CamelParser2.0: A State-of-the-Art Dependency Parser for Arabic

Ahmed Elshabrawy,[†‡] **Muhammed AbuOdeh,**[†] **Go Inoue,**[†‡] **Nizar Habash**[†]
[†]Computational Approaches to Modeling Language (CAMeL) Lab,
New York University Abu Dhabi (NYUAD)
[‡]Mohamed bin Zayed University of Artificial Intelligence (MBZUAI)
{ahmed.elshabrawy,go.inoue}@mbzuai.ac.ae, {m.abuodeh,nizar.habash}@nyu.edu

Abstract

We present **CamelParser2.0**, an open-source Python-based Arabic dependency parser targeting two popular Arabic dependency formalisms, the Columbia Arabic Treebank (CATiB), and Universal Dependencies (UD). The **CamelParser2.0** pipeline handles the processing of raw text and produces tokenization, part-of-speech and rich morphological features. As part of developing **CamelParser2.0**, we explore many system design hyper-parameters, such as parsing model architecture and pretrained language model selection, achieving new state-of-the-art performance across diverse Arabic genres under gold and predicted tokenization settings.

1 Introduction

Dependency parsing is a natural language processing (NLP) task used to analyze the grammatical structure of a sentence by identifying and representing the relationships between its words. Dependency parsing assigns a directed tree structure to the sentence, with words as nodes and syntactic dependencies as edges (see Figure 1). Dependency parsing, and syntactic parsing in general, has long been considered an important NLP enabling technology and analysis tool (Jurafsky and Martin, 2009). The interest in using syntactic structures in NLP in the neural age remains, e.g., as analytical tools for studying large language models (Kulmizev, 2023), for guided data augmentation for Neural Machine Translation (Duan et al., 2023), Semantic Role Labeling (Tian et al., 2022), and Grammatical Error Correction (Li et al., 2022; Zhang et al., 2022).

There have been previous developments in Arabic dependency parsing (Habash and Roth, 2009; Marton et al., 2013; Zhang et al., 2015; Shahrour et al., 2016; Al-Ghamdi et al., 2023). However, they are not based on state-of-the-art (SOTA) developments in neural dependency parsing and pre-

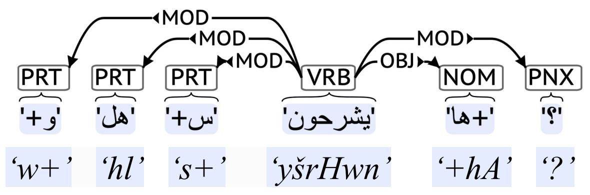

Figure 1: An example CATiB dependency tree (Habash et al., 2009) for the short question وهل سيشرحونها؟ *whl syšrHwnhA?*[2] 'and will they explain it?'.

trained language models, nor can they be easily integrated into larger project pipelines. Furthermore, they are not trained on larger and more diverse treebanks that have been developed recently. Many have only been tested with gold tokenization, not as part of a full pipeline from sentence to tree – a notable exception is the work of Zhang et al. (2015) who modeled segmentation and parsing jointly.

In this work, we investigate the effect of many system design hyper-parameters including parsing model architecture, pretrained language model selection, and training data configurations to achieve unprecedented dependency parsing performance on multiple Arabic genres. Hence, we present **CamelParser2.0**, an open-source dependency parsing pipeline that achieves SOTA performance on Columbia Arabic Treebank (CATiB) and Universal Dependencies (UD) parsing of Arabic across multiple genres from Modern Standard Arabic (MSA) and Classical Arabic (CA).

Our contributions are: (1) achieving new **state-of-the-art** on both CATiB and UD formalisms in multiple Arabic genres on all metrics; (2) developing and releasing an **open-source Python-based** pipeline for Arabic parsing;[1] and (3) **benchmarking** a large number of hyper-parameters to ensure the best system design choices.

[1]https://github.com/CAMeL-Lab/camel_parser
[2]HSB Arabic transliteration (Habash et al., 2007)

2 Related Work

2.1 Dependency Parsing

There are two main approaches to dependency parsing: transition-based (Yamada and Matsumoto, 2003; Nivre et al., 2006) and graph-based (McDonald et al., 2005). Both approaches have recently been implemented with neural models to improve performance. For example, Dozat and Manning (2016) develop a graph-based parser that uses a biaffine attention mechanism on a neural model to achieve SOTA/near SOTA results on six different languages including Czech, a morphologically rich language with flexible word order. On the other hand, Mohammadshahi and Henderson (2019) develop a transformer mechanism that conditions on graphs to be used with a neural transition-based parser to achieve SOTA results on 13 languages. The evaluations that guide the development of these architectures are mainly carried out on higher resource languages, such as English and other European languages.

In this work, we investigate how neural dependency parsing performs on Arabic given its relatively fewer resources, especially in certain classical genres, such as pre-Islamic texts.

2.2 Arabic Treebanks

The primary treebank for Arabic syntactic analysis is the Penn Arabic Treebank (PATB) (Maamouri et al., 2004), which uses a phrase structure grammar. It has been converted to a dependency representation that uses two different formalisms: CATiB (henceforth, PATB-CATiB) (Habash and Roth, 2009), and UD (NUDAR Treebank) (Taji et al., 2017). The two formalisms are compared in some detail by Taji et al. (2017).

The first dependency treebank developed for Arabic is the Prague Arabic Dependency Treebank (PADT) (Smrž et al., 2002). PADT is in part based on PATB; and it was later extended to UD (henceforth, PADT-UD).[3] Since then, several treebanks have been developed such as the Columbia Arabic Treebank (CATiB) (Habash and Roth, 2009), Quran Corpus (Dukes and Buckwalter, 2010), *i3rab* treebank (Halabi et al., 2021), and Arabic Poetry Treebank (ArPoT) (Al-Ghamdi et al., 2021). Most recently, Habash et al. (2022) released the Camel Treebank (CamelTB), which is a multi-genre Arabic dependency treebank in the CATiB formalism spanning CA texts from the 6th Century to MSA texts from the 21st century. PADT (Smrž et al., 2002), CATiB (Habash et al., 2009), and UD (Nivre et al., 2017) are dependency tree representations with different POS tags, dependency relation labels, and attachment rules.

In this work, we make use of recent developments in Arabic treebanking to explore the performance of different parsing model architectures, with different training dataset configurations, and with different dependency formalisms (CATiB and UD) on multiple Arabic genres, and under gold and predicted tokenization conditions.

2.3 Arabic Parsing

Regarding evaluating parser design in Arabic dependency parsing, the work done by Marton et al. (2013) examines the impact of morphological features on dependency parsing performance under both gold and predicted conditions. They observe differences in feature importance when using predicted features due to changes in prediction accuracy for each examined feature. They find that definiteness, person, number, gender, and undiacritized lemma are most helpful under predicted conditions. Their results are observed using MaltParser, a transition-based model, with a feature-based SVM classifier (Nivre et al., 2006), which differs from the recent neural SOTA models that learn features from the training data implicitly.

Kankanampati et al. (2020) leverage the Easy-First LSTM-based architecture proposed by Kiperwasser and Goldberg (2016), but experiment with sharing tree representations and BiLSTM layers between CATiB and UD formalisms to achieve significant error reduction on both.

More recently, Al-Ghamdi et al. (2023) employ an approach that treats dependency parsing as a sequence labeling task (Strzyz et al., 2019). They apply various pretrained BERT models under different fine-tuning and architectural setups. They explore the performance of this approach on (a) PADT (Smrž et al., 2002), (b) part 2 of PATB in the CATiB formalism (Maamouri et al., 2004; Habash and Roth, 2009), and (c) ArPoT (Al-Ghamdi et al., 2021).

To our knowledge, the current state-of-the-art in terms of *publicly available* dependency parsing systems in Arabic is the **CamelParser1.0** for the CATiB formalism (Shahrour et al., 2016), and **UDPipe 2** for the UD formalism (Straka, 2018).

[3]https://github.com/UniversalDependencies/UD_Arabic-PADT/

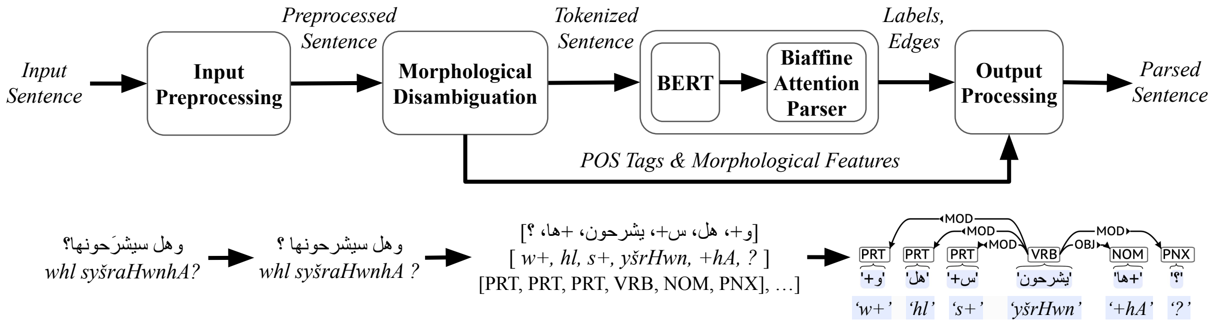

Figure 2: A diagram of the **CamelParser2.0** pipeline paired with a simple example of raw text input.

Our work is closest in high-level design to **CamelParser1.0**, which uses the MADAMIRA morphological disambiguation system based on SVM classifiers and morphological analyzers (Pasha et al., 2014) and an SVM-based parsing system called MaltParser (Nivre et al., 2006). It reported results on the Penn Arabic Treebank (Maamouri et al., 2004), which is limited to the newswire genre. Since these results were reported, significant advancements have been made in both dependency parsing and morphological analysis through the use of pretrained language models like BERT and neural model architectures (Dozat and Manning, 2016; Inoue et al., 2022); and more datasets in Arabic genres beyond newswire have been created (Habash et al., 2022). We use CAMeL Tools (Obeid et al., 2020) as part of the implementation of **CamelParser2.0**.

By utilizing the aforementioned developments in Arabic treebanking and neural dependency parsing, we experiment on PATB-CATiB, CamelTB, NUDAR, and PADT to improve the dependency parsing performance in Arabic in multiple MSA and CA genres and across the CATiB and UD formalisms. Due to a different experimental setup and data scope explored by Al-Ghamdi et al. (2023), we cannot directly compare our results on all metrics and datasets; however, we observe that our approach outperforms their reported results on the test set of PADT. Additionally, by comparing our findings to the existing SOTA pipelines, **CamelParser1.0** and **UDPipe 2**, as well as the reported results in Kankanampati et al. (2020), we observe that **CamelParser2.0** sets the new SOTA in Arabic dependency parsing for both gold and predicted tokenization settings.

3 The CamelParser2.0 Pipeline

In this section, we present the details of the **CamelParser2.0** pipeline (Figure 2). The pipeline accommodates varying levels of pre-processing in the input. Depending on the extent to which the input has been pre-processed, the pipeline conducts morphological disambiguation. Once input tokens have been identified, they are passed to the dependency parsing system which outputs dependency arcs and labels. The dependency relations are then combined with the form and additional part-of-speech tags and morphological features, which are either specified in the input or generated in the morphological disambiguation step, to output a CoNLL-X/CoNLL-U file format (Buchholz and Marsi, 2006; De Marneffe et al., 2014).

3.1 Input Formats

Before parsing begins, the input to the pipeline is directed to the proper step based on its format. Currently, we support the following input formats.

Raw Text Raw Arabic text is first cleaned by normalizing Unicode characters, removing diacritics and other characters that are not Arabic, ASCII, or Latin-1, and performing whitespace tokenization (Obeid et al., 2020). The text is then passed to the Morphological Disambiguation step (Figure 2).

Pre-Tokenized and Tagged Text Files containing token and *optional* Part-of-Speech (POS) tag tuples are supported. The input is passed to the parser directly. Since the parser does not require POS tags, they will not be produced if only tokenized text is provided.

CoNLL-X/CoNLL-U The pipeline also accepts input in the CoNLL-X/CoNLL-U tab-separated file format (Buchholz and Marsi, 2006; De Marneffe et al., 2014).

3.2 Tokenization and POS tagging

When the input is already tokenized, we pass that information onto the dependency parsing system. As for raw untokenized text, we make use of a **Morphological Disambiguation** system which predicts the tokens and the POS tags of these tokens (see Figure 2). The user determines whether to use a more accurate but more resource-intensive BERT unfactored disambiguator (Inoue et al., 2022) or a lighter Maximum Likelihood Estimation (MLE) disambiguator, both of which are included in CAMeL Tools (Obeid et al., 2020). We then extract the tokens, lemmas, and primary POS tags (CATiB or UD), as well as a set of morphological features provided by CAMeL Tools: MADA POS, position-marked proclitics and enclitics (`prc3`, `prc2`, `prc1`, `prc0`, `enc0`), person, gender, number, aspect, voice, mood, state, case, and rationality. We add a feature `token_type` to signify if the token is a baseword or clitic (indicated by its location, e.g., `prc2`).

3.3 Dependency Parsing

The next component of our parsing pipeline is the dependency parsing model, which expects tokenized Arabic data as input. We use the SuPar Biaffine Dependency Parser (Zhang, 2021), which is based on the work of Dozat and Manning (2016) with a key difference. Instead of using a GLoVe vector-based encoding layer, we generate word embeddings using a BERT model. To achieve this, a BERT model is used to generate WordPiece-level embeddings by summing up the last four layers of the BERT model (Devlin et al., 2018). Then, to generate the token-level embeddings, the corresponding WordPieces' embeddings of each token are pooled using a mean.

The output of this step is the dependency relations and labels of the input text. The POS and morphological features are integrated in the final dependency representation in an output postprocessing step (see Figure 2).

In this paper, for comparison purposes, we also report on using the MaltParser system introduced by Nivre et al. (2006) which is employed by the previous SOTA parsing system for Arabic, **CamelParser1.0** (Shahrour et al., 2016).

4 Experimental Setup

Our experimental setup involves training multiple dependency parsing models with different training data configurations which are then evaluated on multi-genre development and test sets under both gold and predicted tokenization settings to gauge accuracy and robustness across multiple genres in Arabic. The details of the various experimental setups are outlined below.

4.1 Data

The data we use to train and evaluate includes PATB-CATiB and CamelTB (CATiB representation), and PADT-UD and NUDAR (UD representation). Table 1 lists the corpora and their sub-corpora and indicates their genres, variety (MSA or CA), and sizes. We note that PADT-UD text data contains a subset of PATB. CamelTB has a variety of different sub-corpora across multiple genres, some of which are similar to PATB (WikiNews and QALB). The PATB (PATB-CATiB and NUDAR) was split according to the recommendations by Diab et al. (2013). We follow the recommendations of the creators of PADT for its data splits.[4] We split the CamelTB data according to the recommendations by Habash et al. (2022) in CamelTB v1.1.[5]

In our experiments, we examine a number of training data combinations to provide the best robustness and accuracy across multiple Arabic genres. We do not train on individual CamelTB genres because of the limited amount of data we have; but we report results for them. Similar to Kankanampati et al. (2020), we exclude all non-projective trees in the training, but not in the dev and test.

4.2 Metrics

Dependency Parsing Accuracy Evaluation of dependency parsing models is done primarily through three metrics:

- **Labeled Attachment Score (LAS)**: The percentage of tokens with correct head/parent and correct label/relation to that parent.
- **Unlabeled Attachment Score (UAS)**: The percentage of tokens with correct head/parent.
- **Label Score (LS)**: The percentage of tokens with correct label/relation.

LAS is the primary metric we report on.

[4] `https://github.com/UniversalDependencies/UD_Arabic-PADT/`

[5] `http://treebank.camel-lab.com/`

Rep	Corpus		Text Source	Var	Cent	Genre	Sents	Words	Tokens
CATiB	**PATB-CATiB**		Penn Arabic Treebank (Parts 1-2-3)	MSA	21st	News	**19,738**	**628,598**	**738,889**
CATiB	CamelTB	Odes	Suspended Odes (Mu'allaqat)	CA	6th	Poetry	784	7,465	10,170
CATiB	CamelTB	Quran	Quranic Surahs	CA	7th	Quranic	572	11,699	15,791
CATiB	CamelTB	Hadith	Hadiths from Sahih Bukhari	CA	7th	Prophetic Sayings	1,190	12,467	15,745
CATiB	CamelTB	1001	One Thousand and One Arabian Nights	CA	12th	Stories	1,145	11,831	17,109
CATiB	CamelTB	Hayy	Hayy ibn Yaqdhan (Ibn Tufail)	CA	12th	Philosophical Novel	1,198	19,674	26,583
CATiB	CamelTB	OT	Old Testament	MSA	19th	Bible Translation	535	9,097	11,788
CATiB	CamelTB	NT	New Testament	MSA	19th	Bible Translation	573	9,593	12,293
CATiB	CamelTB	Sara	Sara (Al-Akkad)	MSA	20th	Novel	1,585	35,356	46,375
CATiB	CamelTB	ALC	Arabic Learner Corpus	MSA	21st	Student Essays (L2)	727	9,221	12,047
CATiB	CamelTB	BTEC	Basic Traveling Expressions Corpus	MSA	21st	Phrasebook	2,000	15,935	18,602
CATiB	CamelTB	QALB	QALB Corpus	MSA	21st	Online Commentary	923	11,454	14,139
CATiB	CamelTB	WikiNews	WikiNews	MSA	21st	News	996	18,314	21,481
CATiB	CamelTB	ZAEBUC	Zayed Bilingual Undergraduate Corpus	MSA	21st	Student Essays (L1)	1,109	15,778	19,787
				CamelTB Total			**13,337**	**187,884**	**241,910**
				PATB-CATiB+CamelTB Total			**33,075**	**816,482**	**980,799**
UD	**PADT-UD**		Prague Arabic Dependency Treebank	MSA	21st	News	**7,664**	**17,357**	**113,500**
UD	**NUDAR**		NYUAD UD Arabic Treebank	MSA	21st	News	**19,738**	**628,598**	**738,889**

Table 1: The various datasets we experiment with in developing **CamelParser2.0**. **Rep** (Representation) specifies the treebank formalism. **Var** is the Arabic variant. **Cent** is the century. **Sents** is the number of sentences.

Statistical Significance In certain cases, we test for statistical significance using a one-tailed Welch's t-test following the recommendations of Dror et al. (2018). We treat each sentence as an independent experiment and calculate a sentence-level accuracy of parsing which we use to conduct the statistical significance testing.

4.3 Tokenization

Previous work on dependency parsing tends to judge performance purely on gold tokenization (Marton et al., 2013; Shahrour et al., 2016; Dozat and Manning, 2016; Mohammadshahi and Henderson, 2019), although there are many recent exceptions (Shao et al., 2018; More et al., 2019; Habash et al., 2022). We report on both gold and predicted tokenization to study the performance under real-world conditions. We use the BERT unfactored disambiguator (Inoue et al., 2022) in CAMeL Tools (Obeid et al., 2020). On our dev datasets (PATB and CamelTB sub-corpora), the average predicted word-level tokenization accuracy is 96.8%, with a wide range from WikiNews (99.8%) to Odes (91.3%), with PATB at 99.1%. This range of performance is consistent with our expectations since the CAMeL Tools MSA disambiguator is trained on PATB train data (news genre).

4.4 Parsing Models

We compare our **CamelParser2.0** neural dependency parsing architecture, as described in section 3.3 with other pre-existing parsing system baselines. The first baseline, **MaltParser** (v1.9.2) (Nivre et al., 2007), forms the core of the previous SOTA for dependency parsing in Arabic, **CamelParser1.0** (Shahrour et al., 2016). We compare to it directly and as part of **CamelParser1.0** (second baseline). The third baseline is UDPipe 2, whose models are currently available from the LINDAT UDPipe REST Service.[6] The last baseline is the system of Kankanampati et al. (2020); we report their published numbers where appropriate.

It is important to note that the experimentation Kankanampati et al. (2020) report on is mainly to leverage parallel data in two formalisms (CATiB and UD) and not necessarily to achieve an overall SOTA parser for Arabic. Nevertheless, they achieve impressive results so we compare against their best reported numbers. We do not leverage their multitask learning approach for **CamelParser2.0**; however, it could prove useful for future work to explore combining our approaches by sharing representations in the Biaffine parsing ar-

[6]`https://ufal.mff.cuni.cz/udpipe/2`

	LAS	UAS	LS
MaltParser	80.7	83.0	93.4
CamelParser1.0 (Shahrour et al., 2016)	83.8	86.4	93.2
Kankanampati et al. (2020)	86.2	88.1	-
CamelParser2.0	**91.3**	**92.4**	**97.0**

Table 2: Scores of various dependency parsing systems trained on the PATB-CATiB and evaluated on the test set of PATB-CATiB. **CamelParser2.0** achieves the SOTA on all metrics and improves on **CamelParser1.0** (Shahrour et al., 2016) by almost 7.5 points on the LAS. Kankanampati et al. (2020) do not report on the LS.

chitecture proposed by Dozat and Manning (2016) between different formalisms to further improve parsing performance across formalisms.

4.5 BERT Model Selection

We also experiment with four pretrained BERT models. The first three are from CamelBERT (Inoue et al., 2021): **CamelBERT-MSA** is pretrained on MSA data, **CamelBERT-CA** is pretrained on CA data, and **CamelBERT-MIX** is pretrained on MSA, CA, and Dialectal Arabic data. We make use of them because they give us an understanding of how pretrained data interplays with parsing performance on differing genres and variants. Additionally, they were created under the same settings, hence, they reduce experimental variation. Furthermore, we make use of **AraBERT v2.0** (Antoun et al., 2020) as it improves upon AraBERTv0.2 which has been shown previously to achieve SOTA performance on a range of Arabic NLP tasks (Inoue et al., 2021).

5 Results and Analysis

We present the results of the experiments we conducted as part of developing **CamelParser2.0**.

5.1 Comparing System Baselines

In Table 2, we report **CamelParser2.0**'s performance against previous SOTA baselines under the same exact training/testing conditions with gold tokenization. All systems are trained on PATB-CATiB training data and evaluated on PATB-CATiB test. It should be noted that **MaltParser** and **CamelParser1.0** use the same base algorithms and implementations; however, **CamelParser1.0** does further hyper-parameter optimization and feature selection to improve performance on Arabic as opposed to **MaltParser** which just uses the default configuration. We also include the best results reported by Kankanampati et al. (2020), however, we cannot compare our results on the LS as they do not report them. For **CamelParser2.0**, we use our baseline BERT model (CamelBERT-MSA). We observe that across all metrics, **CamelParser2.0** achieves significant improvements over all the reported systems including a 46.3%, 44.1%, and 55.9% error reduction on the LAS, UAS, and LS respectively when compared to the previous SOTA pipeline **CamelParser1.0**. Therefore, we only move forward with testing **CamelParser2.0** for the rest of our experiments.

5.2 Comparing Training Data Configurations

We compare different training datasets and their combination. We use the same **CamelParser2.0** model with CamelBERT-MSA, and report on both gold and predicted tokenization to determine which training data configuration yields the best results on LAS. As seen in Table 3, in the first three columns under the Gold/Predicted Tokenization and CamelBERT headers, the overall trend is that using training data from both PATB-CATiB and CamelTB to train the parser yields the best results on all averages in both the gold and predicted tokenization cases. This is unsurprising given the larger training data size and inclusion of multiple genres. There are some instances where using a smaller training configuration is better than using the larger combined configuration (e.g., Hadith, Hayy, NT); however, they are not statistically significant. On average there are larger gains on both accuracy and robustness to be had from using more training data.

5.3 Comparing BERT Embedding Models

We then experiment with different BERT models as embedding layers (Table 3). Unsurprisingly, the best-performing models on the MSA and CA multi-genre data were CamelBERT-MSA and CamelBERT-CA, not CamelBERT-MIX which was trained with dialectal data.

We observe the following differences depending on the BERT model used. There was a sta-

		Gold Tokenization						Predicted Tokenization				
		CamelBERT					AraBERT	CamelBERT				AraBERT
		MSA			CA	MIX		MSA			CA	
PATB-CATiB		**X**		**X**	**X**	**X**	**X**	**X**		**X**	**X**	**X**
CamelTB			**X**	**X**	**X**	**X**	**X**		**X**	**X**	**X**	**X**
1001	CA	86.2	90.7	**91.9**	91.2	91.2	**92.8**	84.2	88.9	90.1	**90.7**	**90.8**
ALC	MSA	87.3	88.9	**89.2**	88.6	88.9	**90.1**	86.0	87.3	**87.5**	86.5	**88.7**
BTEC	MSA	82.0	86.0	**86.2**	85.2	85.0	**87.1**	81.2	**85.1**	**85.1**	84.5	**86.1**
Hadith	CA	81.2	90.4	90.2	**91.2**	90.7	**91.2**	79.6	87.9	88.2	**88.9**	**88.9**
Hayy	CA	86.6	90.4	90.2	**91.0**	89.3	**91.3**	85.6	88.9	88.7	**89.2**	**89.8**
NT	MSA	74.5	81.1	79.8	**81.2**	80.6	80.2	71.8	**78.5**	76.4	77.1	76.9
Odes	CA	72.7	76.9	77.7	**80.2***	77.1	78.7	68.6	71.7	72.5	**75.2***	74.8
OT	MSA	77.1	82.4	**82.5**	82.3	82.4	**83.4**	74.4	79.4	**79.7**	79.5	**80.5**
QALB	MSA	82.8	87.6	87.6	**88.0**	**88.0**	87.7	82.3	86.7	86.9	**87.3**	**87.3**
Quran	CA	73.8	84.1	84.4	**85.4**	84.3	**85.5**	72.8	**82.3**	83.0	83.2	**83.5**
Sara	MSA	80.2	86.3	**86.6**	86.3	85.9	**87.0**	79.0	**85.0**	84.1	82.6	83.9
WikiNews	MSA	89.0	90.3	**90.4***	86.9	89.5	90.3	88.9	90.1	**90.2***	87.9	90.1
ZAEBUC	MSA	88.2	90.0	**91.1**	89.6	90.9	**92.0**	87.6	89.5	**90.7**	89.3	**91.7**
PATB	MSA	**92.2**	85.2	92.1	90.9	91.5	**92.3**	**91.7**	85.0	91.6	90.4	**91.8**
CamelTB Average		81.7	86.5	86.5	**87.2**	86.4	**87.5**	80.2	84.7	**84.4**	85.6	**85.6**
Total Average		82.4	86.5	86.9	**87.5**	86.8	**87.8**	81.0	84.7	**85.0**	85.9	**86.1**
MSA Average		83.7	86.4	**86.9**	86.6	87.0	**87.8**	82.5	85.2	**85.3**	85.0	**86.3**
CA Average		80.1	86.5	86.9	**89.7**	86.5	87.9	78.2	83.9	84.5	**88.0**	**85.6**

Table 3: The **LAS** of different training configurations on the **Dev sets** of the CamelTB sub-corpora and PATB-CATiB. We test under both Gold and Predicted tokenization conditions, and using different BERT embedding models. The overall best-performing configuration is underlined and in bold, while the best-performing CamelBERT model is in bold. Results with an asterisk indicate statistical significance ($p < 0.05$) for results discussed in Section 5.3.

tistically significant +2.5 gain with gold tokenization and +2.7 gain with predicted tokenization on the LAS from using CamelBERT-CA instead of CamelBERT-MSA on CamelTB-Odes. Furthermore, there was a statistically significant -3.5 drop with gold tokenization and -2.3 drop with predicted tokenization from using CamelBERT-CA over CamelBERT-MSA on CamelTB-WikiNews. However, on average, there is not much of a performance difference between CamelBERT-CA and CamelBERT-MSA; in other cases, the differences were not statistically significant. Nevertheless, it seems that using CamelBERT-CA yields improvements on the parser's performance on CA texts, despite being pretrained on fewer data, and CamelBERT-MSA yields improvements on the performance on MSA texts. These results are consistent with the observations of Inoue et al. (2021) and support the importance of careful selection of the BERT embedding model depending on the data being parsed.

Finally, we also compare with AraBERT, which outperforms CamelBERT on macro average across almost all sub-corpora. AraBERT is better or equal to CamelBERT in 10 out of 14 cases in both Gold and Predicted conditions; however, none of the improvements are statistically significant when compared genre-by-genre.

5.4 CATiB Test Set Results

We report the performance of our best performing models on CATiB formalism from Table 3 on the unseen test sets in Table 4. We observe similar patterns in the results as discussed before. Hence, we make similar recommendations for model selection given the data.

		Gold Tokenization			Predicted Tokenization		
		CamelBERT		AraBERT	CamelBERT		AraBERT
		MSA	CA		MSA	CA	
PATB-CATiB		X	X	X	X	X	X
CamelTB		X	X	X	X	X	X
1001	CA	91.9	**92.0**	**92.2**	89.6	**89.7**	**89.9**
ALC	MSA	**87.5**	86.9	**87.6**	**86.0**	85.7	**86.5**
BTEC	MSA	**84.9**	84.3	**85.5**	**83.6**	83.0	**84.0**
Hadith	CA	92.4	**93.9**	92.2	90.5	**91.8**	90.9
Hayy	CA	91.7	**91.8**	**92.6**	90.3	**90.5**	**91.1**
NT	MSA	**84.7**	84.2	84.6	**79.1**	78.7	78.8
Odes	CA	77.3	**81.5**	78.8	75.3	**77.0**	75.5
OT	MSA	**87.4**	87.3	**87.8**	**82.4**	80.8	**82.6**
QALB	MSA	86.5	**86.7**	**86.8**	**86.1**	85.1	85.9
Quran	CA	82.7	**83.6**	**83.9**	80.3	**80.8**	**81.0**
Sara	MSA	**84.5**	83.9	84.2	**78.9**	78.1	**79.3**
WikiNews	MSA	**90.1**	88.9	**90.1**	**90.0**	88.7	**90.0**
ZAEBUC	MSA	**91.8**	91.4	**92.5**	**90.6**	90.4	**91.3**
PATB	MSA	**91.3**	89.8	**91.3**	**89.6**	**89.6**	**91.0**
CamelTB Average		87.2	**87.4**	**87.6**	**84.8**	84.6	**85.1**
Total Average		87.5	**87.6**	**87.9**	**85.2**	85.0	**85.6**
MSA Average		**87.6**	87.0	**87.8**	**85.1**	84.5	**85.5**
CA Average		87.2	**88.6**	87.9	85.2	**86.0**	85.7

Table 4: The **LAS** of different training configurations on the **Test sets** of the CamelTB sub-corpora and PATB-CATiB. Only the best-performing models from the evaluation on the dev sets are included. The overall best-performing configuration is underlined and in bold, while the best-performing CamelBERT model is in bold.

5.5 Parsing UD with CamelParser2.0

The focus of the previous experiments has been on the performance on the CATiB formalism; however, we also examine the system's performance on UD data. We do so by training our dependency parsing model on PADT-UD and NUDAR, and evaluate on the respective dev and test sets. Due to differing annotation styles between these two UD corpora, cross-evaluation results in poor performance. Hence, we do not report those results here.

We only include **CamelParser2.0** with AraBERT and CamelBERT-MSA because these datasets consist of only MSA, and those models performed the best on MSA data based on our experimentation with CATiB dependency parsing. Furthermore, we use the same disambiguation system to generate the predicted tokens for two reasons: UDPipe 2's disambiguation system was not able to segment the sentences properly so we were unable to align the output for evaluation and secondly we get to observe the performance of the systems while controlling for tokenization accuracy. Results are in Table 5. Furthermore, we only include the best-reported results by Kankanampati et al. (2020) on Gold Tokenization because that is the only experimental setup they report on. We observe that we indeed achieve the SOTA on UD datasets when we compare against UDPipe 2 and Kankanampati et al. (2020). We also observe that CamelBERT-MSA performs better on these datasets than AraBERT.

		Gold Tokenization		Predicted Tokenization	
System	**Train**	**Dev**	**Test**	**Dev**	**Test**
UDPipe 2	PADT-UD	82.5	82.7	81.6	80.9
CamelParser2.0+CamelBERT	PADT-UD	**83.2**	**83.9**	**82.5**	**82.4**
CamelParser2.0+AraBERT	PADT-UD	82.7	83.4	82.2	82.0
Kankanampati et al. (2020)	NUDAR	85.2	84.8	-	-
CamelParser2.0+CamelBERT	NUDAR	**89.1**	**88.9**	**88.7**	**88.8**
CamelParser2.0+AraBERT	NUDAR	89.0	**88.9**	88.1	88.4

Table 5: The **LAS** of different systems evaluated on datasets that use the UD formalism using both gold and predicted tokenization. The first three systems are trained on the PADT and the last three systems are trained on the PATB in the UD formalism (NUDAR). Evaluation is done on the respective **Dev and Test** sets of each corpus.

6 Conclusion and Future Work

We presented **CamelParser2.0**, a new SOTA open-source, Python-based Arabic dependency parser that supports UD and CATiB formalisms and multiple Arabic genres. We make **CamelParser2.0** publicly available.[7] In the future, we plan to continue to enhance the **CamelParser2.0** models and integrate them in downstream applications to support Arabic NLP. We also plan to extend the parser to cover multiple Arabic dialects.

Acknowledgements

We acknowledge the support of the High Performance Computing Center at New York University Abu Dhabi.

Limitations

We recognize that the current parser has limitations, as it is primarily tailored to the most commonly used dependency representation formalisms. However, it does not accommodate other formalisms, such as those rooted in Arabic's extensive traditional syntactic literature (Dukes and Buckwalter, 2010; Halabi et al., 2021). The primary challenge here revolves around the availability of resources. Additionally, we acknowledge that the parser's current focus is on Modern Standard Arabic (MSA) and Classical Arabic (CA), and there is a notable absence of research in the field of Dialectal Arabic parsing (Chiang et al., 2006). It's worth noting that there are numerous pretrained language models available for experimentation. Regrettably, due to limited computational resources, we are unable to explore this avenue. Lastly, we acknowledge that we do not report on extrinsic metrics or performance in downstream tasks.

[7]https://github.com/CAMeL-Lab/camel_parser

References

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.

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).

Wissam Antoun, Fady Baly, and Hazem Hajj. 2020. AraBERT: Transformer-based model for Arabic language understanding. In *Proceedings of the 4th Workshop on Open-Source Arabic Corpora and Processing Tools, with a Shared Task on Offensive Language Detection*, pages 9–15, Marseille, France.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*, pages 149–164, New York City, New York.

David Chiang, Mona Diab, Nizar Habash, Owen Rambow, and Safiullah Shareef. 2006. Parsing Arabic Dialects. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Trento, Italy.

Marie-Catherine De Marneffe, 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 Language Resources and Evaluation Conference (LREC)*, volume 14, pages 4585–92, Reykjavik, Iceland.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Mona Diab, Nizar Habash, Owen Rambow, and Ryan Roth. 2013. LDC Arabic treebanks and associated corpora: Data divisions manual. *arXiv preprint arXiv:1309.5652*.

Timothy Dozat and Christopher D. Manning. 2016.

Deep biaffine attention for neural dependency parsing. *CoRR*, abs/1611.01734.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

Sufeng Duan, Hai Zhao, and Dongdong Zhang. 2023. Syntax-aware data augmentation for neural machine translation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 31:2988–2999.

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.

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.

Nizar Habash, Reem Faraj, and Ryan Roth. 2009. Syntactic Annotation in the Columbia Arabic Treebank. In *Proceedings of the International Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Nizar Habash and Ryan Roth. 2009. CATiB: The Columbia Arabic Treebank. In *Proceedings of the Joint Conference of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 221–224, Suntec, Singapore.

Nizar Habash, Abdelhadi Soudi, and Tim Buckwalter. 2007. On Arabic Transliteration. In A. van den Bosch and A. Soudi, editors, *Arabic Computational Morphology: Knowledge-based and Empirical Methods*, pages 15–22. Springer, Netherlands.

Dana Halabi, Ebaa Fayyoumi, and Arafat Awajan. 2021. I3rab: A new Arabic dependency treebank based on Arabic grammatical theory. *Transactions on Asian and Low-Resource Language Information Processing*, 21(2):1–32.

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.

Dan Jurafsky and James H. Martin. 2009. *Dependency Parsing*. Pearson Prentice Hall.

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).

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Easy-first dependency parsing with hierarchical tree LSTMs. *Transactions of the Association for Computational Linguistics*, 4:445–461.

A Kulmizev. 2023. *The Search for Syntax: Investigating the Syntactic Knowledge of Neural Language Models Through the Lens of Dependency Parsing*. Ph.D. thesis, Uppsala University.

Zuchao Li, Kevin Parnow, and Hai Zhao. 2022. Incorporating rich syntax information in grammatical error correction. *Information Processing & Management*, 59(3):102891.

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.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Alireza Mohammadshahi and James Henderson. 2019. Graph-to-graph transformer for transition-based dependency parsing. *CoRR*, abs/1911.03561.

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, Željko Agić, Lars Ahrenberg, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Eckhard Bick, Cristina Bosco, Gosse Bouma, Sam Bowman, Marie Candito, Gülşen Cebiroğlu Eryiğit, Giuseppe G. A. Celano, Fabricio Chalub, Jinho Choi, Çağrı Çöltekin, Miriam Connor, Elizabeth Davidson, Marie-Catherine de Marneffe, Valeria de Paiva, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Kira Droganova, Puneet Dwivedi, Marhaba Eli, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Cláudia Freitas, Katarína Gajdošová, Daniel Galbraith, Marcos Garcia, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Memduh

Gökırmak, Yoav Goldberg, Xavier Gómez Guinovart, Berta Gonzáles Saavedra, Matias Grioni, Normunds Grūzītis, Bruno Guillaume, Nizar Habash, Jan Hajič, Linh Hà Mỹ, Dag Haug, Barbora Hladká, Petter Hohle, Radu Ion, Elena Irimia, Anders Johannsen, Fredrik Jørgensen, Hüner Kaşıkara, Hiroshi Kanayama, Jenna Kanerva, Natalia Kotsyba, Simon Krek, Veronika Laippala, Phuong Lê Hồng, Alessandro Lenci, Nikola Ljubešić, Olga Lyashevskaya, Teresa Lynn, Aibek Makazhanov, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, André Martins, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Amir More, Shunsuke Mori, Bohdan Moskalevskyi, Kadri Muischnek, Nina Mustafina, Kaili Müürisep, Luong Nguyễn Thị, Huyền Nguyễn Thị Minh, Vitaly Nikolaev, Hanna Nurmi, Stina Ojala, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Guy Perrier, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Lauma Pretkalniņa, Prokopis Prokopidis, Tiina Puolakainen, Sampo Pyysalo, Alexandre Rademaker, Loganathan Ramasamy, Livy Real, Laura Rituma, Rudolf Rosa, Shadi Saleh, Manuela Sanguinetti, Baiba Saulīte, Sebastian Schuster, Djamé Seddah, Wolfgang Seeker, Mojgan Seraji, Lena Shakurova, Mo Shen, Dmitry Sichinava, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Mária Šimková, Kiril Simov, Aaron Smith, Alane Suhr, Umut Sulubacak, Zsolt Szántó, Dima Taji, Takaaki Tanaka, Reut Tsarfaty, Francis Tyers, Sumire Uematsu, Larraitz Uria, Gertjan van Noord, Viktor Varga, Veronika Vincze, Jonathan North Washington, Zdeněk Žabokrtský, Amir Zeldes, Daniel Zeman, and Hanzhi Zhu. 2017. Universal dependencies 2.0.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006. MaltParser: A data-driven parser-generator for dependency parsing. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*, pages 2216–2219, Genoa, Italy.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gulsen Eryigit, Sandra Kubler, Svetoslav Marinov, and Erwin Marsi. 2007. MaltParser: A Language-independent System for Data-driven Dependency Parsing. *Natural Language Engineering*, 13(2):95–135.

Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl 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.

Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, 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 Language Resources and Evaluation Conference (LREC)*, pages 1094–1101, Reykjavik, Iceland.

Anas Shahrour, Salam Khalifa, Dima Taji, and Nizar Habash. 2016. CamelParser: A system for Arabic syntactic analysis and morphological disambiguation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 228–232.

Yan Shao, Christian Hardmeier, and Joakim Nivre. 2018. Universal word segmentation: Implementation and interpretation. *Transactions of the Association for Computational Linguistics*, 6:421–435.

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.

Michalina Strzyz, David Vilares, and Carlos Gómez-Rodríguez. 2019. Viable dependency parsing as sequence labeling. 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 717–723, Minneapolis, Minnesota. Association for Computational Linguistics.

Dima Taji, Nizar Habash, and Daniel Zeman. 2017. Universal dependencies for Arabic. In *Proceedings of the Workshop for Arabic Natural Language Processing (WANLP)*, Valencia, Spain.

Yuanhe Tian, Han Qin, Fei Xia, and Yan Song. 2022. Syntax-driven approach for semantic role labeling. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 7129–7139, Marseille, France.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 195–206, Nancy, France.

Yu Zhang. 2021. SuPar GitHub repository - v1.1.4.

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.

Yue Zhang, Bo Zhang, Zhenghua Li, Zuyi Bao, Chen Li, and Min Zhang. 2022. SynGEC: Syntax-enhanced grammatical error correction with a tailored GEC-oriented parser. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2518–2531, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.