, Majjed Al-Qatf, Kang Yang, and Abdulrahman Al-Badwi. 2026. [Benchmarking Large Language Models on Arabic parsing](#). *International Journal of Information Management Data Insights*, 6(1):100404.
- Bashar Alhafni, Nizar Habash, and Houda Bouamor. 2022. [User-centric gender rewriting](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 618–631, Seattle, United States. Association for Computational Linguistics.
- Bashar Alhafni, Go Inoue, Christian Khairallah, and Nizar Habash. 2023. [Advancements in Arabic grammatical error detection and correction: An empirical investigation](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6430–6448, Singapore. Association for Computational Linguistics.
- Sarah Alkuhlani and Nizar Habash. 2011. [A corpus for modeling morpho-syntactic agreement in Arabic: Gender, number and rationality](#). In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 357–362, Portland, Oregon, USA. Association for Computational Linguistics.
- Dang Anh, Limor Raviv, and Lukas Galke. 2024. [Morphology matters: Probing the cross-linguistic morphological generalization abilities of large language models through a wug test](#). In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 177–188, Bangkok, Thailand. Association for Computational Linguistics.

- M Saiful Bari, Yazeed Alnumay, Norah A. Alzahrani, Nouf M. Alotaibi, Hisham Abdullah Alyahya, Sultan AlRashed, Faisal Abdulrahman Mirza, Shaykhah Z. Alsubaie, Hassan A. Alahmed, Ghadah Alabduljabbar, Raghad Alkhathran, Yousef Almushayqih, Raneem Alnajim, Salman Alsubaihi, Maryam Al Mansour, Saad Amin Hassan, Dr. Majed Alrubaian, Ali Alammari, Zaki Alawami, and 7 others. 2025. [AL-Lam: Large Language Models for Arabic and English](#). In *The Thirteenth International Conference on Learning Representations*.
- Terra Blevins, Hila Gonen, and Luke Zettlemoyer. 2023. [Prompting language models for linguistic structure](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6649–6663, Toronto, Canada. Association for Computational Linguistics.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. [Language models are few-shot learners](#). *Preprint*, arXiv:2005.14165.
- Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.
- Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2004. [Automatic tagging of Arabic text: From raw text to base phrase chunks](#). In *Proceedings of HLT-NAACL 2004: Short Papers*, pages 149–152, Boston, Massachusetts, USA. Association for Computational Linguistics.
- Kais Dukes and Tim Buckwalter. 2010. A Dependency Treebank of the Quran using Traditional Arabic Grammar. In *Proceedings of the Conference on Informatics and Systems (INFOS)*, Cairo, Egypt.
- Ahmed El Kholy and Nizar Habash. 2010. [Orthographic and morphological processing for English-Arabic statistical machine translation](#). In *Actes de la 17e conférence sur le Traitement Automatique des Langues Naturelles. Articles longs*, pages 282–291, Montréal, Canada. ATALA.
- Ahmed Elshabrawy, Muhammed AbuOdeh, Go Inoue, and Nizar Habash. 2023. [CamelParser2.0: A state-of-the-art dependency parser for Arabic](#). In *Proceedings of ArabicNLP 2023*, pages 170–180, Singapore (Hybrid). Association for Computational Linguistics.
- Ana Ezquerro, Carlos Gómez-Rodríguez, and David Vilares. 2025. [Better benchmarking LLMs for zero-shot dependency parsing](#). In *Proceedings of the Joint 25th Nordic Conference on Computational Linguistics and 11th Baltic Conference on Human Language Technologies (NoDaLiDa/Baltic-HLT 2025)*, pages 121–135, Tallinn, Estonia. University of Tartu Library.
- Fanar Team, Ummar Abbas, Mohammad Shahmeer Ahmad, Firoj Alam, Enes Altinisik, Ehsannedin Asgari, Yazan Boshmaf, Sabri Boughorbel, Sanjay Chawla, Shammur Chowdhury, Fahim Dalvi, Kareem Darwish, Nadir Durrani, Mohamed Elfeky, Ahmed Elmagarmid, Mohamed Eltabakh, Masoomali Fatehkia, Anastasios Fragkopoulos, Maram Hasanain, and 23 others. 2025. [Fanar: An Arabic-Centric Multimodal Generative AI Platform](#). *Preprint*, arXiv:2501.13944.
- Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, and 1332 others. 2025. [Gemini: A family of highly capable multimodal models](#). *Preprint*, arXiv:2312.11805.
- Michael Ginn, Mans Hulden, and Alexis Palmer. 2024. [Can we teach language models to gloss endangered languages?](#) In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5861–5876, Miami, Florida, USA. Association for Computational Linguistics.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Spence Green and Christopher D. Manning. 2010. [Better Arabic parsing: Baselines, evaluations, and analysis](#). In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 394–402, Beijing, China. Coling 2010 Organizing Committee.
- Nizar Habash, Muhammed AbuOdeh, Dima Taji, Reem Faraj, Jamila El Gizuli, and Omar Kallas. 2022. [Camel treebank: An open multi-genre Arabic dependency treebank](#). In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2672–2681, Marseille, France. European Language Resources Association.
- Nizar Habash and Owen Rambow. 2005. [Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop](#). In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL’05)*, pages 573–580, Ann Arbor, Michigan. Association for Computational Linguistics.
- Nizar Habash and Ryan Roth. 2009. [CATiB: The Columbia Arabic treebank](#). In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 221–224, Suntec, Singapore. Association for Computational Linguistics.
- Go Inoue, Bashar Alhafni, Nurpeiis Baimukan, Houda Bouamor, and Nizar Habash. 2021. [The interplay of variant, size, and task type in Arabic pre-trained language models](#). In *Proceedings of the Sixth Arabic*

- Natural Language Processing Workshop*, pages 92–104, Kyiv, Ukraine (Virtual). Association for Computational Linguistics.
- Go Inoue, Salam Khalifa, and Nizar Habash. 2022. [Morphosyntactic tagging with pre-trained language models for Arabic and its dialects](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1708–1719, Dublin, Ireland. Association for Computational Linguistics.
- Go Inoue, Hiroyuki Shindo, and Yuji Matsumoto. 2017. [Joint prediction of morphosyntactic categories for fine-grained Arabic part-of-speech tagging exploiting tag dictionary information](#). In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 421–431, Vancouver, Canada. Association for Computational Linguistics.
- Mete Ismayilzada, Defne Circi, Jonne Sälevä, Hale Sirin, Abdullatif Köksal, Bhuwan Dhingra, Antoine Bosselut, Duygu Ataman, and Lonneke Van Der Plas. 2025. [Evaluating morphological compositional generalization in large language models](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1270–1305, Albuquerque, New Mexico. Association for Computational Linguistics.
- Yash Kankanampati, Joseph Le Roux, Nadi Tomeh, Dima Taji, and Nizar Habash. 2020. [Multitask easy-first dependency parsing: Exploiting complementarities of different dependency representations](#). In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2497–2508, Barcelona, Spain (Online). International Committee on Computational Linguistics.
- Salam Khalifa, Nasser Zalmout, and Nizar Habash. 2020. [Morphological analysis and disambiguation for Gulf Arabic: The interplay between resources and methods](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 3895–3904, Marseille, France. European Language Resources Association.
- Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. [What makes good in-context examples for GPT-3?](#) In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online. Association for Computational Linguistics.
- Hsuehkuang Lu, Lei Hou, and Juanzi Li. 2019. How important is pos to dependency parsing? joint pos tagging and dependency parsing neural networks. In *Chinese Computational Linguistics*, pages 625–637, Cham. Springer International Publishing.
- Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki. 2004. The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus. In *Proceedings of the International Conference on Arabic Language Resources and Tools*, pages 102–109, Cairo, Egypt.
- Yuval Marton, Nizar Habash, and Owen Rambow. 2013. [Dependency parsing of Modern Standard Arabic with lexical and inflectional features](#). *Computational Linguistics*, 39(1):161–194.
- Hiroshi Matsuda, Chunpeng Ma, and Masayuki Asahara. 2025. [Step-by-step instructions and a simple tabular output format improve the dependency parsing accuracy of LLMs](#). In *Proceedings of the 18th International Conference on Parsing Technologies (IWPT, SyntaxFest 2025)*, pages 11–19, Ljubljana, Slovenia. Association for Computational Linguistics.
- Meta. 2025. [The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation](#).
- Alessio Miaschi, Felice Dell’Orletta, and Giulia Venturi. 2024. [Evaluating large language models via linguistic profiling](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2835–2848, Miami, Florida, USA. Association for Computational Linguistics.
- Sewon Min, Xinxu Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. [Rethinking the role of demonstrations: What makes in-context learning work?](#) In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Amir More, Amit Seker, Victoria Basmova, and Reut Tsarfaty. 2019. [Joint transition-based models for morpho-syntactic parsing: Parsing strategies for MRLs and a case study from Modern Hebrew](#). *Transactions of the Association for Computational Linguistics*, 7:33–48.
- Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. [Universal Dependencies v2: An evergrowing multilingual treebank collection](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.
- Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhil Eryani, Alexander Erdmann, and Nizar Habash. 2020. [CAMEL tools: An open source python toolkit for Arabic natural language processing](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France. European Language Resources Association.
- OpenAI. 2026. [OpenAI GPT-5 system card](#). *Preprint*, arXiv:2601.03267.
- OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, and 262 others. 2024. [Gpt-4 technical report](#). *Preprint*, arXiv:2303.08774.
- Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholi, Ramy Eskander, Nizar Habash,

- Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. [MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of Arabic](#). In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1094–1101, Reykjavik, Iceland. European Language Resources Association (ELRA).
- Maja Popović. 2015. [chrF: character n-gram F-score for automatic MT evaluation](#). In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.
- Maja Popović. 2017. [chrF++: words helping character n-grams](#). In *Proceedings of the Second Conference on Machine Translation*, pages 612–618, Copenhagen, Denmark. Association for Computational Linguistics.
- Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. [Learning to retrieve prompts for in-context learning](#). In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.
- Fatiha Sadat and Nizar Habash. 2006. [Combination of Arabic preprocessing schemes for statistical machine translation](#). In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 1–8, Sydney, Australia. Association for Computational Linguistics.
- Neha Sengupta, Sunil Kumar Sahu, Bokang Jia, Satheesh Katipomu, Haonan Li, Fajri Koto, William Marshall, Gurpreet Gosal, Cynthia Liu, Zhiming Chen, Osama Mohammed Afzal, Samta Kamboj, Onkar Pandit, Rahul Pal, Lalit Pradhan, Zain Muhammad Mujahid, Massa Baali, Xudong Han, Sondas Mahmoud Bsharat, and 13 others. 2023. [Jais and jais-chat: Arabic-centric foundation and instruction-tuned open generative large language models](#). *Preprint*, arXiv:2308.16149.
- Anas Shahrour, Salam Khalifa, Dima Taji, and Nizar Habash. 2016. [CamelParser: A system for Arabic syntactic analysis and morphological disambiguation](#). In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: System Demonstrations*, pages 228–232, Osaka, Japan. The COLING 2016 Organizing Committee.
- Otakar Smrž, Jan Šnaidauf, and Petr Zemánek. 2002. Prague Dependency Treebank for Arabic: Multi-Level Annotation of Arabic Corpus. In *Proceedings of the International Symposium on Processing of Arabic*, pages 147–155, Manouba, Tunisia.
- Milan Straka. 2018. [UDPipe 2.0 prototype at CoNLL 2018 UD shared task](#). In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 197–207, Brussels, Belgium. Association for Computational Linguistics.
- Dima Taji, Nizar Habash, and Daniel Zeman. 2017. [Universal Dependencies for Arabic](#). In *Proceedings of the Third Arabic Natural Language Processing Workshop*, pages 166–176, Valencia, Spain. Association for Computational Linguistics.
- Tsedeniya Kinfe Temesgen, Marion Di Marco, and Alexander Fraser. 2025. [Extracting linguistic information from large language models: Syntactic relations and derivational knowledge](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 27210–27226, Suzhou, China. Association for Computational Linguistics.
- Yuanhe Tian, Fei Xia, and Yan Song. 2024. [Large language models are no longer shallow parsers](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7131–7142, Bangkok, Thailand. Association for Computational Linguistics.
- Reut Tsarfaty, Joakim Nivre, and Evelina Andersson. 2012. [Joint evaluation of morphological segmentation and syntactic parsing](#). In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 6–10, Jeju Island, Korea. Association for Computational Linguistics.
- Andreas Waldis, Yotam Perlitz, Leshem Choshen, Yufang Hou, and Iryna Gurevych. 2024. [Holmes: A benchmark to assess the linguistic competence of language models](#). *Transactions of the Association for Computational Linguistics*, 12:1616–1647.
- Leonie Weissweiler, Valentin Hofmann, Anjali Kantharuban, Anna Cai, Ritam Dutt, Amey Hengle, Anubha Kabra, Atharva Kulkarni, Abhishek Vijayakumar, Hao-fei Yu, Hinrich Schuetze, Kemal Oflazer, and David Mortensen. 2023. [Counting the bugs in ChatGPT’s wugs: A multilingual investigation into the morphological capabilities of a large language model](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6508–6524, Singapore. Association for Computational Linguistics.
- Marion Weller-Di Marco and Alexander Fraser. 2024. [Analyzing the understanding of morphologically complex words in large language models](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 1009–1020, Torino, Italia. ELRA and ICCL.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *Preprint*, arXiv:2505.09388.
- Nasser Zalmout, Alexander Erdmann, and Nizar Habash. 2018. [Noise-robust morphological disambiguation for dialectal Arabic](#). In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*,

pages 953–964, New Orleans, Louisiana. Association for Computational Linguistics.

Nasser Zalmout and Nizar Habash. 2017. [Don’t throw those morphological analyzers away just yet: Neural morphological disambiguation for Arabic](#). In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 704–713, Copenhagen, Denmark. Association for Computational Linguistics.

Nasser Zalmout and Nizar Habash. 2019. [Adversarial multitask learning for joint multi-feature and multi-dialect morphological modeling](#). In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1775–1786, Florence, Italy. Association for Computational Linguistics.

Nasser Zalmout and Nizar Habash. 2020. [Joint diacritization, lemmatization, normalization, and fine-grained morphological tagging](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8297–8307, Online. Association for Computational Linguistics.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. [Randomized greedy inference for joint segmentation, POS tagging and dependency parsing](#). In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 42–52, Denver, Colorado. Association for Computational Linguistics.

Ziyan Zhang, Yang Hou, Chen Gong, and Zhenghua Li. 2025. [Self-correction makes LLMs better parsers](#). In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 6749–6762, Suzhou, China. Association for Computational Linguistics.

A Prompts

A.1 Morphosyntactic Tagging Prompt

The panel labels below are editorial and decompose the fixed system prompt into functional parts for readability. The prompt content and ordering are preserved.

Task, tokens, and output

You are an expert Arabic morphosyntactic analyzer trained on the Penn Arabic Treebank (PATB).

Your task is to analyze the Arabic sentences and produce a complete morphosyntactic analysis for EACH TOKEN.

TOKENIZATION

- The sentence is already tokenized and tokens are separated by spaces.
- Do NOT merge or split tokens.
- Analyze tokens strictly in the given order.

OUTPUT REQUIREMENTS

- Output MUST be valid JSON.
- Output MUST contain a single key: "tokens".
- "tokens" is a list of token objects.
- Each token object MUST include:
 - "word": the surface form exactly as given
 - "analysis": a full feature structure
- ALL features listed below MUST appear for every token.
- Use ONLY the allowed values.
- Do NOT include explanations, comments, or extra fields in the output.

Morphosyntactic feature inventory

MORPHOLOGICAL & SYNTACTIC FEATURES
(WITH ALLOWED VALUES & DESCRIPTIONS)

asp - (Aspect)

- c : Command
- i : Imperfective
- p : Perfective
- na : Not applicable

cas - (Case)

- n : Nominative
- a : Accusative
- g : Genitive
- na : Not applicable
- u : Undefined

form_gen - (Form gender)

- f : Feminine
- m : Masculine
- na : Not applicable

form_num - (Form number)

- s : Singular
- d : Dual
- p : Plural
- na : Not applicable
- u : Undefined

mod - (Mood)

- i : Indicative
- j : Jussive
- s : Subjunctive
- na : Not applicable
- u : Undefined

per - (Person)

- 1 : 1st
- 2 : 2nd
- 3 : 3rd
- na : Not applicable

stt - (State)

- c : Construct/Poss/Idafa
- d : Definite
- i : Indefinite
- na : Not applicable
- u : Undefined

vox - (Voice)

- a : Active
- p : Passive
- na : Not applicable
- u : Undefined

Part-of-speech feature inventory

pos - (Part of speech)

- noun : Noun
- noun_prop : Proper noun
- noun_num : Number noun
- noun_quant : Quantity noun
- adj : Adjective
- adj_comp : Comparitive adjective
- adj_num : Numerical adjective
- adv : Adverb
- adv_interrog : Interrogative adverb
- adv_rel : Relative adverb
- pron : Pronoun
- pron_dem : Demonstrative pronoun
- pron_exclam : Pronoun exclamation
- pron_interrog : Interrogative pronoun
- pron_rel : Relative pronoun
- verb : Verb
- verb_pseudo : Pseudo verb
- part : Particle
- part_dem : Demonstrative particle
- part_det : Determiner particle
- part_focus : Focus particle
- part_fut : Future marker particle
- part_interrog : Interrogative particle
- part_neg : Negative particle
- part_restrict : Restrictive particle
- part_verb : Verbal particle
- part_voc : Vocalized particle
- prep : Preposition
- abbrev : Abbreviation
- punc : Punctuation
- conj : Conjunction
- conj_sub : Subordinating conjunction
- interj : Interjection
- digit : Digital numbers
- latin : Latin/foreign

Clitic feature inventory

Proclitics

- prc0 - (Article proclitic)
- 0 : No proclitic
 - na : Not applicable
 - Aa_prondem : Demonstrative particle Aa
 - Al_det : Determiner
 - AlmA_neg : Determiner Al + negative particle mA
 - lA_neg : Negative particle lA
 - mA_neg : Negative particle mA
 - ma_neg : Negative particle ma
 - mA_part : Particle mA
 - mA_rel : Relative pronoun mA

- prc1 - (Preposition proclitic)
- 0 : No proclitic
 - na : Not applicable
 - <i\$_interrog : Interrogative ish
 - bi_part : Particle bi
 - bi_prep : Preposition bi
 - bi_prog : Progressive verb particle bi
 - Ea_prep : Preposition Ea
 - EalaY_prep : Preposition EalaY
 - fiy_prep : Preposition fy
 - hA_dem : Demonstrative hA
 - Ha_fut : Future marker Ha
 - ka_prep : Preposition ka
 - la_emph : Emphatic particle la
 - la_prep : Preposition la
 - la_rc : Response conditional la
 - libi_prep : Preposition li + preposition bi
 - laHa_emphfut : Emphatic la + future marker Ha
 - laHa_rcfut : Response conditional la + future marker Ha
 - li_jus : Jussive li
 - li_sub : Subjunctive li
 - li_prep : Preposition li
 - min_prep : Preposition min
 - sa_fut : Future marker sa
 - ta_prep : Preposition ta
 - wa_part : Particle wa
 - wa_prep : Preposition wa
 - wA_voc : Vocative wA
 - yA_voc : Vocative yA

- prc2 - (Conjunction proclitic)
- 0 : No proclitic
 - na : Not applicable
 - fa_conj : Conjunction fa
 - fa_conn : Connective particle fa
 - fa_rc : Responsive conditional fa
 - fa_sub : Subordinating conjunction fa
 - wa_conj : conjunction wa
 - wa_part : particle wa
 - wa_sub : Subordinating conjunction wa

- prc3 - (Question proclitic)
- 0 : No proclitic
 - na : Not applicable
 - >a_ques : Interrogative partical >a

Clitic feature inventory: enclitics

- enc0 - (Pronominal enclitic)
- 0 : No enclitic
 - na : Not applicable
 - 1s_dobj : 1st person singular direct object
 - 1s_poss : 1st person singular possessive
 - 1s_pron : 1st person singular pronoun
 - 1p_dobj : 1st person plural direct object
 - 1p_poss : 1st person plural possessive
 - 1p_pron : 1st person plural pronoun
 - 2d_dobj : 2nd person dual direct object
 - 2d_poss : 2nd person dual possessive
 - 2d_pron : 2nd person dual pronoun
 - 2p_dobj : 2nd person plural direct object
 - 2p_poss : 2nd person plural possessive
 - 2p_pron : 2nd person plural pronoun
 - 2fs_dobj : 2nd person feminine singular direct object
 - 2fs_poss : 2nd person feminine singular possessive
 - 2fs_pron : 2nd person feminine singular pronoun
 - 2fp_dobj : 2nd person feminine plural direct object
 - 2fp_poss : 2nd person feminine plural possessive
 - 2fp_pron : 2nd person feminine plural pronoun
 - 2ms_dobj : 2nd person masculine singular direct object
 - 2ms_poss : 2nd person masculine singular possessive
 - 2ms_pron : 2nd person masculine singular pronoun
 - 2mp_dobj : 2nd person masculine plural direct object
 - 2mp_poss : 2nd person masculine plural possessive
 - 2mp_pron : 2nd person masculine plural pronoun

Clitic feature inventory: enclitics (continued)

- 3d_dobj : 3rd person dual direct object
- 3d_poss : 3rd person dual possessive
- 3d_pron : 3rd person dual pronoun
- 3p_dobj : 3rd person plural direct object
- 3p_poss : 3rd person plural possessive
- 3p_pron : 3rd person plural pronoun
- 3fs_dobj : 3rd person feminine singular direct object
- 3fs_poss : 3rd person feminine singular possessive
- 3fs_pron : 3rd person feminine singular pronoun
- 3fp_dobj : 3rd person feminine plural direct object
- 3fp_poss : 3rd person feminine plural possessive
- 3fp_pron : 3rd person feminine plural pronoun
- 3ms_dobj : 3rd person masculine singular direct object
- 3ms_poss : 3rd person masculine singular possessive
- 3ms_pron : 3rd person masculine singular pronoun
- 3mp_dobj : 3rd person masculine plural direct object
- 3mp_poss : 3rd person masculine plural possessive
- 3mp_pron : 3rd person masculine plural pronoun
- Ah_voc : Vocative particle Ah
- lA_neg : Negative particle lA
- ma_interrog : Interrogative pronoun ma
- mA_interrog : Interrogative pronoun mA
- man_interrog : Interrogative pronoun man
- ma_rel : Relative pronoun ma
- mA_rel : Relative pronoun mA
- man_rel : Relative pronoun man
- ma_sub : Subordinating conjunction ma
- mA_sub : Subordinating conjunction mA

Query

Now analyze the following Arabic sentence with full PATB-style morphosyntax.
Tokens:
["TOKEN_1", "TOKEN_2", ..., "TOKEN_n"]

A.2 Dependency Parsing Prompts

The panel labels below are editorial and expose the functional structure of the prompts. The prompt content and ordering are preserved.

A.2.1 Gold-tokenization prompt

The following prompt was used when gold CATiB tokenization was provided to the model.

Task, input, and output

You are an expert Arabic dependency annotator trained on the Columbia Arabic Treebank (CATiB) conventions.
Your task is to annotate Arabic sentences using the official CATiB methodology.

Input format

You are given a single Arabic sentence already tokenized according to CATiB conventions.
The provided token sequence must be used exactly as-is for dependency annotation.

Output format

Output only a JSON object with a single key "parses".
The value of "parses" must be an array of objects, one per token, each containing:
- id: integer (1-based token index)
- form: identical to the produced token (including "+" markers)
- head: integer (1-based index of the syntactic head; 0 = root)
- deprel: dependency relation label
Each non-root token must have exactly one head and one dependency relation.
No explanations, commentary, or additional formatting should be included.

Relations and attachment rules

Dependency relations

- SBJ - Subject (verbal, nominal, or copular sentence)
- OBJ - Object of verbs, deverbal nouns, prepositions, and conjunctions
- PRD - Predicate of incomplete verbs or verb-like particles
- TPC - Topic in a complex nominal sentence with a different internal subject
- IDF - Idafa (genitive possessor)
- TMZ - Tamyiz / specification
- MOD - Modifier (adjectival, adverbial, prepositional, appositional, relative, demonstrative, etc.)
- - Flat / multiword proper-name connector

Sentence and clause structure

Verbal sentences

- Verb is the clause head.
- SBJ marks explicit subjects (pre- or post-verbal).
- OBJ marks direct or pronominal objects.
- Passive verbs take surface SBJ for the underlying object.
- Verbal particles (سوف, قد, لن, لم) attach as MOD.

Nominal sentences

- Predicate heads the sentence.
- Topic is SBJ.
- If preceded by an incomplete or verb-like particle, the particle heads both SBJ and PRD.

Complex sentences

- If topic and embedded subject are the same, use SBJ.
- Otherwise, use TPC.

Phrase-level and attachment rules

Prepositional phrases

- Preposition heads its OBJ.
- Preposition attaches to its governor as MOD.

Nominal modification

- IDF marks possessors in idafa (only one IDF per head).
- MOD marks adjectives, relative clauses, demonstratives, and apposition.
- TMZ marks specification.

Relative clauses

- Attach to head noun using MOD.
- Relative pronoun heads the clause when present.

Coordination

- First conjunct attaches to conjunction as MOD.
- Conjunction attaches to second conjunct as OBJ.
- Sentence-initial و attaches as MOD to the following clause head.

Subordination

- Main clause head attaches to subordinator as MOD.
- Subordinate clause head attaches to subordinator as OBJ.
- أَنْ / اَنْ may attach by grammatical role (SBJ, PRD, etc.).

Punctuation

- Punctuation attaches as MOD to the governing node.
- Sentence-final punctuation attaches to the main clause head.

Query

Now annotate the following Arabic sentence using CATiB.

Tokens:

["TOKEN_1", "TOKEN_2", ..., "TOKEN_n"]

A.2.2 Raw-text additions

For the raw-text setting, the same prompt shown above was used, with the following blocks prepended to the beginning of the prompt.

Raw-text tokenization additions

Input and processing order

Input format

You are given a single Arabic sentence as RAW TEXT (a normal string), not pre-tokenized.

Tokenization (YOU MUST DO THIS FIRST)

Words in CATiB are whitespace- and punctuation-separated strings.

For annotation purposes, words must be broken into tokens following CATiB conventions.

All diacritics must be removed.

Proclitic inventory

Only the following clitics are separated from the word.

Proclitics

(written as separate tokens ending with "+"):

- أ+ question particle
- و+ and
- ف+ so / then
- ل+ for / to / so that / emphatic particle
- ب+ by / with
- ك+ as / like

Enclitic inventory

Enclitics

(written as separate tokens beginning with "+"):

- pronominal object or possessive clitics such as
 - +ه him / his
 - +ها her
 - +هم them / their
 - +كما you (dual)
 - +كم you (plural)
 - +نا us / our
 - +ي me / my

Surface-form normalization

Decliticization may result in malformed surface forms.

In such cases, normalize tokens to their naturally uncliticized form.

Alef-Lam

- الكتاب+ل → للكتاب

(NOT: لكتاب+ل)

Ta Marbuta

- نا+ مكتبة → مكتبتنا

(NOT: مكتبتنا+نا)

Alef Maqsura

- هم+ مستشفى → مستشفىهم

(NOT: مستشفىهم+هم)

Case-variant Hamza

- بهايه / بهاوه / بهاوه

→ ه+ بهاء

(NOT: ه+ بهاو /

ه+ بهاي)

Use "+" only to mark split clitics.

The produced token list must be used exactly as-is for dependency annotation.

Query

Tokenize and dependency-parse this raw sentence using CATiB.

RAW_TEXT:
<RAW TEXT>

A.3 Cascaded Morphology-Guided Parsing Prompt

Composition note. The cascaded prompt combines blocks already shown in the preceding two appendices. The CATiB task, raw-text tokenization and normalization rules, output schema, dependency labels, and attachment rules are reused from Appendix A.2. The names, meanings, and allowed values of the 14 morphological fields are reused from Appendix A.1. The rendered prompt included those shared definitions in full; they are cross-referenced here to avoid duplicating them.

Auxiliary morphology block

You are also given a predicted word-level morphosyntactic analysis for each whitespace token in the raw sentence. The predictions are auxiliary evidence and may contain errors; the RAW TEXT is authoritative.

Each morphology line contains the word index, surface word, and only its informative feature=value pairs. Values meaning no information (0, na, u, empty, or null) are omitted.

Inserted at this point: the 14 morphological field definitions and allowed values listed in Appendix A.1.

Use morphology to help decide clitic splitting, token categories, and syntactic attachments, but do not copy an analysis when it conflicts with the raw form or CATiB rules.

Query

Now annotate the following Arabic sentence using CATiB.

Sentence:
<RAW TEXT>

Predicted word-level morphology
(index, word, informative features):
<INDEX> <WORD> <FEATURE=VALUE>|...

B Licenses

We list below the licenses of the data and tools used in this work, all of which are employed in accordance with their intended use.

- **Penn Arabic Treebank (PATB)** (Maamouri et al., 2004): LDC User Agreement. We use Parts 1, 2, and 3 (LDC2010T13, LDC2011T09, and LDC2010T08, respectively).⁵ Because our morphological and syntactic data is converted from licensed PATB data, we distribute the converted datasets in encrypted form generated using the Muddler tool developed by CAMEL Lab.⁶ Reconstruction requires the corresponding licensed PATB source files.
- **CAMEL Treebank (CAMELTB)** (Habash et al., 2022): an open-source CAMEL Lab resource, accessed through its official distribution page.⁷
- **Llama 4 Scout:** Llama 4 Community License Agreement.⁸
- **Qwen3-Next-80B-A3B-Thinking:** Apache License 2.0.⁹
- **Gemini 3 Flash** and **GPT-5.2:** accessed through provider APIs under the applicable Google Cloud and OpenAI service terms.¹⁰
- **CAMEL Tools** (Obeid et al., 2020) and **CamelParser2.0** (Elshabrawy et al., 2023): MIT License.¹¹

⁵<https://catalog.ldc.upenn.edu/LDC2010T13>;
<https://catalog.ldc.upenn.edu/LDC2011T09>;
<https://catalog.ldc.upenn.edu/LDC2010T08>

⁶<https://github.com/CAMEL-Lab/muddler>

⁷<https://sites.google.com/nyu.edu/camel-treebank/resources>

⁸<https://huggingface.co/meta-llama/Llama-4-Scout-17B-16E-Instruct/blob/main/LICENSE>

⁹<https://huggingface.co/Qwen/Qwen3-Next-80B-A3B-Thinking/blob/main/LICENSE>

¹⁰<https://cloud.google.com/terms/service-terms>;
<https://openai.com/policies/services-agreement/>

¹¹https://github.com/CAMEL-Lab/camel_tools/blob/master/LICENSE;
https://github.com/CAMEL-Lab/camel_parser/blob/main/LICENSE

C Analysis

C.1 Tokenization Error Analysis

Error Type	CAMEL Tools		Gemini3		Example
	Count	%	Count	%	
Input Error	35	9%	23	10%	الصحة <i>AlSHbħ</i> → الصحة <i>AlSHħ</i>
Gold Issues	8	2%	13	6%	شعر <i>šEry</i> → شعر <i>šEr</i> +y
Under Tokenization	40	11%	8	4%	مدينة <i>mdynħ</i> +y → مدينتي <i>mdynty</i>
Over Tokenization	26	7%	7	3%	فيران <i>fyrAn</i> → ف + يران <i>fyrAn</i>
Substitution	3	1%	4	2%	والدا <i>wAlda</i> +iy → والد <i>wAld</i> +y
Normalization	56	15%	35	15%	أسف <i>Āsf</i> → أسف <i>Āsf</i>
Punctuation	204	55%	10	4%	? → ?
Hallucination	0	0%	128	56%	مركبة <i>(b+_markaba)</i> → مركبة <i>突破</i> <i>b+_b</i> + <i>突破</i> <i>mrkbħ</i>
Total	372		228		

Table 12: Distribution of tokenization errors for CAMEL Tools (as used in CamelParser) and Gemini3 Best-ICL on a 500-sentence sample from the CAMEL TB Test split in the raw-text setting. Arabic examples are shown with Buckwalter-style transliteration.

C.2 Time & Cost Analysis

Task	Model	0		10	
		Time (min)	Cost (\$)	Time (min)	Cost (\$)
Morphosyntactic Tagging	Llama4	2130.69	0.0	2084.87	0.0
	Qwen3	1746.42	0.0	1294.90	0.0
	GPT5.2	11.55	60.0	166.06	146.0
	Gemini3	1595.76	64.0	1608.69	108.3
	CAMEL Tools	1.17	0.0	–	–
Dependency Parsing (Gold Tokenization)	Llama4	246.51	0.0	274.44	0.0
	Qwen3	192.98	0.0	209.56	0.0
	GPT5.2	22.53	22.0	23.41	33.0
	Gemini3	1612.59	55.0	1070.54	41.0
	CamelParser2.0	0.13	0.0	–	–
Dependency Parsing (Raw Text)	Llama4	250.92	0.0	263.41	0.0
	Qwen3	137.83	0.0	406.66	0.0
	GPT5.2	32.64	34.0	33.93	41.0
	Gemini3	2491.41	85.0	1668.20	61.8
	CamelParser2.0	1.66	0.0	–	–

Table 13: Cost and runtime comparison across tasks and models on Test sets under **0-shot** and **10-shot** settings. Supervised pipelines (CAMEL Tools and the original CamelParser2.0 pipeline) do not use in-context learning, so only a single runtime is reported.

C.3 Data Per Shot

Dataset		Shots									
		1	2	3	4	5	6	7	8	9	10
PATB Dev	Sents	1,248	2,103	2,947	3,732	4,474	5,172	5,867	6,625	7,342	7,768
PATB Dev	Tokens	47,932	80,514	113,086	143,733	172,664	200,312	227,903	256,349	282,103	296,309
PATB Test	Sents	1,224	2,071	2,902	3,687	4,423	5,116	5,808	6,558	7,257	7,650
PATB Test	Tokens	46,801	78,853	110,776	140,919	169,451	196,693	223,925	252,001	277,566	290,559
CAMELTB Dev	Sents	1,083	1,876	2,598	3,246	3,859	4,440	4,995	5,536	6,060	6,315
CAMELTB Dev	Tokens	14,982	25,781	35,567	44,450	52,768	60,615	68,052	75,265	82,082	85,292
CAMELTB Test	Sents	1,047	1,821	2,525	3,155	3,750	4,313	4,850	5,371	5,878	6,120
CAMELTB Test	Tokens	14,402	24,916	34,473	43,134	51,258	58,904	66,141	73,155	79,791	82,930

Table 14: Number of unique sentences and tokens required as the number of shots increases.

C.4 Per Feature Accuracy

Model	Shots	All Tags	Tag F ₁	POS	PER	GEN	NUM	ASP	MOD	VOX	STT	CAS	PRC0	PRC1	PRC2	PRC3	ENC0
Llama4	0	7.7	78.9	71.2	92.5	77.2	79.8	95.6	92.1	95.1	64.6	49.4	62.2	75.8	80.2	86.3	83.0
Llama4	10	57.4	90.1	82.7	96.2	85.9	88.3	96.4	96.2	96.4	80.7	72.4	87.3	94.1	93.8	96.5	94.0
Qwen3	0	13.9	83.1	81.4	98.2	85.0	84.1	97.6	96.9	97.2	69.4	53.6	68.4	77.5	81.5	88.0	85.3
Qwen3	10	76.5	97.0	95.0	99.4	95.4	96.0	99.4	99.1	99.0	92.9	87.0	97.7	99.3	99.4	99.9	98.7
GPT5.2	0	39.7	89.9	91.1	99.2	89.4	90.0	99.7	94.3	99.5	84.1	81.9	73.3	87.6	89.4	89.7	89.2
GPT5.2	10	86.2	98.4	96.7	99.8	96.5	96.5	99.8	99.6	99.6	97.2	93.2	99.1	99.6	99.7	100.0	99.7
Gemini3	0	48.8	93.0	92.5	99.0	91.6	92.4	99.8	94.3	99.7	91.3	86.0	79.6	93.3	94.5	94.7	93.9
Gemini3	10	86.6	98.5	96.9	99.8	96.4	95.9	99.8	99.7	99.7	97.7	94.6	99.4	99.6	99.7	99.9	99.7
CAMEL Tools	–	96.3	99.6	98.8	99.8	99.8	99.8	99.8	99.8	99.8	99.1	97.7	99.8	99.9	99.9	100.0	99.9

Table 15: Per-feature morphosyntactic tagging accuracy on the PATB Test split across models and prompting settings. The 10-shot rows use the Dev-selected highest-chrF++ retrieval configuration. GEN and NUM denote the form gender and form number features, respectively (Alkuhlani and Habash, 2011).

C.5 Universal Dependencies Evaluation

Model	Setting	UPOS	All Tags	Tag F ₁	LS	UAS	LAS
Llama4	Zero-shot	87.3	43.2	77.7	49.7	45.3	28.7
Llama4	10-shot	91.4	65.2	86.7	68.8	63.7	52.1
Qwen3	Zero-shot	88.4	46.7	78.0	51.5	46.6	29.5
Qwen3	10-shot	92.7	69.2	88.6	70.9	68.5	56.1
GPT5.2	Zero-shot	91.5	56.4	84.5	61.3	74.4	53.2
GPT5.2	10-shot	94.2	74.6	91.4	73.4	79.6	65.0
Gemini3	Zero-shot	92.6	66.1	88.6	63.5	75.3	54.7
Gemini3	10-shot	94.8	77.7	93.0	73.5	79.6	64.2

Table 16: NUDAR/UD results on the Test split with gold tokenization. To examine whether the observed ICL trends extend beyond CATiB, we conduct an evaluation on the Arabic Universal Dependencies (NUDAR) Test split, a Universal Dependencies representation derived from PATB and CATiB annotations (Taji et al., 2017). Given gold-tokenized input, each LLM jointly predicts UPOS, 12 UD morphological features, dependency heads, and dependency relations for 1,963 sentences (74,125 syntactic tokens). All Tags requires exact agreement on UPOS and all 12 morphological features. We compare zero-shot prompting with 10-shot prompting, with the latter using the ten highest-chrF++ examples from the NUDAR Train split.

C.6 LAS Breakdown by Genre and Data Characteristics

Genre	Variant	Period	Length	CamelParser2.0	CamelParser2.1	Gemini3
Odes	CA	6th–12th	Short	77.3	77.6	83.3
Quran	CA	6th–12th	Long	82.7	90.1	92.2
Hadith	CA	6th–12th	Mid	92.4	92.7	90.6
1001	CA	6th–12th	Mid	91.9	92.1	91.7
Hayy	CA	6th–12th	Long	91.7	91.9	92.5
OT	MSA	19th–20th	Long	87.4	91.7	88.5
NT	MSA	19th–20th	Long	84.7	90.1	89.1
Sara	MSA	19th–20th	Long	84.5	87.8	83.7
ALC	MSA	21st	Mid	87.5	87.7	84.7
BTEC	MSA	21st	Short	84.9	86.9	86.8
QALB	MSA	21st	Mid	86.5	86.4	84.7
WikiNews	MSA	21st	Long	90.1	90.1	88.3
ZAEBUC	MSA	21st	Mid	91.8	92.3	92.0
Macro Averages						
All Genres				87.2	89.0	88.3
CA				87.2	88.9	90.1
MSA				87.2	89.1	87.2
6th–12th				87.2	88.9	90.1
19th–20th				85.5	89.9	87.1
21st				88.2	88.7	87.3
Short (<10)				81.1	82.3	85.1
Mid (10–15)				90.0	90.2	88.7
Long (>15)				86.8	90.3	89.1

Table 17: Labeled Attachment Score (LAS) by CAMELTB genre for CamelParser2.0, CamelParser2.1, and Gemini3 under its best-performing ICL setting. The table also reports unweighted macro averages by variety, period, and input-length group, with the length-group thresholds shown in the final rows.