applied sciences

MDPI

Article

Fine-Tuning BERT-Based Pre-Trained Models for Arabic Dependency Parsing

Sharefah Al-Ghamdi *, Hend Al-Khalifa and Abdulmalik Al-Salman

College of Computer and Information Sciences, King Saud University, P.O. Box 2614, Riyadh 13312, Saudi Arabia; hendk@ksu.edu.sa (H.A.-K.); salman@ksu.edu.sa (A.A.-S.)
* Correspondence: sharefah@ksu.edu.sa

Abstract: With the advent of pre-trained language models, many natural language processing tasks in various languages have achieved great success. Although some research has been conducted on fine-tuning BERT-based models for syntactic parsing, and several Arabic pre-trained models have been developed, no attention has been paid to Arabic dependency parsing. In this study, we attempt to fill this gap and compare nine Arabic models, fine-tuning strategies, and encoding methods for dependency parsing. We evaluated three treebanks to highlight the best options and methods for fine-tuning Arabic BERT-based models to capture syntactic dependencies in the data. Our exploratory results show that the AraBERTv2 model provides the best scores for all treebanks and confirm that fine-tuning to the higher layers of pre-trained models is required. However, adding additional neural network layers to those models drops the accuracy. Additionally, we found that the treebanks have differences in the encoding techniques that give the highest scores. The analysis of the errors obtained by the test examples highlights four issues that have an important effect on the results: parse tree post-processing, contextualized embeddings, erroneous tokenization, and erroneous annotation. This study reveals a direction for future research to achieve enhanced Arabic BERT-based syntactic parsing.

Keywords: syntactic parsing; dependency parsing; fine-tuning methods; machine learning; neural networks; deep learning; language models; Arabic natural language processing

Citation: Al-Ghamdi, S.; Al-Khalifa, H.; Al-Salman, A. Fine-Tuning BERT-Based Pre-Trained Models for Arabic Dependency Parsing. *Appl. Sci.* **2023**, *13*, 4225. https://doi.org/10.3390/app13074225

Academic Editor: Kuei-Hu Chang

Received: 14 February 2023
Revised: 24 March 2023
Accepted: 25 March 2023
Published: 27 March 2023

1. Introduction

Syntactic parsing is an automatic analysis of natural language that defines the grammatical arrangements of words and their relationships by assigning a syntactic structure to a sentence [1,2]. In dependency representation, the parse tree describes the syntactic structure using binary relations called dependencies. Each relation is composed of two lexical words arguments: the dependent (modifier) word and the head word [3]. Syntactic analysis aids in the meaning comprehension of a sentence for most natural languages. Therefore, it is a critical level of analysis for other natural language processing (NLP) applications, such as machine translation, information extraction, and generation [4–6]. Dependency parsing is a challenging task for morphologically rich languages with free word order, long-distance, and non-projective dependencies [7].

Arabic is a Semitic language that is considered to possess a rich morphology and complex syntax [8,9]. Recent research has focused on investigating the diverse approaches to dependency parsing and has strived to improve the parsers' accuracy for several languages in a multilingual manner [10,11]. However, no profound study has been conducted on dependency parsing using neural network techniques for Arabic compared to Amharic, Tamil, Greek, German, and Turkish [12–17].

This study investigated several directions related to dependency parsing as a sequence labeling problem. Specifically, we fine-tuned Bidirectional Encoder Representations from Transformers (BERT) for Arabic parsing. We further provide a number of comparisons

between existing BERT-based pre-trained Arabic models, different fine-tuning techniques, and three encoding methods for the position of the head token. A comprehensive exploration of these aspects is presented and discussed. To present research gaps and priorities for improvement, four weaknesses were argued, namely, the parse tree post-processing, contextualized embeddings, erroneous part of speech (POS) annotation, and erroneous tokenization. The exploration findings indicate that Arabic dependency parsing as sequence labeling using a fine-tuned, BERT-based, pre-trained model could achieve promising results. In addition, comparisons and discussions were conducted on three Arabic treebanks, which are presented later in this paper. Our work provides researchers with explored choices for Arabic dependency parsing as a sequence labeling problem.

The main contributions of this paper are summarized in the following two goals:

- Examining the different aspects of fine-tuning BERT-based models for syntactic dependency analysis of the Arabic language.
- Identifying the models and methods that provide the best results and highlighting their weaknesses.

The remainder of this paper is organized as follows. Section 2 introduces background knowledge regarding the Arabic language and dependency parsing as a sequence labeling problem, as well as an overview of BERT-based models. Section 3 presents the related work. The corpora and their statistics are presented in Section 4. Section 5 explores the used methodology. The evaluation results are discussed in Section 6. The error analysis is introduced in Section 7, followed by the conclusions and future work in Section 8.

2. Background

This section introduces the Arabic language and its varieties, the formulation of dependency parsing as a sequence labeling problem, and BERT pre-trained language models.

2.1. Arabic Language

Arabic is a Semitic language spoken primarily in Arab countries. It is divided into three types: Dialect Arabic (DA), Modern Standard Arabic (MSA), and Classical (Traditional or Quranic) Arabic (CA). These forms differ in morphology, syntax, and lexical mixes. In addition, Arabic is generally a free word request language, although the essential word order in Arabic is verb-subject-object (VSO). This causes difficulty in NLP tasks, especially syntactic parsing [18].

Less-resourced languages, such as classical Arabic, the ancient form of Arabic, need more attention from the NLP community. Therefore, in [19], we constructed the Classical Arabic Poetry Dependency treebank (ArPoT). In contrast, in this work, we have explored the dependency parsing for different Arabic treebanks using models pre-trained on the Arabic language variants: MSA, CA, and a mix of MSA, DA, and CA.

2.2. Sequence Labeling for Dependency Parsing

Most NLP tasks can be directly considered multiclass classification problems (sequence tagging) because they are classified into pre-defined categories. For example, Name Entity Recognition (NER) identifies each entity as a person, organization, or place. Further, in part-of-speech (POS) tagging, the words are categorized as noun, verb, preposition, and so on. However, in the case of dependency syntactic parsing, the problem is more challenging because syntactic analysis is the task of mapping the sequence of words to its parse tree.

The parse tree in dependency treebanks is a directed graph that shows each token in the sentence as dependent on only one other token (its head). Moreover, it specifies the syntactic relation between the token and its head. Thus, the dependency parsing task classes are defined depending on the length of sentences and the syntactic reactions used in the treebank annotation. Each token requires a label (class) containing information about the head token's position and the syntactic dependency relation. The head position indicates the index of the head (parent) token of the dependent token in the parse tree of

a sentence. The second part is the syntactic dependency relation between the dependent token and its head. Thus, the concatenation of the head position and the dependency relation forms one class in the classification problem.

Strzyz et al. [20] encoded the dependency tree as a sequence of labels, where each token T_i in the sequence $T_0, \ldots, T_n$ is associated with a particular label of the form $(x_i \text{ @ } l_i)$, where x_i encodes the position of T_i's head, and l_i is the dependency relation label between T_i and its head. Three of the four encoding types for x_i that were applied by Strzyz et al. [20] were evaluated in this study: naive positional encoding, relative positional encoding, and relative POS-based encoding. These three methods were chosen because they can represent non-projective dependencies that exist in Arabic sentences. Figure 1 shows the three applied encodings on an Arabic parse tree from the ArPoT treebank, whereas Table 1 lists them, along with the explanation of labels.

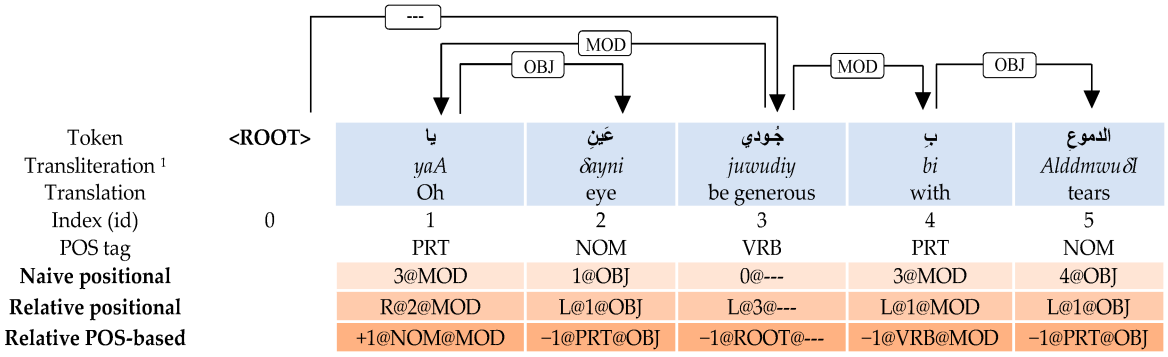


Figure 1. Types of encoding on an example for Arabic parse tree from ArPoT [19,20]. [1] In this work, we applied Arabic to Buckwalter transliteration using CAMeL Tools at: https://github.com/CAMeL-Lab/camel_tools (accessed on 4 February 2023).

Table 1. Head position encoding methods with an explanation of labels.

Identifier	Encoding	Label	Label Meaning
E1	Naive positional	4@OBJ	Id of the head is 4, and the syntactic relation is OBJ
E2	Relative positional	R@2@MOD	The head is the second right (R) token, and the syntactic relation is MOD
E3	Relative POS-based	−1@VRB@MOD	The head is the first left token that has VRB pos tag, and the syntactic relation is MOD

2.3. BERT Pre-Trained Language Models

In the NLP community, large-scale unlabeled corpora are utilized, and self-supervised learning is employed to produce deep pre-trained models, such as GPT and BERT, which capture linguistic knowledge from the text. The development of such pre-trained models has risen to prominence following the early emergence of pre-training shallow networks, such as Word2Vec and GloVe, to capture the semantic meanings of words, as well as the development of Transformers architectures [21].

The introduction of BERT, a bidirectional deep Transformer, has significantly developed the field of pre-trained models. Adapting BERT for certain NLP tasks can be accomplished in two stages: pre-training and fine-tuning [22]. In this work, we primarily applied Arabic pre-trained language models and fine-tuned them for a token-level NLP subtask, which is the syntactic dependency parsing. We tuned the token classifier to assign a syntactic dependency label to each token in the input sentence.

For Arabic, there are several publicly available BERT-based pre-trained language models. To select a proper pre-trained language model for syntactic dependency parsing, we compared nine BERT-based models, as shown in Table 2.

Table 2. Pre-trained language models.

	Model Name *	Variants	Ref.
Camel-MSA	CAMeL-Lab/BERT-based-arabic-camelbert-msa	MSA	[23]
Camel-CA	CAMeL-Lab/BERT-based-arabic-camelbert-ca	CA	[23]
Camel-mix	CAMeL-Lab/BERT-based-arabic-camelbert-mix	MSA/DA/CA	[23]
multilingual	BERT-based-multilingual-uncased	MSA	[22]
Arabic	asafaya/BERT-based-arabic	MSA	[24]
AraBERTv1	aubmindlab/BERT-based-AraBERTv01	MSA	[25]
AraBERTv2	aubmindlab/BERT-based-AraBERTv02	MSA	[25]
ARBERT	UBC-NLP/ARBERT	MSA	[26]
GigaBERT	lanwuwei/GigaBERT-v4-Arabic-and-English	MSA	[27]

* Model name as in: https://huggingface.co (accessed on 24 October 2021).

3. Related Work

This section discusses recent work on syntactic parsing using pre-trained language models, including BERT for sequence labeling and fine-tuning for various downstream tasks. The need for exploring BERT fine-tuning for syntactic dependency parsing is emphasized for its potential in improving Arabic syntactic parsers.

Compared to transition- and graph-based methods, Strzyz et al. [20] showed that it is possible to obtain fast and accurate parsers using a conventional BILSTM-based model for sequence labeling. However, the study was conducted only on the English language. Later, Vilares et al. [28] cast parsing as sequence labeling, and then with just pre-trained encoders, syntactic parsing models were presented. The work compared different pre-trained architectures, such as GloVe, word2vec, FastText, ELMO, and BERT, on an English treebank only. The fully tuned BERT surpassed existing sequence tagging parsers on the same English dependency treebank. The paper gave a modest presentation of the scores for some of the dependency relations in the Treebank, such as obj, advmod, det, and nsubj, but did not analyze why recognizing some relations was more accurate than others, and there was no in-depth analysis of the dataset errors.

Recent work [13] explored the use of BERT models jointly with a MALT parser for transition-based dependency parsing of Tamil, a morphologically rich, agglutinative language. The results show that appropriate vector representations from BERT trained on Indic languages directly improve the overall parser performance.

An abundance of recent work has evaluated the fine-tuning of pre-trained language models for downstream tasks. Fine-tuning BERT-based Arabic pre-trained language models for sequence labeling problems has shown great success in named entity recognition, POS tagging, sentiment analysis, dialect identification, and poetry meter classification [23,29]. Consequently, an investigation of BERT fine-tuning methods for syntactic dependency parsing is also needed. This will provide a foundation for improving the development of Arabic syntactic parsers.

From an analytical perspective, several studies [30–32] have attempted to understand and explain what occurs during BERT fine-tuning. Merchant et al. [30] explained the extent to which the layers were affected during the fine-tuning process. The results show that, unlike natural language inference and reading comprehension, dependency parsing reconfigures most of the BERT models and involves deeper changes to the encoder (more top layers of BERT are affected).

In this work, we intend to explore the Arabic dependency parsing as a sequence labeling problem. We aim to identify the optimal selections to perform syntactic dependency parsing by fine-tuning BERT-based pre-trained Arabic models. To implement this task, we aim to identify the best head position encoding method, pre-trained model, and fine-tuning strategy. Moreover, we present a detailed analysis of the errors given by the optimal parsing model to understand what happened during fine-tuning, and we shed light on the reasons for the weaknesses of the chosen model.

4. Corpora

Our experiments were conducted on three Arabic dependency treebank datasets: the Prague Arabic dependency treebank (PADT) [33], the conversion of the Penn Arabic Treebank (ATB) part2 v3.1 [34] as part of the Columbia Arabic Treebank (CATiB), and the dependency treebank for Classical Arabic poetry (ArPoT v1.0) [19].

Corpora Statistics

PADT is the largest corpus in this study, followed by CATiB and ArPoT. In Table 3, we show the number of tokens in the training, development, and testing datasets for each treebank, whereas Figure 2 shows the percentages of the data split for them.

Table 3. Fine-tuned datasets statistics.

Treebank	All	Training	Development	Testing
PADT	282,384	223,881 *	30,239	28,264
CATiB	169,319	135,219 *	16,972 *	17,128
ArPoT	35,459	28,506	2771	4182

* These statistics before some tokens were excluded from sentences that exceeded the maximum length of BERT (512).

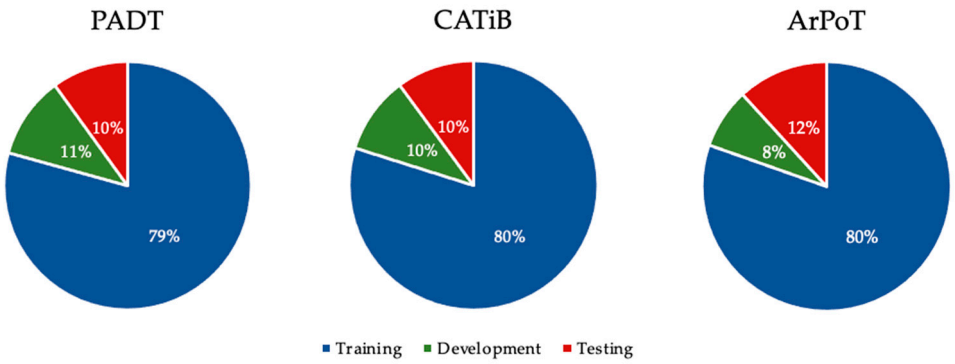


Figure 2. Data split into training, development, and testing for each dataset.

PADT and CATiB are MSA newswires text, whereas ArPoT is for CA verses. PADT is in the Universal Dependencies annotation style [33], whereas CATiB and ArPoT follow the annotation rules of [35]. However, unlike ArPoT, CATiB contains the "PNX" POS tag for punctuation, because ArPoT sentences are unpunctuated. Further, CATiB has one token with the "UNK" POS tag, which means the (unknown) type. Table 4 shows the number of POS tags and dependency relation labels for each treebank.

Table 4. POS tags and Dependency Relations Labels in each dataset.

Treebank	POS Tags	Dependency Relations Labels
PADT	16	33
CATiB	6 *	8
ArPoT	5	8

* Without "UNK" tag.

The percentages of POS tags and dependency relation labels for all corpora are presented in Figure 3. The left column (a, c, and e) for the POS tags and the right column (b, d, and f) show the dependency relation labels. The distributions show the nature of the language, where some POS tags and dependency relations are more common than others. In Section 7, we study the results of parser accuracy on unbalanced distribution for different types of tokens. More detailed statistics of the corpora used for fine-tuning are presented in Table 5, including:

1. Sentences: the number of sentences in the corpus.
2. Max_L: the maximum length for the corpus sentences.
3. Avg_L: the average number of tokens for sentences divided by the total number of words (tokens) in the corpus.
4. Max_DL: the maximum dependency length (distance between dependents and heads).
5. Avg_DL: the average dependencies length.
6. TTR: the type–token ratio (TTR), which is the total number of unique words (types) divided by the total number of words (tokens) in a corpus.

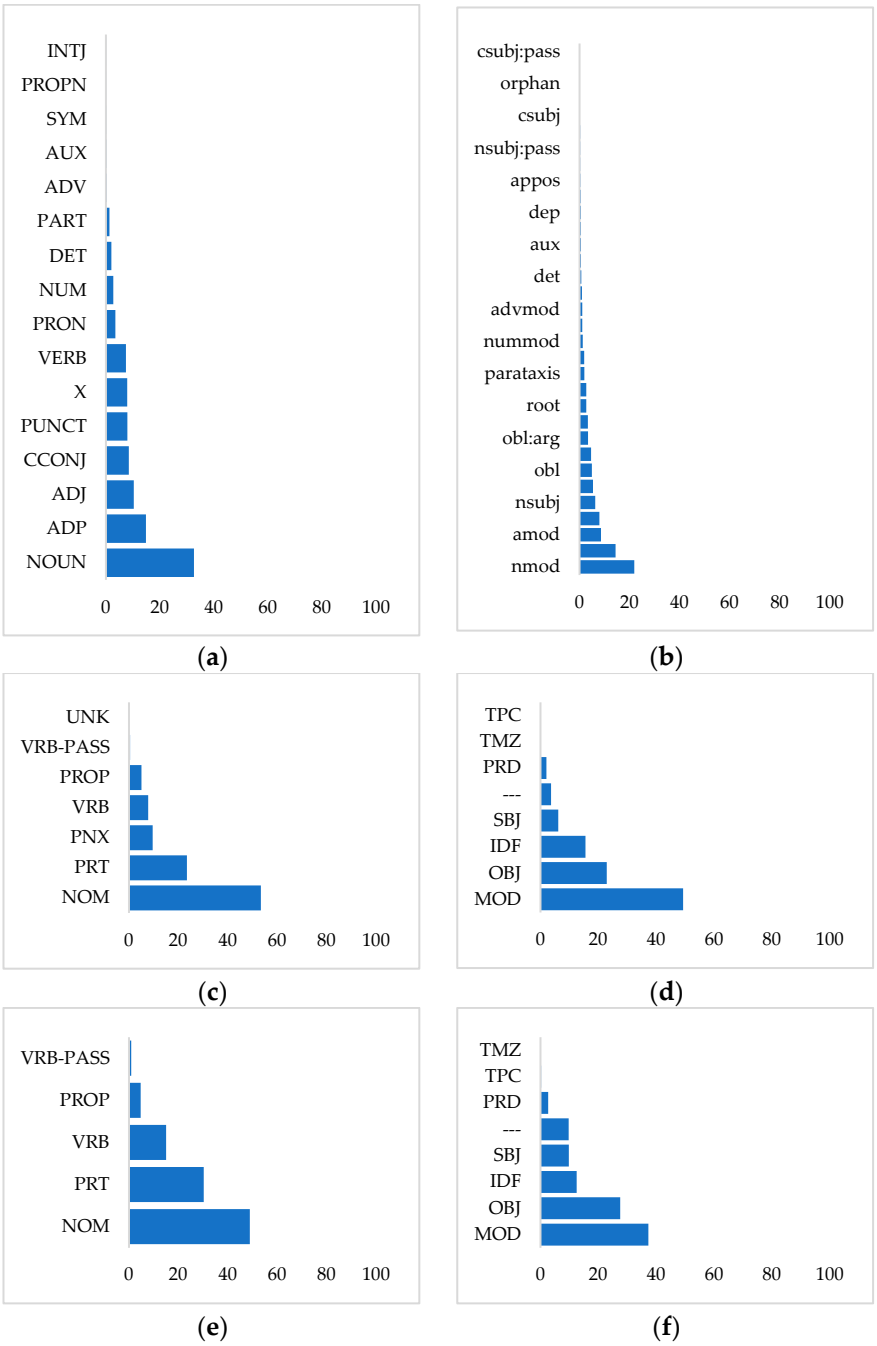


Figure 3. Distributions of POS tags and dependency relations labels (DRs). (**a**) PADT—POS. (**b**) PADT—DR. (**c**) CATiB—POS. (**d**) CATiB—DR. (**e**) ArPoT—POS. (**f**) ArPoT—DR.

Table 5. Number of sentences (Sentences), maximum sentence length (Max_L), average sentence length (Avg_L), maximum dependencies length (Max_DL), average dependencies length (Avg_DL), and TTR type–token ratio (TTR).

	PADT	CATiB	ArPoT
Sentences	7664	2591	2400 *
Max_L	398	609	68
Avg_L	36.85	65.28	14.77
Max_DL	397	479	46
Avg_DL	4.07	4.21	2.07
TTR	9.34	9.5	28.26

* After sentence alignment in [19] for 2685 verses.

Generally, sentences in CATiB are longer than those in PADT, and ArPoT sentences are the shortest because of the nature of classical verses. In addition, the average length of dependencies in CATiB is larger. For the lexical variety measured by TTR [36], ArPoT has more vocabulary variety.

5. Methodology

This section describes our methodology, including the evaluation metrics and a detailed description of the experimental results. In this study, we aimed to demonstrate the best possible options for fine-tuning BERT models for Arabic dependency parsing by conducting the following experiments.

- The most effective head position encoding method for each treebank is identified in Section 6.1.
- The accurate pre-trained model for the syntactic parsing task on each corpus was differentiated by tuning all Arabic pre-trained models under study. In the first experiment, head position encoding methods with higher scores were utilized. In this process, out-of-vocabulary (OOV) words and the average number of tokens per word (P/W) created by tokenizers were verified (see Section 6.2).
- A comparison of fine-tuning techniques is presented in Section 6.3. Based on the previous sections of the experiments, we utilized the best encoding method and pre-trained model for each treebank.

For evaluation metrics, two standard evaluation metrics for dependency parsing were used: unlabeled attachment score (UAS), which represents the percentage of tokens assigned to the correct head, and labeled attachment score (LAS), which is the percentage of tokens assigned to both the correct head and the correct dependency label between the token and its head [37]. In addition, in some sections, the label score (LS) was used as the number of correct labels for dependency.

In all experiments, we relied on the training code for [28], but we allowed multiple roots while it was seen in the corpora. We used GPU in Google Colab and conducted the experiments in April 2022 (before the Python packages upgrading in November 2022, in case of future reconsideration). The same experimental settings were applied in all runs: maximum length = 512, batch size = 8, epoch = 10, and learning rate = 5×10^{-5}.

6. Evaluation and Discussion

Several experiments have been carried out to evaluate the dependency parsing for Arabic as a sequence labeling problem by fine-tuning the BERT-based pre-trained models. First, the three encoding methods presented in Section 2.2 were evaluated to measure their parsing accuracy on MSA and CA treebanks. Then, in Section 2.3, comparison studies were performed on nine different BERT-based Arabic models. Finally, three fine-tuning techniques were compared using the optimal selections from the first two experiments.

6.1. Choosing the Best Encoding Method

Although the encoding of the head position is a great step towards simplifying syntactic parsing as a sequence labeling problem, the effect of a wide variety of labels on the parser accuracy requires more investigation. Generally, the number of labels (classes) in the dataset is a key factor affecting the accuracy of the classification models [38,39]. Thus, during the exploration of the best encoding, we shed light on the number of different labels using each of the methods under study (see Section 2.2).

The specifications of each dataset, such as sentence length (number of tokens' ids), number of POS tags, and number of dependency relations, provide a diversity of encoding labels (classes). Therefore, we presented the number of labels for each dataset in Table 6. It shows that encoding method E1 produces more classes than E2 and E3 for all treebanks. On the other hand, E2 carries fewer labels for ArPoT, and E3 carries fewer labels for PADT and CATiB. In response, the results of fine-tuning the BERT-based models using the three labeling methods in Table 7 show that the smaller the number of labels, the more accurate the classifier. E2 reported a higher UAS/LAS for ArPoT, and E3 reported the best results for PADT and CATiB. Therefore, we named the models in the following sections using ArPoT_{E2}, PADT_{E3}, and CATiB_{E3}. CAMeL BERT-based pre-trained models have been utilized here: Camel-CA for the classical corpus (ArPoT) and Camel-MSA for the newswire corpora CATiB and PADT.

Table 6. Number of labels in each treebank using three head position encodings.

	E1	**E2**	**E3**
PADT	4513	2017	1793
CATiB	1923	563	297
ArPoT	305	164	173

Table 7. Average UAS/LAS over 5 runs on development (DEV) and test datasets using three head position encodings.

Treebank (Model)	**DEV**			**TEST**		
	E1	**E2**	**E3**	**E1**	**E2**	**E3**
PADT (Camel-MSA)	64.25/60.65	81.79/77.46	83.12/79.00	85.48/55.40	82.18/77.77	83.06/79.17
CATiB (Camel-MSA)	46.88/45.89	83.89/82.40	86.35/85.23	44.26/43.07	83.75/82.20	86.52/85.33
ArPoT (Camel-CA)	64.63/57.77	78.74/70.77	75.47/70.28	64.39/58.38	78.91/72.39	76.70/72.06

6.2. Choosing the Best BERT Model

The Arabic BERT-based models were evaluated on the corpora (PADT, CATiB, and ArPoT), and the UAS/LAS for the testing datasets are presented in Table 8. In our observations, the most accurate model among the nine fully tuned models was Ara-BERTv2. This corresponds to the results of [23], where AraBERTv2 was the best for six other NLP subtasks. Furthermore, the dependency parsing task is sensitive to the language variants of the pre-trained model. For instance, Camel-MSA and Camel-CA have the same setup, but they are trained on different Arabic variants, MSA and CA, respectively. Hence, the UAS/LAS of the fine-tuned Camel-MSA model on the MSA corpora (PADT and CATiB) is higher than that of the Camel-CA model. On the other hand, the fine-tuned Camel-CA model performed better on the Classical Arabic corpus (ArPoT). Similarly, the multilingual model (multilingual) reported worse scores for all corpora. Table 9 displays the Arabic training datasets used for pre-training the BERT-based models under study.

Table 8. Test UAS/LAS for all Arabic pre-trained models under study using the best encoding methods for each treebank.

	TEST UAS/LAS		
Model	**$PADT_{E3}$**	**$CATiB_{E3}$**	**$ArPoT_{E2}$**
Camel-MSA	83.10/79.17	86.47/85.29	78.72/71.83
Camel-CA	80.78/76.60	84.95/83.32	79.65/72.21
Camel-mix	82.37/78.37	85.64/84.35	78.48/71.95
multilingual	74.02/68.54	76.22/72.50	72.33/60.71
Arabic	80.02/76.52	82.65/80.59	73.58/64.30
AraBERTv1	82.76/78.82	86.76/85.57	77.76/70.95
AraBERTv2	84.03/80.26	87.54/86.41	79.79/74.13
ARBERT	80.37/76.11	78.31/75.95	75.06/66.19
GigaBERT	80.41/76.06	83.29/81.28	73.39/62.31

Table 9. Data sources for the Arabic BERT-based models pre-training.

Source\Model	**Camel-MSA**	**Camel-CA**	**Camel-mix**	**Multilingual**	**Arabic**	**AraBERTv1**	**AraBERTv2**	**ARBERT**	**GigaBERT**
AR_Wiki	✓		✓		✓	✓	✓	✓	✓
El-Khair	✓		✓			✓	✓	✓	
OSIAN	✓		✓			✓	✓	✓	
OpenITI		✓	✓						
OSCAR 2019								✓	✓
OSCAR 2020	✓		✓		✓		✓		
Gigaword	✓		✓					✓	✓
Dialectal Corpora			✓						
Multi_Wiki				✓					
News						✓			
As-safir							✓		
Hindawi								✓	
Code-switch									✓

To investigate the effect of the pre-training configurations on fine-tuning the dependency parsing, we compared the pre-training settings of the models and the pre-training data size. Table 10 shows that training with larger data and vocabulary sizes does not necessarily lead to higher parsing accuracy. For instance, although ARBERT is trained on more than double the size of the pre-training data and vocabulary for AraBERTv1, it performed worse on all corpora. This confirms the Inoue et al. [23] suggestion for other NLP tasks where data size may not be an important factor in fine-tuning the performance. Looking at the pre-training objective and tokenizer in Table 10, we can see that they do not affect the model accuracy. They are the same in the case of the BERT-based models with a higher and lower UAS/LAS (AraBERTv2 and multilingual, respectively). Therefore, focusing on the most accurate BERT-based model for all corpora (AraBERTv2), we can see that the training batches sizes are larger, which verifies the Anil et al. [40] proposed enhancement of using an increasing batch size schedule for pre-training to grow the accuracy of the BERT-based model.

Table 10. Arabic BERT models configurations. MLM: masked language modeling. WWM: whole word masking. SWM: sub-word masking. NSP: next sentence prediction. WP: word piece. SP: sentence piece. MSL: maximum sentence length. TD: training duration.

Model	(MLM) Pre-Training Objective	Tokenizer	Batch Size	MSL	TD (Days)	Total Steps	Words	Data Size	Vocabulary Size
Camel-MSA	WWM + NSP	WP	1024/256	128/512	~4.5	1 M	12.6 B	107 GB	30 K
Camel-CA	WWM + NSP	WP	1024/256	128/512	~4.5	1 M	847 M	6 GB	30 K
Camel-mix	WWM + NSP	WP	1024/256	128/512	~4.5	1 M	17.3 B	167 GB	30 K
multilingual	SWM + NSP	WP	-	-	-	-	-	-	120 K
Arabic	WWM + NSP	WP	128	-	-	4 M	8.2 B	95 GB	32 K
AraBERTv1	WWM + NSP	SP	512/128	128/512	4	1.25 M	2.7 B	23 GB	60 K
AraBERTv2	WWM + NSP	WP	2560/384	128/512	36	3 M	8.6 B	77 GB	60 K
ARBERT	WWM + NSP	WP	256	128	16	8 M	6.5 B	61 GB	100 K
GigaBERT	SWM + NSP	WP	512	128/512	-	1.48 M	10.4 B	-	50 K

To explore the effect of OOV tokens on the parsing accuracy of fine-tuned BERT-based models, we present the rate of OOV tokens generated by the tokenizers for each dataset in Table 11.

Table 11. The rate of OOV tokens indicated by each model over the fine-tuning data.

	Camel-$_{MSA,CA,mix}$	Multilingual	Arabic	AraBERTv1	AraBERTv2	ARBERT	GigaBERT
$PADT_{E3}$	1.27	10.82	10.82	1.76	1.54	12.46	10.82
$CATiB_{E3}$	0.15	17.10	14.71	2.59	0.32	18.42	14.71
$ArPoT_{E2}$	0.00	3.43	3.43	0.00	0.00	3.43	3.43

In all corpora, the ratio of OOV produced by the CAMeL pre-trained models was lower. This was followed by AraBERTv2, which was the most accurate on dependency parsing. However, after checking the OOV tokens, we observed that punctuation is the most common OOV given by the CAMeL BERT models (Camel-MSA and Camel-CA Camel-mix) and AraBERT models (AraBERTv1 and AraBERTv2). Thus, these models did not record any OOV tokens in the unpunctuated treebank (ArPoT).

In contrast, the rest of the models would identify very common words with "ء/HAMZAH" as OOV, such as "إلى" and "إن", because of encoding issue (We used tokenizer.convert_tokens_to_ids method rather than tokenizer.encode). Such particles are essential for sentence comprehension and affect how the models understand a sentence.

In the same context, Table 12 shows the average number of sub-words (pieces per word) given by the tokenizer of each model (We used pytorch-pretrained-bert, which gives different tokenization than the transformers), where the lower value means that most words are not divided into sub-words by the model tokenizer. Overall, the rates show that, on average, the tokenizers did not split the word into more than one word. A higher rate of pieces per word was recorded by the multilingual model on all treebanks. The ARBERT model records the lowest splitting rate on PADT (mostly, each word is one token), where it is trained on a 100 K vocabulary size. However, it was not as accurate as the AraBERT and CAMeL models. This can result from the factors mentioned earlier, such as the OOV rate and training batch size.

Table 12. Average number of sub-words provided by each model's tokenizer.

	Camel-$_{MSA,CA,mix}$	Multilingual	Arabic	AraBERTv1	AraBERTv2	ARBERT	GigaBERT
PADT$_{E3}$	1.1	1.6	1.1	1.1	1.1	1.0	1.1
CATiB$_{E3}$	1.2	1.5	1.1	1.1	1.1	1.1	1.1
ArPoT$_{E2}$	1.3	1.7	1.2	1.4	1.2	1.1	1.2

Interestingly, the words in ArPoT exhibited a higher rate of splitting into more sub-words than in the other two corpora. This is a result of classical vocabulary not being utilized in MSA. For example, the AraBERTv1 tokenizer splits the word "مهدود/collapse" into three pieces: "مهد/cradle", "##و", and "##د". The effect of tokenization that changes word meaning will be discussed in Section 7.4.

6.3. Evaluation of the Tuning Techniques

While the BERT models are well-regulated and relatively complex, several research papers have studied the modeling of their performance and investigated the need to tune all the pre-trained layers to accomplish NLP tasks [31]. Merchant et al. [30] have shown that dependency parsing requires tuning more layers to capture the syntactic information of the English dataset. To explore Arabic treebanks, we evaluated the following three fine-tuning techniques using the AraBERTv2 BERT model:

1. Full Tuning (FT): Further train the entire architecture of the pre-trained model on the experimental datasets.
2. Full Freezing (FF): All layers of the pre-trained model are frozen and tested on the datasets under study. In this case, the weights of the 12 layers are not updated.
3. Partial Freezing (PF): The weights of some layers of the pre-trained model are kept frozen, while other layers are retrained.

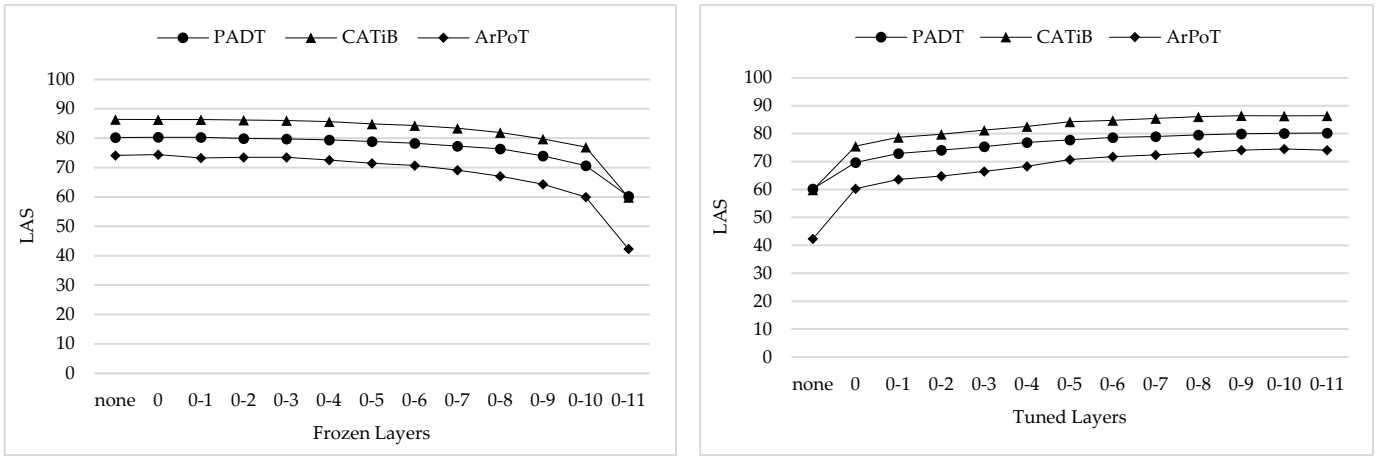
We also examined the attachment of two neural network (NN) layers to the FT and FF, and then trained the new model. Two types of neural networks have been explored, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) [41,42]. FT+LSTM and FF+LSTM are fully tuned and fully freeze models with two-layered BILSTM. In contrast, FT+GRU and FF+GRU are fully tuned and fully freeze models with two attached (GRU) layers.

Table 13 shows the results of testing the six models (FT, FT+LSTM, FT+GRU, FF, FF+LSTM, and FF+GRU). It is noticeable that FT without attached NN layers recorded the best UAS/LAS for all three treebank test sets. On the other hand, GRU layers performed better than the LSTM layer, which is consistent with Alsaaran and Alrabiah's results on the Arabic NER task [29]. However, the fully tuned BERT model with a two-layered BILSTM decoder achieved better results on the English treebank in [20].

Table 13. UAS/LAS on the test datasets for AraBERTv2 BERT model fully tuned (FT) and fully freezes (FF) with and without additional layers (GRU and LSTM).

	FT	FT+LSTM	FT+GRU	FF	FF+LSTM	FF+GRU
PADT$_{E3}$	84.03/80.26	82.90/78.92	83.64/79.74	65.30/60.25	72.54/67.72	75.44/70.98
CATiB$_{E3}$	87.54/86.41	85.92/84.82	87.50/86.39	62.88/59.88	77.29/74.74	79.76/77.66
ArPoT$_{E2}$	79.79/74.13	76.88/70.83	79.51/73.63	56.84/42.32	69.87/60.93	72.64/64.73

We tuned the BERT layers partially and tracked the model accuracy using partial freezing of the bottom layers progressively. We froze one more layer in each run. Moreover, we gradually tuned one more bottom layer for each run and reported the LAS of the model. Figure 4a,b shows the results of both experiments, which correspond to the hypothesis of [30] that the hierarchical nature of analysis requires additional layers to understand syntactic dependencies.



(**a**) (**b**)

Figure 4. LAS of BERT-based pre-trained models on three corpora by (**a**) freezing an increasing number of layers during fine-tuning and (**b**) fine-tuning at earlier layers.

More detailed scores for this experiment are listed in Table 14. Although fine-tuning of the higher layers is required in all cases, some datasets converge in earlier layers. For example, CATiB and ArPoT provided a higher UAS/LAS on the test dataset by tuning the first 10 and 11 layers, respectively. On the other hand, the best scores on the PADT required all 12 layers, which indicates full tuning. Moreover, the results show that excluding some of the higher layers from the tuning process is more effective than full tuning, for example, 87.65/86.49 versus 87.54/86.41 for CATiB and 80.32/74.53 versus 79.79/74.13 for ArPoT. Section 7 analyzes these errors from different perspectives.

Table 14. UAS/LAS of partial freezing (PF) on test datasets.

Frozen Layers	$PADT_{E3}$	$CATiB_{E3}$	$ArPoT_{LR}$	Tuned Layers	$PADT_{E3}$	$CATiB_{E3}$	$ArPoT_{LR}$
0	84.01/80.34	87.41/86.32	80.27/74.41	0	74.11/69.72	77.92/75.54	69.58/60.28
0–1	83.85/80.28	87.39/86.32	79.48/73.27	0–1	77.08/72.93	80.83/78.72	71.66/63.61
0–2	83.68/79.93	87.24/86.15	79.63/73.48	0–2	78.41/74.12	81.56/79.76	73.08/64.83
0–3	83.58/79.74	87.13/86.02	79.48/73.46	0–3	79.49/75.37	82.95/81.29	74.03/66.48
0–4	83.18/79.40	86.71/85.62	78.86/72.55	0–4	80.92/76.88	84.12/82.62	75.59/68.29
0–5	82.65/78.85	86.02/84.86	78.00/71.47	0–5	81.80/77.78	85.51/84.27	76.83/70.73
0–6	82.06/78.28	85.48/84.32	77.21/70.73	0–6	82.59/78.64	86.01/84.77	77.79/71.76
0–7	81.07/77.32	84.59/83.41	75.63/69.11	0–7	82.97/79.01	86.62/85.46	78.55/72.36
0–8	80.17/76.35	83.21/81.94	73.58/67.07	0–8	83.48/79.57	87.21/86.09	78.89/73.19
0–9	77.89/73.99	81.28/79.73	71.64/64.32	0–9	83.83/79.98	87.65/86.49	80.01/74.10
0–10	74.54/70.66	78.75/76.95	67.86/59.97	0–10	84.00/80.15	87.46/86.39	80.32/74.53
0–11	65.30/60.25	62.88/59.88	56.84/42.32	0–11	84.03/80.26	87.54/86.41	79.79/74.13

7. Error Analysis

This section examines the findings and highlights the major problems in our experiments. To do so, we tracked the UAS and LS for each POS tag and each dependency relation label in the dataset. These two scores show errors in the head assignment and dependency relations labeling. Because some POS tags are presented in the fine-tuning datasets more than others, we chose to explore whether rare POS tags would lead to lower scores on parsing. Figure 5 shows the UAS and LS over each POS tag on the three treebanks during the fine-tuning process. Tags and labels are listed in the figure's legend from most common to least common in the data, according to the statistics in Figure 3.

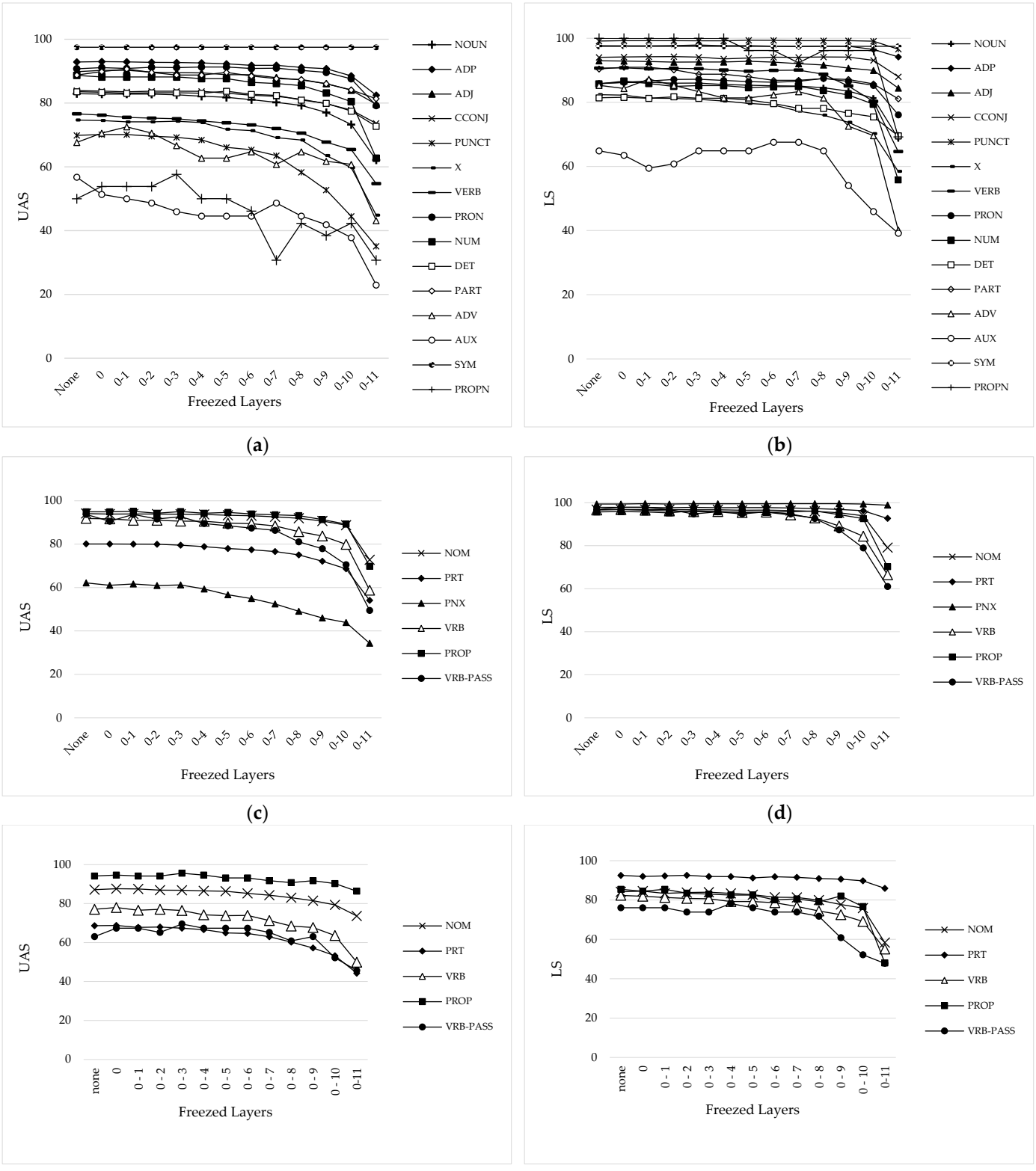


Figure 5. UAS and LS over POS tags and dependency relations in each test dataset freezing an increasing number of layers during fine-tuning. (**a**) PADT UAS. (**b**) PADT LS. (**c**) CATiB UAS. (**d**) CATiB LS. (**e**) ArPoT UAS. (**f**) ArPoT LS.

Generally, full tuning improves the scores of most POS tags in the corpora. In addition, it has been shown that the LS for dependency relation labeling (Figure 5b,d,f) is higher

than the UAS for head finding (Figure 5a,c,e). For most POS tags, the LS of fully tuned models is greater than 80%, whereas the UAS reaches less than 40% in some cases.

In terms of head finding, we discovered that several tags covering a significant number of training examples, such as punctuation and particles (PUNCT in PADT and PRT in CATiB and ArPoT), reported a lower UAS than other uncommon tags, such as symbols and proper nouns (SYM in PADT and PROP in CATiB and ArPoT). This clearly indicates that the spread of a POS type does not necessarily make head-finding for parser models more precise. Due to the concatenation of the head position and relational dependencies in the sequence model's label (class), which results in various labels for the same token, prediction is impaired. For example, although the punctuation token "." in PADT and the particle token "و/w/and" in CATiB and ArPoT are frequent, the model accuracy for their label prediction is relatively low (see Table 15).

Table 15. Most frequent tokens for common and uncommon POS tags in the corpora. TP: token ratio of POS tag; C: common; UC: uncommon.

Treebank	Token	POS	TP (Train-Dev)	TP (Test)	Labels	Accuracy
PADT	%	SYM (UC)	87.76	97.5	2	100
	.	PUNCT (C)	33.79	34.92	159	60.53
CATiB	مصر/mSr/Egypt	PROP (UC)	4.81	3.75	16	100.0
	و/w/and	PRT (C)	25.69	26.73	130	70.86
ArPoT	الله/Allh/God	PROP (UC)	4.63	6.8	8	92.86
	و/w/and	PRT (C)	22.43	21.28	28	62.19

In contrast, POS tags with syntactic labels that lean toward stability in most sentences have a higher parsing accuracy. For example, although the SYM in PADT is a rare POS tag in the test set, it achieved a high UAS score (97.5), even with full freeze layers. This is because of the absence of variations in the SYM tokens. In the test set, the "%" tokens constituted 97.5% of the total SYM tokens. These tokens are frequently led by prior number tokens (−1@NUM) and always have an "nmod" dependency relation in the test sentence (see Table 15).

To better understand the faults and effects of data instances on the tuned model's performance, we compared the LAS of the FT and FF models for all test sets of the corpora (Figure 6a–c). These findings demonstrate that different tuning data for the same NLP task drove various learning outcomes. For example, after full fine-tuning, the LAS is improved for all test examples in CATiB, as shown in Figure 6b. However, the LAS on the fully tuned model decreased in 2.1% and 2.6% of the PADT and ArPoT test instances, respectively. The most drastic accuracy drop is seen in example 504 in PADT (−25% in LAS), followed by example 269 in ArPoT (−23.08).

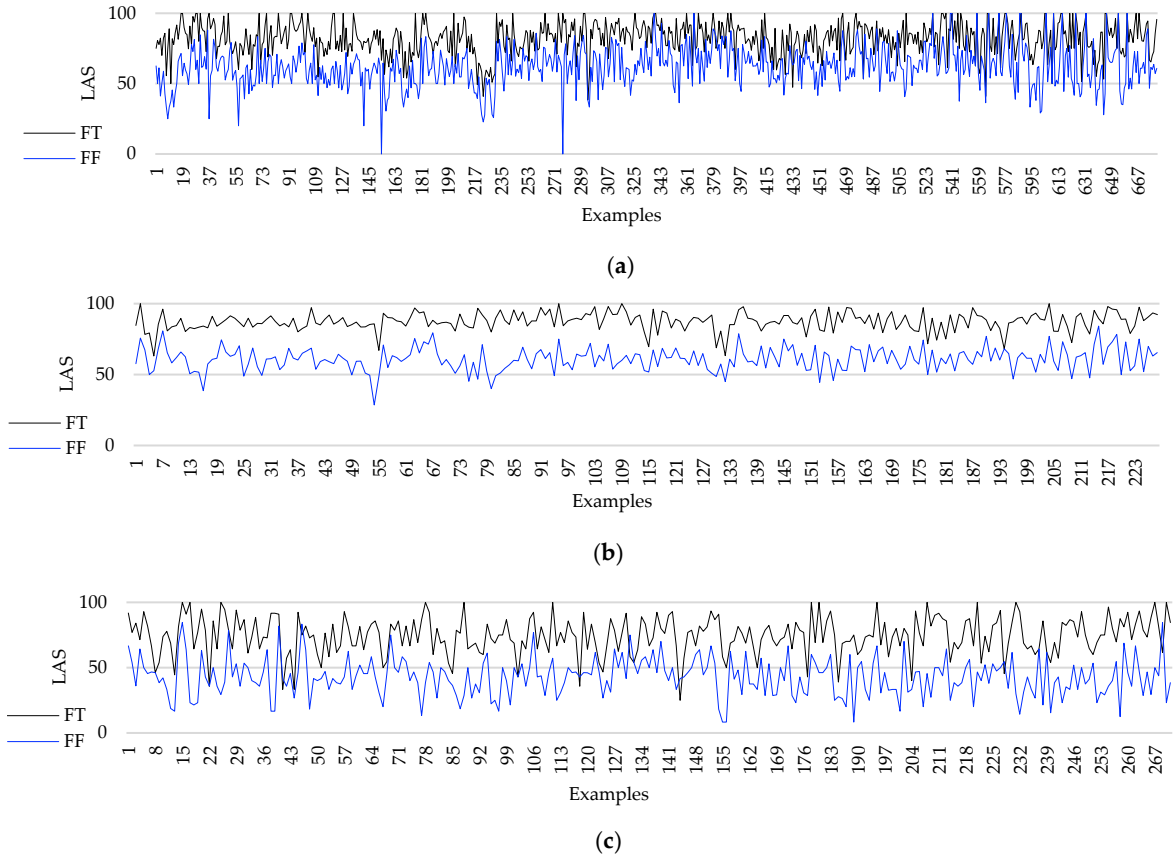


Figure 6. LAS for Fully Tuned (FT) and Fully Freeze (FF) models on all test examples. (**a**) PADT. (**b**) CATiB. (**c**) ArPoT.

Using the FT model, three degenerated examples in PADT and ArPoT (given in Table 16) were reviewed. We revealed the following reasons for the four errors: parse tree post-processing, contextualized embeddings, erroneous tokenization, and erroneous POS annotation. Each of them is discussed in the following subsections with examples.

Table 16. Sample of degenerated test examples in PADT and ArPoT. Each example presented the original sentence in Arabic, its transliteration, and its translation in English.

No	Corps	Example
1	PADT	شركة سعودية تبيع ملابس داخلية من الذهب الخالص $rkp sEwdyp tbyE mlAbs dAxlyp mn Al*hb AlxAlS A Saudi company sells pure gold underwear
2	PADT	عقد اول مناظرة بين المرشحين الديمقراطيين ل الرئاسة فى الولايات المتحدة Eqd Awl mnAZrp byn Almr$Hyn AldymqrATyyn l Alr}Asp fy AlwlAyAt AlmtHdp AlAmrykyp Held the first debate between the Democratic candidates for the presidency in the United States
3	ArPoT	وفي ناتق كان اصطلام سراة هم ليالي أفنى القرح جل إياد w fy nAtq kAn ASTlAm srAp hm lyAly >fnY AlqrH jl <yAd Extermination of Ayad's lofty was at Nateq's nights when a surgeon killed most of them

7.1. Parse Tree Post-Processing

To ensure that the predicted sequence of labels forms a dependency tree (tree structure) that is acyclic and has a root, we applied a post-processing algorithm for [28], with a minor modification to accept multi-heads. Thus, the algorithm changed the labels that are predicted correctly to meet the dependency tree requirements.

This case is illustrated in Figure 7 for Example (1). The model correctly labels the first token (شركة/$rkp/company) with (+1@VERB@nsubj). The label means that the token is headed by the first subsequent (+1) verb (VERB), which is (تبيع/tbyE/sells) with the dependency relation (nsubj) (Figure 7b). Because no token is predicted as a root, the parse tree post-processing assigns the first token to the root and increases the error in the final output (Figure 7c). The red dotted arrow indicates the impactful change.

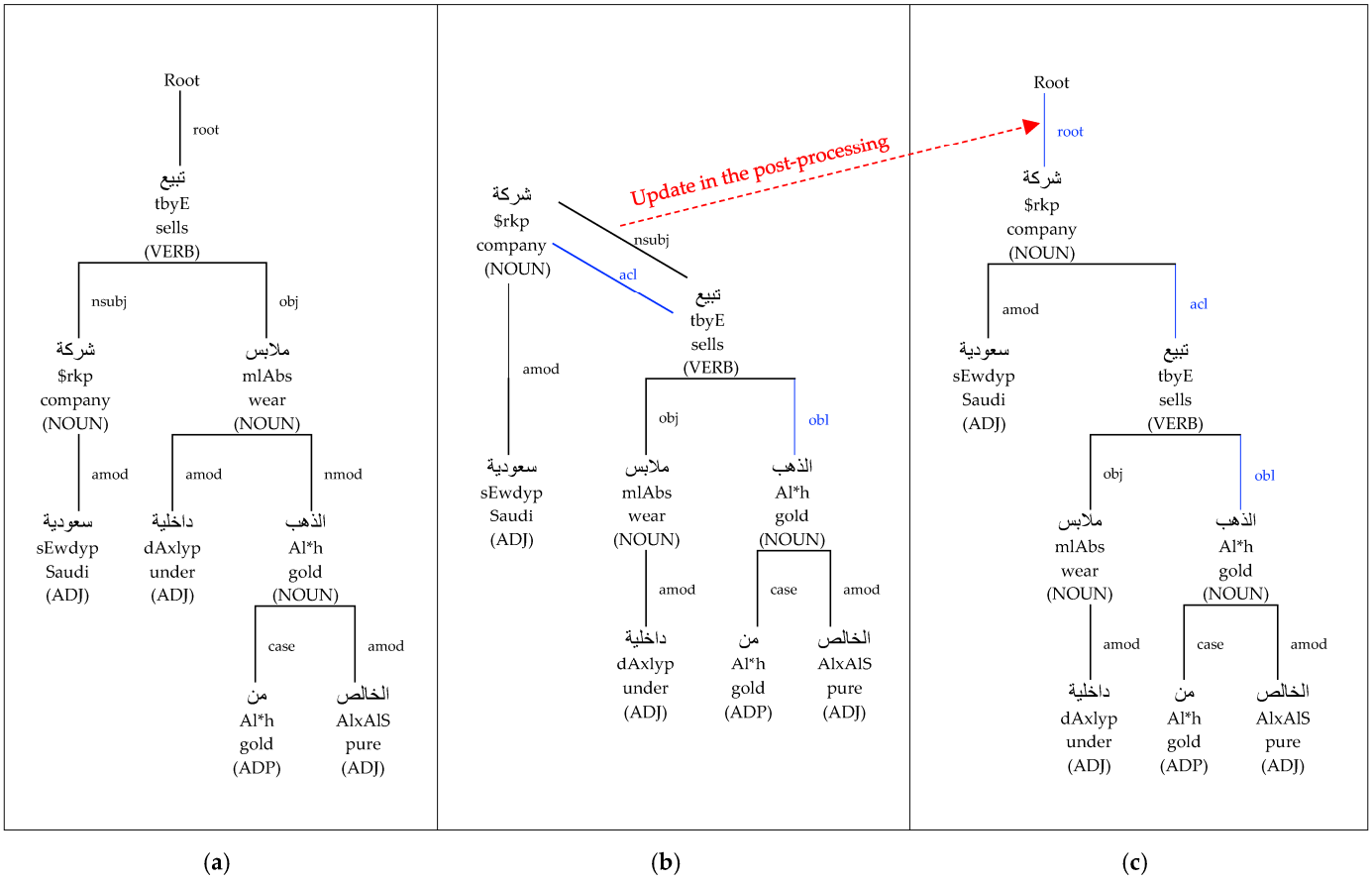
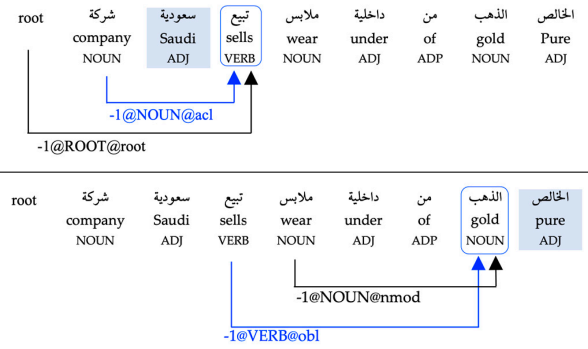
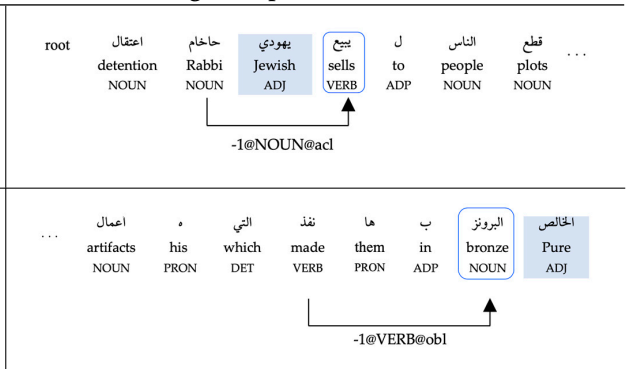
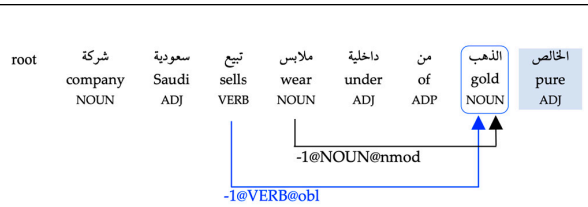
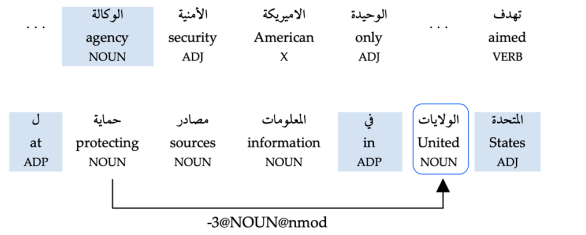
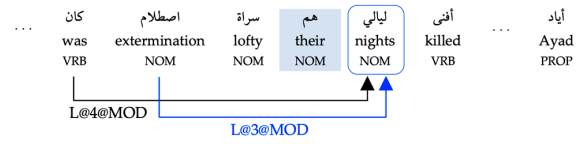
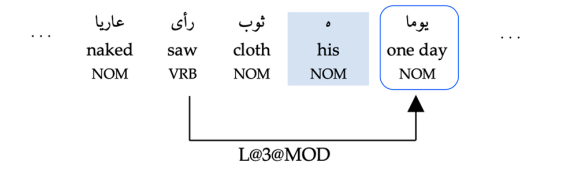


Figure 7. Gold, predicted, and final parse trees for Example (1). Blue edges are incorrect. (**a**) Gold parse tree. (**b**) Predicted parse tree. (**c**) Final parse tree.

7.2. Contextualized Embeddings

The pre-trained BERT structure allows the model to be trained simultaneously, seeing all the left and right contexts of a target word. Therefore, BERT contextualized word embeddings offer high lexical semantics, in contrast to other language models [43]. This is an innovation that is highly productive for most downstream NLP tasks. However, in syntactic parsing, when the same context appears in different grammatical syntactic structures, it will impact the performance of the parsing task. Such a problem arises in predicting some test sentences in the fully tuned parsing models, as shown in Table 17.

Table 17. Examples from the PADT and ArPoT test data that show the effect of contextualized embeddings using obverse-related training sentences. Correct arrows are black, whereas blue arrows indicate wrong assignments. Rectangles enclose the test tokens with incorrect predictions and similar words in the training sentences. In both sentences, the other comparable words in the sequence are shaded.

	Test Example	Training Example with Similar Words
1 PADT		
		
3 ArPoT		

In Example 1, the verb "تبيع/tbyE/(she)sells" is wrongly labeled with (−1@NOUN@acl). This token does not occur in the training and development datasets, but a similar verb, "يبيع/ybyE/(he)sells", has this label. Moreover, it is presented in a similar sequence to "سعودية تبيع/sEwdyp tbyE/Saudi company sells", which is "يهودي يبيع/yhwdy ybyE/Jewish Rabbi sells". This similar context pushed the model to assign the training token labeled to the test token (−1@NOUN@acl). Thus, the verb token is headed by the first prior NOUN token "سعودية/sEwdyp/Saudi" rather than to the root of the parse tree.

The word "الذهب/Al*hb/gold" in the same example was incorrectly tagged with (−1@VERB@obl), despite appearing 16 times in the training and development datasets with the correct label (−1@NOUN@nmod). After a search for a related context, we find a similar word, "البرونز/Albrwnz/Bronze", that appears in a similar sequence with the same posterior adjective "الخالص/AlxAlS/Pure". Therefore, the incorrect predicted label (−1@VERB@obl) assigned "الذهب/Al*hb/gold" to the first left verb "تبيع/tbyE/(she)sells".

Example 2 demonstrates the impact of context on a longer sequence. As an illustration, the phrase "في الولايات المتحدة/fy AlwlAyAt AlmtHdp/in the united states" appears in the training sentences 36 times with the correct spelling of "في/fy/and", but only once with the incorrect "فى". In 10 occurrences, the word "الولايات/AlwlAyAt/states" was labeled with (−1@NOUN@nmod), which is the desired label for the same word in the test sentence. However, only two instances incorrectly anticipated the label (−3@NOUN@obl). One of those

two sentences is shown in the table. It includes similar words, such as the particle "ل/li/for" and the noun "الرئاسة/Alr}Asp/presidency", compared to "/ الوكالةAlwkAlp/Agency".

Attributed to the manipulation of the word order for poetic imperatives and E2 syntactic label encoding, various labels for the same word are shown in the training data. For example, the word "ليالي/lyAly nights" occurs in the training and development sets six times with four different labels. Moreover, none of them is the same as the gold label for that word in example 3 of the ArPoT test set. However, the effect of similar sequences that include similar embeddings leads to unseen (not the same as in training) and incorrect (not the same as gold) labels for that word in the test example. To illustrate such a case, in the training dataset, we find the word "يوما/ywmA/one day", which is preceded by the attached pronoun "ه/h/his". In the test sentence, the similar word "ليالي/lyAly/nights" is also preceded by the attached pronoun "هم/h/their". Moreover, we find training sentences that include the noun word preceded by embeddings comparable to "سراة هم/srAp hm/their lofty" in the test sentence, such as "موالي ها/mwAly hA/her allies", which are all (اسم/noun) and (ضمير متصل/attached pronoun). In 37 instances, the noun token proceeding with comparable embeddings is labeled (L@3@MOD), which is the most common label for that case. Therefore, the word "ليالي/lyAly/nights" is incorrectly assigned to the third left token.

7.3. Erroneous POS Annotation

Because syntactic labeling on PADT was performed using E3 encoding schema (presented in Section 2.2), certain mistakes in the degenerated samples after fine-tuning demonstrate the impact of inaccurate POS tagging of training data on prediction. In these cases, the model generated a valid label; however, the incorrect POS annotation caused the result to be recorded as an incorrect prediction.

For tokens in PADT that cannot be given a legitimate POS category for some reason, the "X" POS tag is used [33]. Figure 8 illustrates the undesirable effects of using such a tag. It displays that the "X" tag is improper for the two tokens "مناظرة/mnAZrp/debate" and "اول/Awl/first" in a phrase from Example 2 in Table 16. The fact is that the first is a noun "NOUN", and the second is an adjective "ADJ". Additionally, although not being a verb in this context, the confusing word "عقد/Eaqd/held" that occurs as the first noun of a news title has the tag "VERB". These errors lead to the following two incorrect predictions:

1. The model correctly labeled the adjective word "اول/Awl/first" with (−1@NOUN@nmod). Thus, it should be headed by the preceding noun word "عقد/Eaqd/held". The algorithm assigned "اول/Awl/first" to the root since no previous token with a "NOUN" POS tag is in the gold annotation (see Figure 8a).
2. The model correctly labeled the noun word "مناظرة/mnAZrp/debate" with (−1@ADJ@nmod); thus, it should be headed by the preceding adjective word "اول/Awl/first". The algorithm assigned "مناظرة/mnAZrp/debate" to the root since no previous token with an "ADJ" POS is in the gold annotation (see Figure 8b).

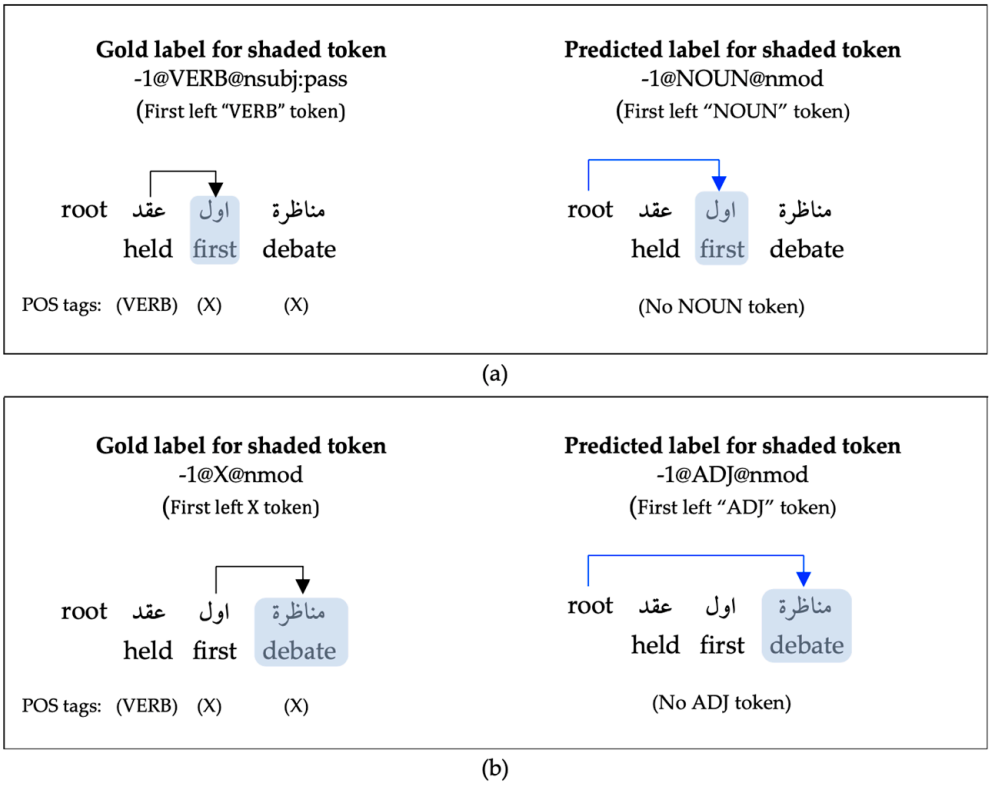


Figure 8. Syntactic label prediction of two tokens (in (**a**,**b**)) with inaccurate POS annotation in test example on PADT.

7.4. Erroneous Tokenization

Tokenization, a critical stage in NLP, separates raw text into words that help interpret word sequences. In this investigation, poor tokenization undoubtedly affected the model's word interpretation and, thus, the syntactical analyzer's performance. The first sub-word of each tokenized word was employed in the preprocessing for sequence labeling during the training phase. Therefore, when the model cannot comprehend the meaning of a sub-word, it may mistakenly identify the sub-word as a verb rather than a noun.

The average of the sub-words produced by the tokenizers of the Arabic BERT models is presented in Table 12 (Section 6.2). Because ArPoT verses words are based on an ancient CA vocabulary, tokenizing them produces the highest number of sub-words. The syntactic labeling for the CA verse (Example 3 in Table 16) shows a prediction mistake because of incorrect tokenization. The word "اصطلام/ASTlAm/extermination" is "اسم فاعل/gerund", which, according to Arabic grammar, must head the next noun in the sequence with a "إضافة/IDF" syntactic relation. That means the expected label for "سراة/srAp/lofty" is "L@1@IDF"; however, the tuned model incorrectly labels it with (L@2@SBJ) after the tokenizer deforms the "اسم فاعل/gerund" by splitting "اصطلام/ASTlAm/extermination" into "اصط/AST" and "لام/lAm". Through investigation, we noticed that the most frequent syntactic label for the second noun in the sequence "verb noun noun" is (L@2@SBJ), particularly when the first noun is an embedding for demonstrative pronouns and attached pronouns, such as "همهاذاكهذا/this-that-him- her-them" (see Figure 9). This indicates that the model does not see the sub-word "اصط/AST" as "اسم فاعل/gerund", but as demonstrative pronouns or attached pronouns.

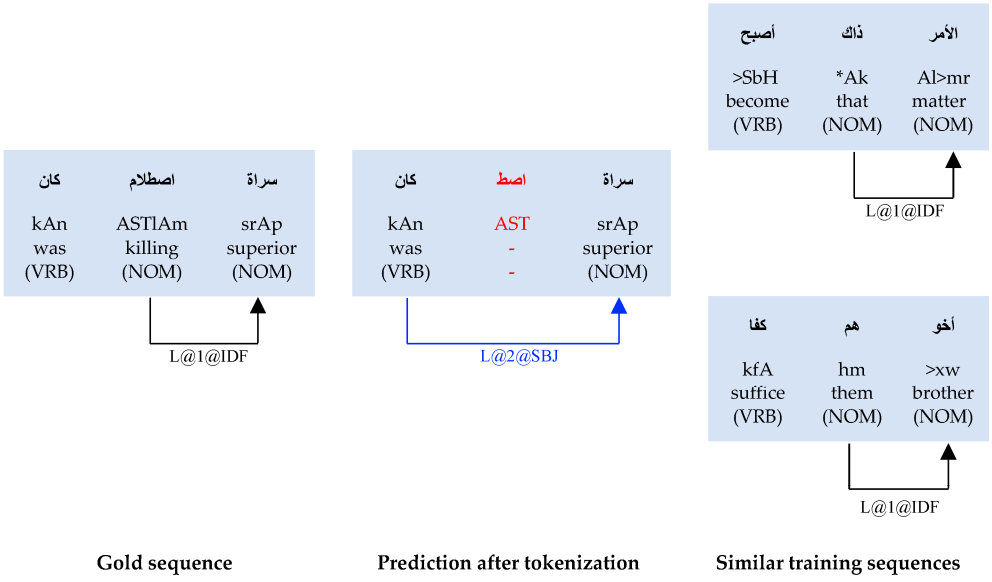


Figure 9. Example of syntactically wrong labeling caused by poor tokenization. The blue arrow represents an incorrect prediction. The word highlighted in red displays an incorrect tokenization.

8. Conclusions and Future Work

In this study, we explored the formulation of dependency parsing for the Arabic language as a sequence labeling problem. We applied three encoding methods to the dependent head position and studied nine Arabic BERT models using different fine-tuning strategies. The effectiveness of the Arabic BERT models varied, but AraBERTv2 performed better on the three studied treebanks. Moreover, because the number of classes (possible labels) given by each dependent head position encoding method relies on the dataset sentence length and annotation schema, there is no preferred encoding method or tuning strategy. However, our experiments confirmed that tuning until the top layers of pre-trained BERT models is required to capture the syntactic information of Arabic sentences.

We plan to work on future enhancements to address the faults stated in the results analysis and to address the issues with the Arabic BERT-based sequence labeling parser. For instance, we can conduct additional research to determine how improved tokenization contributes to better BERT-based parsing. In order to reduce the impact of the labels modifications in post-processing, we might also conduct a further search on how to effectively verify and correct the tree structure of the labeled sequence.

Author Contributions: Conceptualization, S.A.-G. and H.A.-K.; Data curation, S.A.-G.; Formal analysis, S.A.-G. and H.A.-K.; Funding acquisition, S.A.-G., H.A.-K. and A.A.-S.; Investigation, S.A.-G. and H.A.-K.; Methodology, S.A.-G. and H.A.-K.; Project administration, S.A.-G. and H.A.-K.; Resources, S.A.-G., H.A.-K. and A.A.-S.; Software, S.A.-G.; Supervision, H.A.-K. and A.A.-S.; Validation, S.A.-G. and H.A.-K.; Visualization, S.A.-G. and H.A.-K.; Writing—original draft, S.A.-G.; Writing—review & editing, H.A.-K. and A.A.-S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the Deanship of Scientific Research at King Saud University through the initiative of DSR Graduate Students Research Support (GSR).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: PADT datasets can be accessed at https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-3238 (accessed on 4 April 2021). Restrictions apply to the availability of CATiB (the conversion of Penn Arabic Treebank (ATB) part2 v3.1) datasets, which are used under the license of the College of Computer and Information Sciences at King Saud University. The ArPoT treebank is provided by [19] at https://github.com/ArPoT-KSU/ArPoT_v1.0 (accessed on 21 June 2022).

Acknowledgments: The authors would like to thank King Saud University and the College of Computer and Information Sciences. Additionally, the authors would like to thank the Deanship of Scientific Research at King Saud University for funding and supporting this research through the initiative of DSR Graduate Students Research Support (GSR).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jurafsky, D.; Martin, J. *Speech and Language Processing*, 3rd ed.; Prentice Hall: Hoboken, NJ, USA, 2019.
2. Chowdhary, K. Natural Language Processing. In *Fundamentals of Artificial Intelligence*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 603–649.
3. Kübler, S.; McDonald, R.; Nivre, J. Dependency Parsing. *Synth. Lect. Hum. Lang. Technol.* **2009**, *1*, 1–127.
4. Zhang, M.; Li, Z.; Fu, G.; Zhang, M. Syntax-Enhanced Neural Machine Translation with Syntax-Aware Word Representations. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), Minneapolis, MN, USA, 2–7 June 2019; pp. 1151–1161.
5. Sachan, D.S.; Zhang, Y.; Qi, P.; Hamilton, W.L. Do Syntax Trees Help Pre-Trained Transformers Extract Information? In Proceedings of the EACL, Online, 19–23 April 2021.
6. Guo, Q.; Qiu, X.; Xue, X.; Zhang, Z. Syntax-Guided Text Generation via Graph Neural Network. *Sci. China Inf. Sci.* **2021**, *64*, 152102. [CrossRef]
7. Legrand, J.; Collobert, R. Deep Neural Networks for Syntactic Parsing of Morphologically Rich Languages. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Berlin, Germany, 7–12 August 2016; pp. 573–578.
8. Dehdari, J.; Tounsi, L.; van Genabith, J. Morphological Features for Parsing Morphologically-Rich Languages: A Case of Arabic. In Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages, Dublin, Ireland, 6 October 2011; pp. 12–21.
9. Tsarfaty, R. Syntax and Parsing of Semitic Languages. In *Natural Language Processing of Semitic Languages*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 67–128.
10. Nivre, J. Multilingual Dependency Parsing from Universal Dependencies to Sesame Street. In Proceedings of the International Conference on Text, Speech, and Dialogue, Brno, Czech Republic, 8–11 September 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 11–29.
11. Barry, J. Investigating Multilingual Approaches for Parsing Universal Dependencies. Ph.D. Thesis, Dublin City University, Dublin, Ireland, 2022.
12. Seyoum, B.E.; Miyao, Y.; Mekonnen, B.Y. Comparing Neural Network Parsers for a Less-Resourced and Morphologically-Rich Language: Amharic Dependency Parser. In Proceedings of the First Workshop on Resources for African Indigenous Languages, Marseille, France, 11–16 May 2020; pp. 25–30.
13. Kumar, C.A.; Maharana, A.; Murali, S.; Premjith, B.; Kp, S. BERT-Based Sequence Labelling Approach for Dependency Parsing in Tamil. In Proceedings of the Second Workshop on Speech and Language Technologies for Dravidian Languages, Dublin, Ireland, 26 May 2022; pp. 1–8.
14. Sarveswaran, K.; Dias, G. ThamizhiUDp: A Dependency Parser for Tamil. In Proceedings of the 17th International Conference on Natural Language Processing (ICON), Patna, India, 18–21 December 2020; pp. 200–207.
15. Prokopidis, P.; Piperidis, S. A Neural NLP Toolkit for Greek. In Proceedings of the 11th Hellenic Conference on Artificial Intelligence, Athens, Greece, 2–4 September 2020; pp. 125–128.
16. Fankhauser, P.; Do, B.-N.; Kupietz, M. Evaluating a Dependency Parser on DeReKo. In Proceedings of the 8th Workshop on Challenges in the Management of Large Corpora, Marseille, France, 11–16 May 2020; pp. 10–14.
17. Özateş, Ş.B.; Özgür, A.; Güngör, T.; Başaran, B.Ö. A Hybrid Deep Dependency Parsing Approach Enhanced With Rules and Morphology: A Case Study for Turkish. *IEEE Access* **2022**, *10*, 93867–93886. [CrossRef]
18. Shaalan, K.; Siddiqui, S.; Alkhatib, M.; Abdel Monem, A. Challenges in Arabic Natural Language Processing. In *Computational Linguistics, Speech and Image Processing for Arabic Language*; World Scientific: Singapore, 2019; pp. 59–83.
19. Al-Ghamdi, S.; Al-Khalifa, H.; Al-Salman, A. A Dependency Treebank for Classical Arabic Poetry. In Proceedings of the Sixth International Conference on Dependency Linguistics (Depling, SyntaxFest 2021), Sofia, Bulgaria, 21–25 March 2021; pp. 1–9.
20. Strzyz, M.; Vilares, D.; Gómez-Rodríguez, C. Viable Dependency Parsing as Sequence Labeling. In Proceedings of the 2019 Conference of the North, Minneapolis, MN, USA, 2–7 June 2019; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 717–723.
21. Han, X.; Zhang, Z.; Ding, N.; Gu, Y.; Liu, X.; Huo, Y.; Qiu, J.; Yao, Y.; Zhang, A.; Zhang, L. Pre-Trained Models: Past, Present and Future. *AI Open* **2021**, *2*, 225–250. [CrossRef]
22. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *arXiv* **2018**, arXiv:1810.04805.

23. Inoue, G.; Alhafni, B.; Baimukan, N.; Bouamor, H.; Habash, N. The Interplay of Variant, Size, and Task Type in Arabic Pre-Trained Language Models. In Proceedings of the Sixth Arabic Natural Language Processing Workshop, Kiev, Ukraine, 19 April 2021; pp. 92–104.
24. Safaya, A.; Abdullatif, M.; Yuret, D. Kuisail at Semeval-2020 Task 12: Bert-Cnn for Offensive Speech Identification in Social Media. In Proceedings of the Fourteenth Workshop on Semantic Evaluation, Online, 12–13 December 2020; pp. 2054–2059.
25. Antoun, W.; Baly, F.; Hajj, H. Arabert: Transformer-Based Model for Arabic Language Understanding. *arXiv* **2020**, arXiv:2003.00104.
26. Abdul-Mageed, M.; Elmadany, A.; Nagoudi, E.M.B. ARBERT & MARBERT: Deep Bidirectional Transformers for Arabic. *arXiv* **2020**, arXiv:2101.01785.
27. Lan, W.; Chen, Y.; Xu, W.; Ritter, A. Gigabert: Zero-Shot Transfer Learning from English to Arabic. In Proceedings of the 2020 Conference on Empirical Methods on Natural Language Processing (EMNLP), Online, 19–20 November 2020.
28. Vilares, D.; Strzyz, M.; Søgaard, A.; Gómez-Rodríguez, C. Parsing as Pretraining. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 9114–9121.
29. Alsaaran, N.; Alrabiah, M. Classical Arabic Named Entity Recognition Using Variant Deep Neural Network Architectures and BERT. *IEEE Access* **2021**, *9*, 91537–91547. [CrossRef]
30. Merchant, A.; Rahimtoroghi, E.; Pavlick, E.; Tenney, I. What Happens to Bert Embeddings during Fine-Tuning? *arXiv* **2020**, arXiv:2004.14448.
31. Rogers, A.; Kovaleva, O.; Rumshisky, A. A Primer in Bertology: What We Know about How Bert Works. *Trans. Assoc. Comput. Linguist.* **2020**, *8*, 842–866. [CrossRef]
32. Wu, Z.; Liu, N.F.; Potts, C. Identifying the Limits of Cross-Domain Knowledge Transfer for Pretrained Models. *arXiv* **2021**, arXiv:2104.08410.
33. Bouma, G.; Seddah, D.; Zeman, D. Overview of the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies. In Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies, Seattle, WA, USA, 9 July 2020; pp. 151–161.
34. Diab, M.; Habash, N.; Rambow, O.; Roth, R. LDC Arabic Treebanks and Associated Corpora: Data Divisions Manual. *arXiv* **2013**, arXiv:1309.5652.
35. Habash, N.; Faraj, R.; Roth, R. Syntactic Annotation in the Columbia Arabic Treebank. In Proceedings of the MEDAR International Conference on Arabic Language Resources and Tools, Cairo, Egypt, 22–23 April 2009; Volume 83.
36. Richards, B. Type/Token Ratios: What Do They Really Tell Us? *J. Child Lang.* **1987**, *14*, 201–209. [CrossRef] [PubMed]
37. Green, N. Dependency Parsing. In Proceedings of the WDS 2011, Proceedings of Contributed Papers, Matfyzpress, Prague, 31 May–3 June 2011; pp. 137–142.
38. Abramovich, F.; Pensky, M. Classification with Many Classes: Challenges and Pluses. *J. Multivar. Anal.* **2019**, *174*, 104536. [CrossRef]
39. Van Thinh, T.; Duong, P.C.; Nasahara, K.N.; Tadono, T. How Does Land Use/Land Cover Map's Accuracy Depend on Number of Classification Classes? *Sola* **2019**, *15*, 28–31. [CrossRef]
40. Anil, R.; Ghazi, B.; Gupta, V.; Kumar, R.; Manurangsi, P. Large-Scale Differentially Private Bert. *arXiv* **2021**, arXiv:2108.01624.
41. Graves, A. Long Short-Term Memory. In *Supervised Sequence Labelling with Recurrent Neural Networks*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 37–45.
42. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.
43. Wiedemann, G.; Remus, S.; Chawla, A.; Biemann, C. Does BERT Make Any Sense? Interpretable Word Sense Disambiguation with Contextualized Embeddings. *arXiv* **2019**, arXiv:1909.10430.