Open Access Library Journal
2026, Volume 13, e14897
ISSN Online: 2333-9721
ISSN Print: 2333-9705

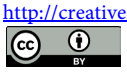
AMorph: An End-to-End Morpheme Level Natural Language Processing Pipeline for Amharic

Natnael Tamirat Molla, Shi Ming

School of Computer Science, Hubei University of Technology, Wuhan, China
Email: mollanatnaeltamirat@hbut.edu.cn, shiming@hbut.edu.cn

How to cite this paper: Molla, N.T. and Ming, S. (2026) AMorph: An End-to-End Morpheme Level Natural Language Processing Pipeline for Amharic. *Open Access Library Journal*, **13**: e14897.
https://doi.org/10.4236/oalib.1114897

Received: January 19, 2026
Accepted: February 9, 2026
Published: February 12, 2026



Abstract

Morphological segmentation is foundational for Natural Language Processing in morphologically rich languages, such as Amharic, yet progress is constrained by limited gold annotations and fragmented toolchains. We present an end-to-end framework that jointly addresses data creation and modeling for Amharic segmentation, part-of-speech tagging, and dependency parsing. Our approach begins with a silver-data generation pipeline that bootstraps segmentation labels from a rule-based analyzer and refines them through automated review, large language model assisted verification, and human-in-the-loop correction. Using the resulting supervision, we train an XLM-RoBERTa segmenter with a connectionist temporal classification (CTC) based character transduction objective, enabling reliable morpheme boundary prediction without explicit character-morpheme alignment. We then introduce a unified multi-task toolkit model that replaces the common practice of training separate systems per task. A shared pretrained encoder is jointly optimized for POS tagging and dependency parsing to better capture cross-task linguistic regularities while remaining parameter-efficient. The observed morpheme segmentation, POS tagging, and dependency parsing results support the conclusion that analyzer-bootstrapped supervision combined with multilingual pretrained encoders is effective for Amharic morphosyntactic modeling in low-resource settings. We release an open-source toolkit with simple APIs and an interactive visualization interface, enabling users to run the pipeline and inspect intermediate and final outputs for practical Amharic NLP development. API, documentation, and pre-trained models are available at https://github.com/Netela-lab/AMorph.

Subject Areas

Natural Language Processing, Computational Linguistics, Low-Resource

Languages

Keywords

Amharic, Morphological Segmentation, Finite-State Morphology, Weak Supervision, Connectionist Temporal Classification (CTC), Multi-Task Learning, Dependency Parsing, Universal Dependencies

1. Introduction

Natural language processing toolkits have rapidly improved the accessibility of high-quality linguistic processing, but their strongest support still concentrates on high-resource languages and standard tokenization assumptions [1]-[5]. For morphologically rich languages such as Amharic, a single orthographic token can encode multiple morphemes expressing tense, agreement, and case, making morpheme-aware analysis essential for reliable downstream modeling [6] [7].

We study the task of building an end-to-end Amharic processing stack that maps raw text to Universal Dependency (UD) style morpheme level analyses: 1) morpheme segmentation, 2) morpheme level Part of Speech (POS) tagging (with language-specific XPOS), and 3) dependency parsing over morphemes. We define success by segmentation accuracy on a held out human-validated set, POS accuracy, and dependency parsing quality measured by Labeled Attachment Score (LAS), together with practical usability through an inference API that operates on raw text.

This problem is difficult for three reasons. First, high-quality morpheme-level supervision for Amharic is limited, and creating it manually is expensive at scale [8]. Second, rule-based analyzers encode valuable linguistic knowledge but are brittle under spelling variation, loanwords, and domain shift, and their outputs are not automatically aligned with UD segmentation conventions [6] [9]. Third, training separate neural models per task on small datasets risks overfitting and fails to exploit the shared structure across segmentation, tagging, and parsing, a scenario where multi-task learning is often beneficial [10] [11].

Overview of our neural Amharic NLP pipeline. Our system takes raw Amharic text as input, and produces UD-style morpheme-level CoNLL-U annotations. Besides this neural pipeline, we also provide an easy-to-use API and a visualization interface.

Existing work only partially closes this gap. Widely used toolkits (e.g. CoreNLP, UDPipe, Flair, spaCy, Stanza) offer strong pipelines, but they do not provide an Amharic morpheme-level stack trained to UD-style conventions, nor a recipe that systematically converts analyzer output into reliable supervision [1]-[5]. On the resource side, prior Amharic corpora and treebank efforts demonstrate progress but do not, by themselves, deliver a unified, morpheme-level neural pipeline that is robust in realistic text processing scenarios [8] [12].

We propose an end-to-end framework (Figure 1): 1) bootstraps UD-style morpheme-level supervision from a rule-based analyzer using Large Language Model (LLM) and human-in-the-loop correction, and 2) trains multilingual transformer models a connectionist temporal classification (CTC) segmenter and a multi-task morphosyntactic analyzer to produce segmentation, POS tags, and dependencies in one stack [9] [10] [13] [14]. Our contributions are:

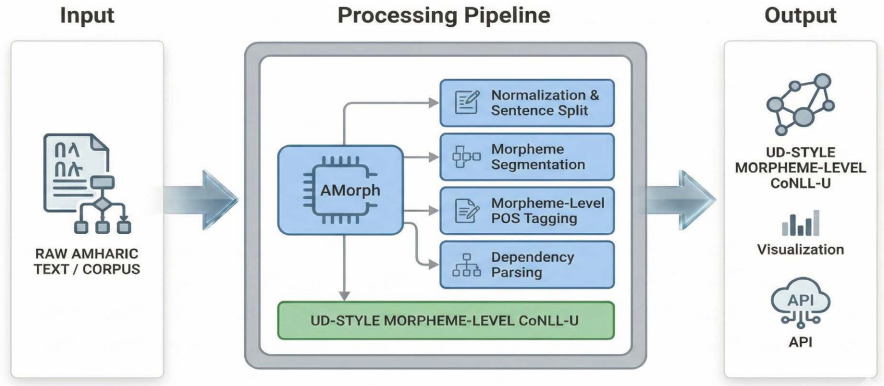


Figure 1. Overview of our neural Amharic NLP pipeline. Our system takes raw Amharic text as input, and produces UD-style morpheme-level CoNLL-U annotations. Besides this neural pipeline, we also provide an easy-to-use API and a visualization interface.

- Silver-to-gold data pipeline. An end-to-end workflow that generates candidate segmentations from HornMorpho [9], applies automated consistency checks and LLM-assisted review, and produces human-adjudicated UD-style morpheme-level annotations.
- CTC-based character segmenter. An XLM-RoBERTa segmenter trained as a CTC transducer to predict morpheme boundaries without explicit character-morpheme alignments, improving robustness beyond rule-based output.
- Unified multi-task morphosyntactic model. A shared-encoder architecture jointly trained for morpheme-level POS tagging and dependency parsing, with biaffine arc/label scoring and projective decoding [15], improving parameter efficiency and cross-task generalization.
- Open-source toolkit and visualization. A practical implementation exposing the full pipeline via simple APIs and inspection/visualization utilities to support real-world Amharic NLP development.

On a held-out, human-validated test set, our system achieves strong morpheme segmentation quality alongside accurate POS tagging and dependency parsing, indicating that analyzer-bootstrapped supervision combined with multilingual pretraining can enable effective Amharic morphosyntactic analysis under limited gold annotation.

2. Related Work

2.1. Multilingual NLP Pipelines and Toolkits

NLP toolkits have lowered the barrier to deploying tokenization, tagging, and

parsing in real applications, but their coverage and performance historically concentrate on high-resource languages. Classic pipeline systems such as Stanford CoreNLP provide a broad suite of annotators in a unified architecture, enabling end-to-end processing for widely studied languages and domains [1]. UD-centric toolkits such as UDPipe operationalize the CoNLL-U ecosystem by offering trainable models that map raw text to UD-style tokenization, morphology, and dependencies across many treebanks [4]. Neural-first toolkits including Stanza and Trankit further demonstrate that language-agnostic architectures trained on UD can deliver strong multilingual performance and convenient Python APIs [5] [16], while frameworks such as Flair and spaCy emphasize usability and strong baselines for common NLP tasks, accelerating research and production workflows [2] [3]. Existing toolkits show that unified pipelines are feasible and impactful, but low-resource languages still often lack strong, ready-to-use models and data aligned to modern standards.

2.2. Universal Dependencies and Amharic Resources

Universal Dependencies (UD) provides a cross-linguistically consistent annotation framework for word segmentation, morphology, and dependency syntax, and has become a de facto standard for evaluating multilingual pipelines [6] [7] [17]. Critically, UD defines annotation units as syntactic words and generally avoids full morpheme segmentation; tokenization guidelines specify when to split orthographic forms (e.g. clitics and fused tokens) and how to represent such splits using multiword tokens [18]. For Amharic, UD Amharic-ATT offers manually curated POS, morphology, and dependency annotations, explicitly noting manual segmentation of clitics for this morphologically rich language [19]. The underlying treebank creation effort establishes practical conventions for representing Amharic morphology and syntax within UD, including decisions about clitics and orthographic token boundaries [20]. UD supplies the evaluation and interoperability target, but Amharic resources remain limited in scale, and bridging morphology-heavy analyses to UD tokenization requires careful, language-specific design.

2.3. Morphological Segmentation in Low-Resource Settings

Morphological segmentation has long been approached through both unsupervised and supervised paradigms. Unsupervised methods such as Morfessor learn morpheme-like units from raw text using probabilistic/MDL principles, offering a data-light baseline that is appealing in low-resource contexts [21]. In morphologically rich languages, segmentation quality can substantially affect downstream tagging and parsing, motivating task-driven segmenters and tokenization modules tailored to specific languages and orthographies. For example, Arabic segmentation systems such as Farasa show that language-dependent segmentation can be addressed efficiently with supervised ranking-based approaches when appropriate training data and conventions are available [22]. More broadly, charac-

ter-level sequence learning objectives such as Connectionist Temporal Classification (CTC) provide a way to predict structured outputs without requiring explicit alignment, making them attractive for noisy supervision and silver-data regimes [14]. Prior work suggests that segmentation is both feasible and beneficial, but high accuracy typically depends on clear conventions and training signal—both of which are scarce for Amharic at the morpheme level.

2.4. Neural Morphosyntax and Multitask Learning with UD

A major trend in multilingual morphosyntax is to share representations across tasks and languages. UDify demonstrates that a single multilingual model can jointly predict UD annotations (POS, features, lemmas, dependencies) across dozens of languages by fine-tuning a shared pretrained encoder with lightweight task heads, and that low-resource languages often benefit most from such sharing [23]. For dependency parsing, biaffine scoring architectures remain a strong and widely adopted choice, providing accurate arc and label predictions with efficient inference [24]. These results motivate multitask designs for under-resourced languages: shared encoders can amortize supervision across correlated tasks (e.g. tagging and parsing) and improve parameter efficiency compared to training separate models per component. Multitask UD models and biaffine parsers provide a strong blueprint for building compact, accurate morphosyntactic analyzers when labeled data is limited.

2.5. Rule-Based Morphology and HornMorpho

Finite-state morphology is a well-established approach to encoding rich morphological systems with explicit linguistic constraints, especially for languages with complex inflectional patterns [25]. HornMorpho is a rule-based morphological analyzer and generator targeting Horn-of-Africa languages (including Amharic, Oromo, and Tigrinya), and it is distributed as a Python program within the broader L3 project ecosystem [9]. As a rule-based system, HornMorpho can provide high-precision analyses for in-vocabulary forms and encode linguistically motivated structure, but it also inherits common limitations of handcrafted analyzers; sensitivity to spelling variation and domain shift, ambiguity that requires disambiguation, and a representation that does not directly match downstream annotation schemes without additional conversion logic [9] [15]. Recent shared-task [26] and benchmarking work in morphology notes HornMorpho as an available analyzer/generator resource and characterizes it as implementing a weighted finite-state approach enriched with feature-structure information, underscoring its continued value as a linguistic prior and data source. HornMorpho offers strong linguistic structure for Amharic analysis, but using it in modern UD-style pipelines requires explicit alignment steps and data-driven components to handle ambiguity and robustness gaps.

Taken together, prior toolkits and UD-based parsers establish effective architectural patterns for multilingual pipelines [4] [5] [15], while finite-state analyzers

such as HornMorpho provide a practical source of linguistic knowledge and weak supervision in low-resource languages [9]. However, existing work does not provide an end-to-end recipe that 1) bootstraps morpheme-level segmentation from a rule-based analyzer, 2) reconciles the output with UD segmentation and tagging conventions, and 3) trains a unified, multitask morphosyntactic model for Amharic in an open-source toolkit. The literature motivates combining UD-first neural pipelines with rule-based morphology, but an Amharic-specific integration at the morpheme level remains underdeveloped.

3. Methodology

We implement an end-to-end Amharic analysis stack that combines an LLM-assisted adjudication workflow for UD-consistent morpheme supervision with two learned modules as shown in **Figure 2**: 1) a token-level morpheme segmenter trained as a weighted CTC character transducer on top of XLM-R [27], and 2) a multi-task morpho-syntactic analyzer with an XLM-R backbone and task-specific heads for morpheme-level XPOS tagging and dependency parsing. The overall system maps raw Amharic text to UD-style morpheme-level CoNLL-U structures.

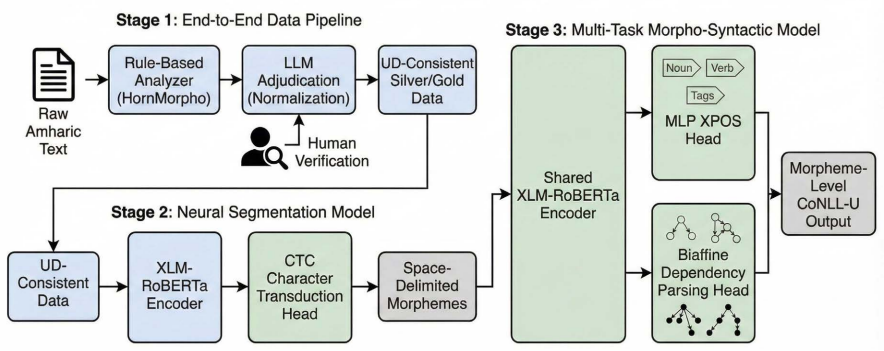


Figure 2. Overview of the proposed training and inference pipeline. The system first constructs UD-consistent morpheme supervision via a HornMorpho → LLM → human workflow, then trains a CTC-based neural segmenter to produce space-delimited morphemes, and finally applies a shared-encoder multi-task model with XPOS and dependency heads to output morpheme-level CoNLL-U annotations.

3.1. LLM-Assisted Adjudication of HornMorpho Segmentations

We use an LLM as a decision-support tool to accelerate human adjudication of HornMorpho candidate segmentations and enforce UD-style consistency. We use OpenAI's GPT-4o with low-temperature decoding to promote stable, schema-faithful outputs.

For each target token, HornMorpho produces one or more candidate segmentations (and associated morphosyntactic information). The LLM is given sentence context, the surface token, the HornMorpho candidates, and a concise description of UD Amharic-ATT segmentation conventions, plus a small set of manually curated examples covering clitics, function morphemes, derivation, and common boundary ambiguities. The model is instructed to either accept a candidate un-

changed or return a corrected UD-consistent segmentation and tag sequence together with a brief justification. LLM output is never treated as gold directly. Annotators review the HornMorpho analysis and the LLM suggestion side-by-side, and either accept HornMorpho, accept the LLM revision, or manually override both. Two native Amharic speaking annotators with linguistics training performed verification, with a third senior annotator adjudicating disagreements; agreement was measured on a double-annotated subset using boundary F1 and exact-match. The adjudicated result is stored as the final UD-style morpheme-level annotation used to train our segmenter and downstream models.

3.2. Problem Setup and Notation

Let a sentence be a sequence of surface (orthographic) tokens

$$X = (x_1, \cdots, x_n).$$

Each surface token x_i corresponds to a sequence of morphemes under our UD-style morpheme tokenization

$$S_i = (s_{i,1}, \cdots, s_{i,m_i}),$$

and the segmented sentence is the flattened morpheme-token sequence

$$S = (s_1, \cdots, s_T), \ T = \sum_{i=1}^{n} m_i.$$

For each morpheme token s_t, the morpho-syntactic analyzer predicts an XPOS tag $y_t \in \mathcal{Y}$ and dependency structure (h_t, r_t) where $h_t \in \{0,1,\cdots,T\}$ is the head index (with 0 denoting ROOT) and $r_t \in \mathcal{R}$ is the dependency relation.

3.3. Weighted CTC Morpheme Segmenter

3.3.1. Target Encoding

Given a surface token x, the gold segmentation is a morpheme sequence

$$S^*(x) = (s_1, \cdots, s_m).$$

We encode the target as a canonical character string by inserting explicit morpheme boundary markers as spaces:

$$\mathbf{y} = "s_1 \ s_2 \ \cdots \ s_m".$$

The output alphabet is

$$\Sigma = \{\varnothing\} \cup \{\text{all characters observed in targets, including space}\},$$

where $\varnothing$ is the CTC blank. In our implementation, blank has ID 0 and unknown has ID 1.

3.3.2. Time-Step Expansion by Character Repetition

CTC requires sufficient input time steps to emit targets that can be longer than the surface token (due to inserted spaces). Let the surface token be a character sequence $x = (c_1, \cdots, c_p)$ We repeat each character by a fixed factor $k \geq 1$:

$$\tilde{x} = \left(\underbrace{c_1, \cdots, c_1}_{k}, \underbrace{c_2, \cdots, c_2}_{k}, \cdots, \underbrace{c_p, \cdots, c_p}_{k} \right).$$

This repetition is implemented explicitly (default $k = 3$), increasing the available CTC time steps.

3.3.3. Encoder and Character-Level State Selection

We feed $\tilde{x}$ to an XLM-R encoder in "split-into-words", which refers to the tokenizer interface where we pass a pre-split list of units rather than a raw string. In our case, the units are individual repeated characters of a single surface token, so we are not relying on Amharic word boundaries or whitespace tokenization. Word segmentation is handled upstream at the orthographic-token level, and this module only predicts morpheme boundaries within each token. mode, treating each repeated character as a separate word unit. The encoder produces contextual states

$$H = \mathrm{Enc}_\theta(\tilde{x}) \in \mathbb{R}^{L \times d}.$$

Because repeated characters can map to multiple subword pieces, we construct a mapping using tokenizer and select a single representation per repeated-character group (the first subword position for each group), yielding indices char positions Table 1. Let

$$U = |\mathrm{char\_positions}|$$

and define the selected character-level states

$$H' = \mathrm{Select}(H; \mathrm{char\_positions}) \in \mathbb{R}^{U \times d}.$$

Table 1. Algorithm A: Training the weighted CTC morpheme segmenter.

Step	Operation	Input → Output
A1	Character repetition (factor k)	$x = (c_1, \cdots, c_p) \to \tilde{x}$
A2	XLM-R encoding	$\tilde{x} \to H \in \mathbb{R}^{L \times d}$
A3	Select char states via	$H \to H' \in \mathbb{R}^{U \times d}$
A4	Linear emission projection	$H' \to E \in \mathbb{R}^{U \times \lvert\Sigma\rvert}$
A5	CTC loss (per example)	$(E, \mathbf{y}) \to \mathcal{L}_{\mathrm{ctc}}(x, \mathbf{y})$
A6	Weighted batch objective	$\{w_i, \mathcal{L}_i\} \to \mathcal{L}_{\mathrm{seg}} = \frac{\sum w_i \mathcal{L}_i}{\sum w_i + \epsilon}$
A7	Filter invalid instances	discard if $\lvert\mathbf{y}\rvert > U$

3.3.4. CTC Emissions and Likelihood

A linear projection produces emission logits

$$E = WH' + b \in \mathbb{R}^{U \times |\Sigma|}, \quad P_t(\cdot \mid x) = \mathrm{softmax}(E_t).$$

CTC marginalizes over all alignment paths $\pi \in \Sigma^U$ that collapse to the target string $\mathbf{y}$ under the CTC collapse mapping $\mathcal{B}(\cdot)$:

$$p(\mathbf{y} \mid x) = \sum_{\pi:\mathcal{B}(\pi)=\mathbf{y}} \prod_{t=1}^{U} P_t(\pi_t \mid x), \quad \mathcal{L}_{\text{ctc}}(x, \mathbf{y}) = -\log p(\mathbf{y} \mid x). \tag{1}$$

3.3.5. Per-Example Weighted Objective

Each training instance carries a non-negative weight $w \geq 0$ reflecting annotation source/quality. For a mini-batch $\mathcal{B}$ we compute a weighted mean of per-example CTC losses:

$$\mathcal{L}_{\text{seg}}(\mathcal{B}) = \frac{\sum_{(x,\mathbf{y},w)\in\mathcal{B}} w\mathcal{L}_{\text{ctc}}(x, \mathbf{y})}{\sum_{(x,\mathbf{y},w)\in\mathcal{B}} w + \epsilon}, \tag{2}$$

with $\epsilon > 0$ for numerical stability (implemented by clamping the denominator to $\geq 10^{-6}$).

3.3.6. Target-Length Filtering

CTC is undefined when the encoded target length exceeds the number of input steps. The collate procedure filters any instance where

$$|\mathbf{y}| > U$$

to ensure valid loss computation.

3.3.7. Greedy Decoding

At inference, we perform greedy CTC decoding [28]:

$$\hat{\pi}_t = \arg\max_{a\in\Sigma} P_t(a \mid x), \tag{3}$$

followed by CTC collapse (merge duplicates, remove blanks) to form $\hat{\mathbf{y}}$. We normalize whitespace and split on spaces to produce the predicted morpheme sequence $\hat{S}(x)$.

3.4. Multi-Task Morpho-Syntactic Analyzer (XPOS + DEP)

3.4.1. Backbone and Morpheme-Level Representations

Given a segmented morpheme-token sequence $S = (s_1, \cdots, s_T)$, we tokenize it into subwords $(u_1, \cdots, u_M)$ and encode with XLM-R:

$$H = f_\theta(u_{1:M}) \in \mathbb{R}^{M\times d}.$$

Tasks are defined at the morpheme level, so we align each morpheme token s_t to a designated subword index $\pi(t) \in \{1, \cdots, M\}$ and gather morpheme vectors:

$$g_t = H_{\pi(t)} \in \mathbb{R}^d, \quad t = 1, \cdots, T.$$

For dependency parsing, we additionally include a ROOT representation at index 0 derived from the encoder's special token vector (CLS) and treat tokens as indexed $0, \cdots, T$.

3.4.2. XPOS Head

The XPOS head is an MLP token classifier producing logits $\ell_t^{\text{pos}} \in \mathbb{R}^{|\mathcal{Y}|}$:

$$p(y_t \mid S) = \text{softmax}(\ell_t^{\text{pos}}).$$

With mask $m_t \in \{0,1\}$ indicating non-padding morphemes, the loss is masked

cross-entropy:

$$\mathcal{L}_{\text{pos}} = \sum_{t=1}^{T} m_t \, \text{CE}\left(\ell_t^{\text{pos}}, y_t\right).$$

(while the framework supports CRF decoding, our Amharic XPOS configuration uses an MLP classifier.)

3.4.3. Dependency Head: Biaffine Graph Parser

We use a graph-based dependency parser that scores all head-dependent arcs. Let indices range over $0,\cdots,T$ with 0 as ROOT (Table 2). Two MLP projections produce head and dependent representations:

Table 2. Algorithm B: Multi-task morpho-syntactic analysis with shared XLM-R encoder, XPOS head, and biaffne dependency parser with Eisner decoding.

Step	Operation	Input → Output
C1	Subword tokenize morphemes	$S=(s_1,\cdots,s_T)\to(u_1,\cdots,u_M)$
C2	XLM-R encoding	$u_{1:M}\to H\in\mathbb{R}^{M\times d}$
C3	Gather morpheme vectors	$H,\pi(t)\to g_t=H_{\pi(t)}$ for $t=1,\cdots,T$
C4	XPOS head (MLP classifier)	$g_{1:T}\to\ell_t^{\text{pos}},\mathcal{L}_{\text{pos}}$
C5	DEP head (biaffine arc/rel)	$g_{0:T}\to A_{i\to j},R_{i\to j},\mathcal{L}_{\text{dep}}$
C6	Step-wise task sampling	sample $t\sim q(t)\propto\lvert\mathcal{D}_t\rvert^{\tau}$
C7	One-step update	backprop only $\mathcal{L}_t$ (shared encoder + task head)
C8	Eisner decoding (inference)	$A\to\hat{h}\in\mathcal{T}_{\text{proj}},\hat{r}_j=\arg\max_r R_{\hat{h}_j\to j}[r]$

$$\mathbf{h}_i = f_{\text{head}}\left(g_i\right),\ \mathbf{d}_j = f_{\text{dep}}\left(g_j\right).$$

Arc scores form a matrix $A\in\mathbb{R}^{(T+1)\times(T+1)}$ via a biaffine scorer:

$$A_{i\to j} = \text{Biaffine}_{\phi_a}\left(\mathbf{h}_i,\mathbf{d}_j\right).$$

Relation logits are similarly computed for each arc:

$$R_{i\to j}\in\mathbb{R}^{|\mathcal{R}|},\ R_{i\to j} = \text{Biaffine}_{\phi_r}\left(\mathbf{h}_i,\mathbf{d}_j\right). \tag{4}$$

Given gold heads h_j and relations r_j, we optimize an interpolated loss:

$$\mathcal{L}_{\text{arc}} = \sum_{j=1}^{T} m_j \, \text{CE}\left(A_{\cdot\to j},h_j\right),\ \mathcal{L}_{\text{rel}} = \sum_{j=1}^{T} m_j \, \text{CE}\left(R_{h_j\to j},r_j\right),$$

$$\mathcal{L}_{\text{dep}} = (1-\alpha)\mathcal{L}_{\text{arc}} + \alpha\mathcal{L}_{\text{rel}}, \tag{5}$$

with $\alpha\in(0,1)$ a fixed interpolation coefficient.

3.4.4. Tree Constraint and Eisner Decoding

At inference, independent head selection can violate tree well-formedness. We therefore decode a valid projective tree using Eisner decoding on arc scores:

$$\hat{h} = \arg\max_{h\in\mathcal{T}_{\text{proj}}} \sum_{j=1}^{T} A_{h_j\to j}, \tag{6}$$

where $\mathcal{T}_{\text{proj}}$ is the set of projective dependency trees. Relations are then decoded conditionally:

$$\hat{r}_j = \arg\max_{r \in \mathcal{R}} R_{\hat{h}_j \to j}[r].$$

3.4.5. Multi-Task Optimization

Multi-task training uses step-wise task sampling. Let tasks be $\mathcal{T}$ and each task have dataset $\mathcal{D}_t$. A task is sampled with

$$q(t) \propto |\mathcal{D}_t|^{\tau},$$

where τ is a temperature parameter. At each step, only the sampled task's loss is computed and backpropagated, updating both the shared encoder and the active task head:

$$\min \mathbb{E}_{t \sim q(t)} \mathbb{E}_{\mathcal{B} \sim \mathcal{D}_t} \left[\mathcal{L}_t(\mathcal{B}) \right].$$

4. Evaluation Metrics

4.1. Segmentation Metrics

Let $S^*(x)$ and $\hat{S}(x)$ be the gold and predicted morpheme sequences for token x.

Exact match (EM).

$$\text{EM}(x) = \mathbf{1}\left[\hat{S}(x) = S^*(x)\right], \ \text{EM} = \frac{1}{N}\sum_{i=1}^{N} \text{EM}(x_i).$$

and boundary-level Precision/Recall/F1, following standard definitions.

4.2. XPOS Tagging Metric

For each morpheme token t with mask $m_t = 1$, token accuracy is

$$\text{Acc}_{\text{pos}} = \frac{\sum_{\text{sent}} \sum_{t=1}^{T} m_t \mathbf{1}[\hat{y}_t = y_t]}{\sum_{\text{sent}} \sum_{t=1}^{T} m_t}.$$

4.3. Dependency Parsing Metrics

Let gold heads/relations be (h_t, r_t) and predictions be $(\hat{h}_t, \hat{r}_t)$ for $t = 1, \cdots, T$.

Labeled Attachment Score (LAS).

$$\text{LAS} = \frac{\sum_{\text{sent}} \sum_{t=1}^{T} m_t \mathbf{1}\left[\hat{h}_t = h_t \wedge \hat{r}_t = r_t\right]}{\sum_{\text{sent}} \sum_{t=1}^{T} m_t}.$$

Projectivity note.

Because decoding uses Eisner, predicted trees are projective. If gold trees contain non-projective arcs, this constraint can limit the achievable LAS; if the gold data is predominantly projective, Eisner provides a strong structural prior.

5. Experiment and Results

This section describes the experimental protocol for 1) morpheme segmentation

and 2) morpheme-level XPOS tagging and dependency parsing.

5.1. Data

5.1.1. Segmentation Data

We train and evaluate the segmenter on a token-level dataset where each instance consists of a surface token x, a gold morpheme sequence $S(x)=(s_1,\cdots,s_m)$, and metadata recording provenance and confidence. We split the dataset into disjoint train/dev/test partitions containing 7783/2267/2034 tokens, respectively.

Canonical target representation.

For CTC training, we map the gold morpheme sequence to a canonical character string by inserting explicit boundary markers as spaces:

$$\mathbf{y}(x)="s_1\ s_2\ \cdots\ s_m".$$

This representation makes morpheme boundaries directly recoverable as whitespace positions in the decoded string.

5.1.2. Morpheme-Level POS and Dependency Data

For XPOS tagging and dependency parsing, we use UD-style sentence-level annotations in CoNLL-U format at the morpheme token level. The resulting partitions contain 4183/512/512 sentences with 70,007/8443/9013 morpheme tokens for train/dev/test, respectively. The XPOS and dependency-relation vocabularies are fixed during training and evaluation, with $|\mathcal{Y}|=63$ XPOS labels and $|\mathcal{R}|=54$ dependency relations.

5.1.3. Weighting for Segmentation Supervision

Each segmentation instance is assigned a non-negative weight $w\geq 0$ that reflects its provenance and reliability. Concretely, we define

$$w=w_{\text{src}}\cdot c\cdot\gamma_{\text{MWT}}\cdot\gamma_{\text{ambig}},$$

where w_{src} is a source-dependent base weight, $c\in[0,1]$ is the confidence score, $\gamma_{\text{MWT}}>1$ up weights multi-morpheme tokens, and $\gamma_{\text{ambig}}<1$ down weights tokens for which the analyzer yields multiple competing analyses. This weighting scheme encourages the model to prioritize higher-confidence, human/LLM-verified supervision while still leveraging large-scale silver data.

5.2. Preprocessing

We canonicalize morpheme segmentations by joining morphemes with single spaces and normalizing whitespace. A harmonization step resolves conflicts among multiple candidate analyses using a deterministic priority rule and filters structurally invalid candidates.

For segmentation, the input is a character sequence derived from each surface token and tokenized with the XLM-R tokenizer in split-into-words mode. For XPOS/DEP, morpheme tokens are subword-tokenized with XLM-R, and we compute an alignment map $\pi(t)$ that selects one subword position per morpheme for morpheme-level prediction.

For XPOS/DEP, we use a maximum input length of 512 subword tokens with truncation and discard examples that overflow this budget. For segmentation, we cap the repeated-character representation to 256 subword positions.

5.3. Models

5.3.1. Segmentation Model (XLM-R + Weighted CTC)

The segmenter is an encoder-only XLM-R model followed by a linear projection to a character vocabulary Σ (including whitespace as a boundary symbol and a CTC blank). To ensure sufficient time steps for CTC, we repeat each input character by a fixed factor ($k = 3$). At inference, we apply greedy CTC decoding (argmax per time step, collapse repeats, remove blanks) and recover morphemes by splitting the decoded canonical string on spaces.

5.3.2. Multi-Task Model (XPOS + Dependency Parsing)

The morpho-syntactic analyzer uses an XLM-RoBERTa Large backbone (24 layers, hidden size 1024) with task-specific heads. The XPOS head is an MLP classifier over 63 labels. The dependency head is a biaffine graph-based parser over 54 relation labels, using separate MLP projections for head and dependent representations. At inference, we apply Eisner decoding to enforce a well-formed projective dependency tree.

5.4. Training

5.4.1. Segmentation Training

We optimize the segmenter with AdamW using a learning rate of 2×10^{-5} and weight decay of 0.01, with a linear learning-rate schedule and 10% warmup. We train for up to 8 epochs with batch size 8, gradient accumulation of 2 steps, and global-norm gradient clipping at 1.0.

5.4.2. Multi-Task XPOS/DEP Training

We train the multi-task analyzer with AdamW using a learning rate of 1×10^{-5} and no weight decay, with a linear schedule and 2% warmup. We apply layer-wise learning-rate decay (factor 0.9) across the 24 transformer layers and train for 8 epochs. Batch sizes are 16 for XPOS and 8 for dependencies. Multi-task mixing uses step-wise task sampling with temperature $\tau = 0.8$.

5.5. Result

Table 3 summarizes held-out test performance for token-level morpheme segmentation. The proposed XLM-R + CTC segmenter substantially outperforms both a rule-based analyzer (HornMorpho) and an unsupervised baseline (Morfessor), achieving 73.65 exact-match accuracy, 82.88 segment F_1, and 88.86 boundary F_1. Relative to HornMorpho, this corresponds to +30.43 exact-match points, +19.64 segment-F_1 points, and +18.36 boundary-F_1 points, while reducing average edit distance from 1.837 to 0.533 (71% lower). To put these gains in context, we note that we do not report a separate supervised neural segmentation baseline, for

the following reasons, we did not include an additional supervised neural baseline (e.g., a standard BERT/XLM-R sequence tagger) because our training signal is largely analyzer-bootstrapped and does not provide clean character-level boundary labels without extra alignment heuristics. In practice, these heuristics become a major confounding factor and make it hard to attribute gains to the model architecture itself. For this reason, we compare our CTC segmenter against the rule-based analyzer and an unsupervised baseline, and leave a controlled comparison with alternative neural segmenters under fully human-labeled supervision to future work.

Table 3. Morpheme segmentation results on the held-out test set. Scores are reported as percentages, except average edit distance (lower is better).

System	Exact Match	Segment F_1	Boundary F_1	Avg. Edit Dist.
XLM-R + CTC (ours)	**73.65**	**82.88**	**88.86**	**0.533**
HornMorpho (rule-based)	43.22	63.24	70.50	1.837
Morfessor (unsupervised)	14.11	24.30	37.98	2.020

Training dynamics are consistent with these gains. Figure 3 shows that boundary-level performance improves rapidly in test-set performance on morpheme segmentation and morpheme-level morphosyntactic analysis. All values are reported as percentages.

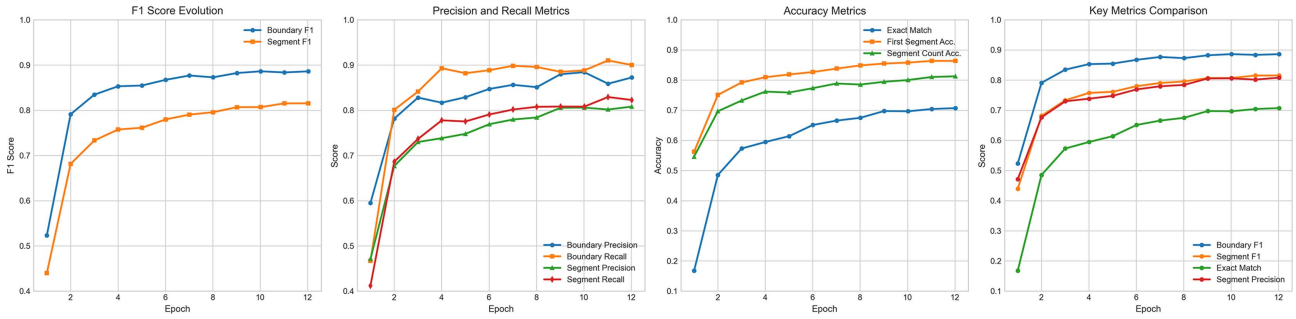


Figure 3. Learning curves for the weighted CTC morpheme segmenter on the development set: segmentation precision/recall/F1, boundary precision/recall/F1, exact match, first-segment accuracy.

Early epochs and then plateaus, while segment-level F_1 and exact match continue to improve more gradually, reflecting that boundary placement stabilizes earlier than full-sequence correctness. Figure 4 provides additional evidence from training dynamics. POS accuracy rises quickly and remains stable on validation, whereas dependency LAS improves more slowly across epochs, reflecting the higher structural ambiguity and label sparsity of morpheme-level parsing.

We evaluate the morpho-syntactic analyzer at the morpheme level using XPOS accuracy and labeled attachment score (LAS). As reported in Table 4, the multi-task model reaches 87.06% morpheme-level POS accuracy and 64.97 LAS on the held-out test set. These results indicate that multilingual pretraining transfers ef-

fectively to morpheme-level POS tagging, and that with the proposed supervision pipeline and multi-task training labeled dependency parsing is also competitive, though it remains the more challenging component in this setting.

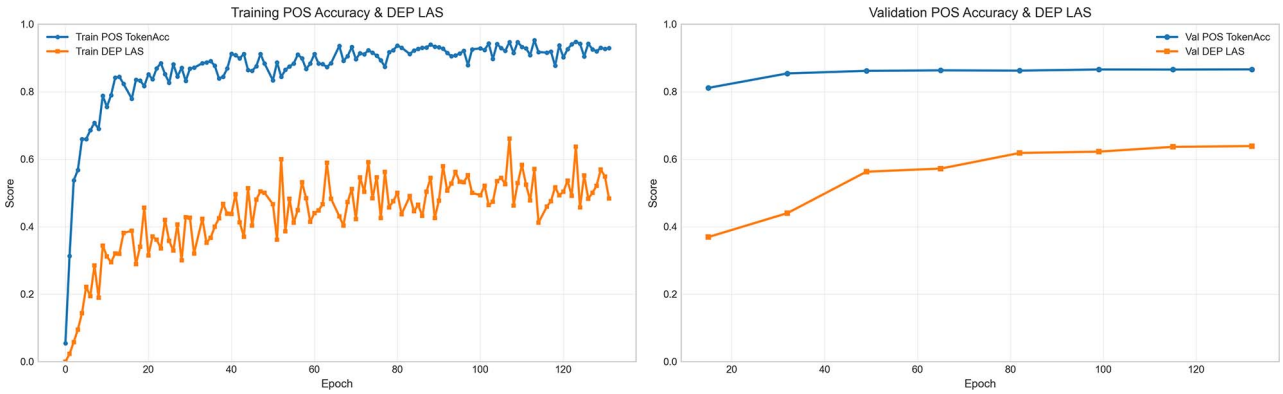


Figure 4. Learning curves for the multi-task morpho-syntactic analyzer POS token accuracy and dependency LAS over training (left) and validation (right) across epochs.

Table 4. Test-set performance on morpheme segmentation and morpheme-level morpho-syntactic analysis. All values are reported as percentages.

Segmentation			
Model	Exact Match	Boundary F_1	Segment F_1
XLM-R + CTC	73.65	88.86	82.88
Morpheme-level morphosyntax			
Model	XPOS Accuracy	LAS	
AMO (multi-task)	87.06	64.97	

Figure 5 further shows a sharp drop in weighted CTC loss during the first epochs followed by steady convergence, indicating that the character-transduction objective is learnable and stable under the proposed weighting and filtering. **Figure 6** shows that POS loss decreases faster and to a lower floor than dependency loss, while dependency optimization remains noisier throughout training, matching the observed gap between POS and LAS.

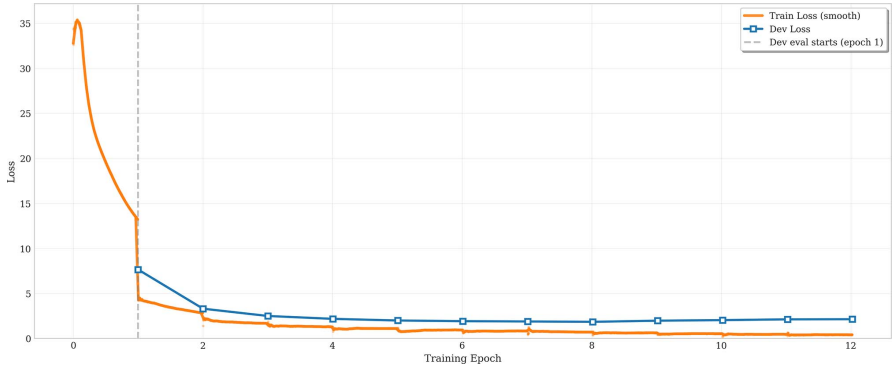


Figure 5. Weighted CTC morpheme segmenter loss over training and development steps.

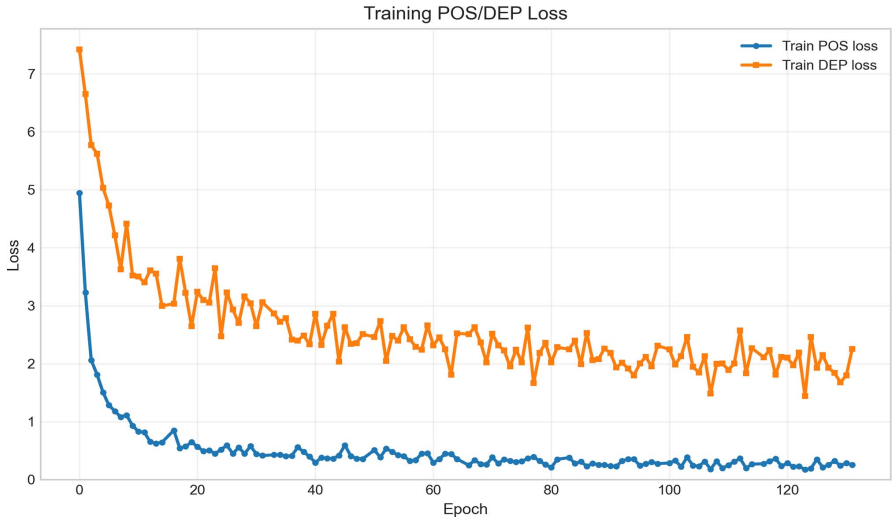


Figure 6. Training dynamics of the multi-task morpho-syntactic analyzer.

6. Conclusions

We presented an end-to-end Amharic processing stack that maps raw text to UD-style morpheme-level CoNLL-U analyses by integrating a silver-to-gold supervision pipeline bootstrapped from a rule-based analyzer and refined via LLM-assisted, human-adjudicated correction, a weighted CTC character-transduction segmenter built on multilingual pretraining, and a unified multi-task morpho-syntactic analyzer that jointly performs morpheme-level XPOS tagging and dependency parsing with a shared Transformer encoder and a biaffine parsing head with projective decoding. This design directly targets practical bottlenecks in low-resource, morphology-rich NLP scarce consistent supervision, analyzer UD convention mismatch, and the inefficiency of maintaining separate models per task.

Empirically, our results show that the proposed segmenter produces accurate UD-consistent morpheme boundaries and that the multi-task analyzer yields strong morpheme-level POS tagging while improving labeled dependency parsing relative to prior tool support for Amharic. These findings support the conclusion that analyzer-bootstrapped supervision, combined with multilingual pretrained encoders and human verification, is an effective strategy for building practical morphosyntactic analyzers under limited gold annotation. Beyond model accuracy, we release a usable inference API and a visualization interface to support deployment, inspection of intermediate outputs, and iterative dataset refinement.

Future work will scale and diversify supervision (additional domains and larger human-validated subsets), strengthen ambiguity handling, evaluate fully end-to-end performance from raw text through morphosyntactic outputs, and relax the projectivity constraint via non-projective decoding when beneficial. We hope this work provides both an effective recipe for building morpheme-aware pipelines in low-resource settings and a practical open-source foundation for Amharic NLP.

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. and McClosky, D. (2014) The Stanford Corenlp Natural Language Processing Toolkit. *Proceedings of* 52*nd Annual Meeting of the Association for Computational Linguistics*: *System Demonstrations*, Baltimore, 23-24 June 2014, 55-60. https://doi.org/10.3115/v1/p14-5010

[2] Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S. and Vollgraf, R. (2019) FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP. *Proceedings of the* 2019 *Conference of the North*, Minneapolis, 2 June-7 June 2019, 54-59. https://doi.org/10.18653/v1/n19-4010

[3] Honnibal, M., Montani, I., Van Landeghem, S. and Boyd, A. (2020) Spacy: Industrial-Strength Natural Language Processing in Python. Zenodo.

[4] Straka, M., Hajič, J. and Straková, J. (2016) UDPipe: Trainable Pipeline for Processing CoNLL-U Files Performing Tokenization, Morphological Analysis, POS Tagging and Parsing. *Proceedings of LREC* 2016, Portorož, May 2016, 4290-4297. https://aclanthology.org/L16-1680/

[5] Qi, P., Zhang, Y., Zhang, Y., Bolton, J. and Manning, C.D. (2020) Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. *Proceedings of the* 58*th Annual Meeting of the Association for Computational Linguistics*: *System Demonstrations*, Online, 5 July-10 July 2020, 101-108. https://doi.org/10.18653/v1/2020.acl-demos.14

[6] Nivre, J., de Marneffe, M.-C., Ginter, F., Hajič, J., Manning, C.D., Pyysalo, S., Schuster, S., Tyers, F. and Zeman, D. (2020) Universal Dependencies v2: An Ever-Growing Multilingual Treebank Collection. *Proceedings of LREC* 2020, Marseille, 11–16 May 2020, 4034-4043. https://aclanthology.org/2020.lrec-1.497/

[7] de Marneffe, M., Manning, C.D., Nivre, J. and Zeman, D. (2021) Universal Dependencies. *Computational Linguistics*, **47**, 255-308.

[8] Gezmu, A.M., Seyoum, B.E., Gasser, M. and Nürnberger, A. (2018) Contemporary Amharic Corpus: Automatically Morpho-Syntactically Tagged Amharic Corpus. *Proceedings of the First Workshop on Linguistic Resources for Natural Language Processing*, Santa Fe, 20 August 2018, 65-70. https://aclanthology.org/W18-3809/

[9] Gasser, M. (2011) HornMorpho: A System for Morphological Processing of Amharic, Oromo, and Tigrinya. *Proceedings of HLT4Dev* 2011, Alexandria, 2-3 May 2011, 94-99.

[10] Caruana, R. (1997) Multitask Learning. *Machine Learning*, **28**, 41-75. https://doi.org/10.1023/a:1007379606734

[11] Ruder, S. (2017) An Overview of Multi-Task Learning in Deep Neural Networks. arXiv:1706.05098. https://arxiv.org/abs/1706.05098

[12] Seyoum, B.E., Miyao, Y., Mekonnen, B. and Yimam, B. (2016) Morpho-Syntactically Annotated Amharic Treebank. Corpus Linguistics Fest (CLiF) 2016. https://ceur-ws.org/Vol-1607/seyoum.pdf

[13] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., *et al.* (2020) Unsupervised Cross-Lingual Representation Learning at Scale. *Proceedings of the* 58*th Annual Meeting of the Association for Computational Linguistics*, Online, 5-10 July 2020, 8440-8451. https://doi.org/10.18653/v1/2020.acl-main.747

[14] Graves, A., Fernández, S., Gomez, F. and Schmidhuber, J. (2006) Connectionist Temporal Classification. *Proceedings of the* 23*rd International Conference on Machine Learning-ICML* '06, New York, 25-29 June, 2006, 369-376. https://doi.org/10.1145/1143844.1143891

[15] Eisner, J.M. (1996) Three New Probabilistic Models for Dependency Parsing. *Proceedings of the* 16*th Conference on Computational Linguistics*, Stroudsburg, 5-9 August 1996, 340-345. https://doi.org/10.3115/992628.992688

[16] Nguyen, M.V., Lai, V.D., Pouran Ben Veyseh, A. and Nguyen, T.H. (2021) Trankit: A Light-Weight Transformer-Based Toolkit for Multilingual Natural Language Processing. *Proceedings of the* 16*th Conference of the European Chapter of the Association for Computational Linguistics*: *System Demonstrations*, Online, 19-23 April 2021, 80-90. https://doi.org/10.18653/v1/2021.eacl-demos.10

[17] Zeman, D., Hajič, J., Popel, M., Potthast, M., Straka, M., Ginter, F., *et al.* (2018) CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. *Proceedings of the CoNLL* 2018 *Shared Task*, Brussels, 31 October-1 November 2018, 1-21. https://doi.org/10.18653/v1/k18-2001

[18] Universal Dependencies (2025) Tokenization and Word Segmentation. Online Documentation. https://universaldependencies.org/u/overview/tokenization.html

[19] Universal Dependencies (2025) UD Amharic-ATT Treebank. Online Documentation. https://universaldependencies.org/treebanks/am_att/

[20] Seyoum, B.E., Miyao, Y. and Mekonnen, B.Y. (2018) Universal Dependencies for Amharic. *Proceedings of LREC* 2018, Miyazaki, 7-12 May 2018, 2216-2222. https://aclanthology.org/L18-1350/

[21] Creutz, M. and Lagus, K. (2007) Unsupervised Models for Morpheme Segmentation and Morphology Learning. *ACM Transactions on Speech and Language Processing*, **4**, 3-es. https://doi.org/10.1145/1217098.1217101

[22] Abdelali, A., Darwish, K., Durrani, N. and Mubarak, H. (2016) Farasa: A Fast and Furious Segmenter for Arabic. *Proceedings of the* 2016 *Conference of the North American Chapter of the Association for Computational Linguistics*: *Demonstrations*, San Diego, 12-17 June 2016, 11-16. https://doi.org/10.18653/v1/n16-3003

[23] Kondratyuk, D. and Straka, M. (2019) 75 Languages, 1 Model: Parsing Universal Dependencies Universally. *Proceedings of the* 2019 *Conference on Empirical Methods in Natural Language Processing and the* 9*th International Joint Conference on Natural Language Processing* (*EMNLP-IJCNLP*), Hong Kong, 3-7 November 2019, 2779-2795. https://doi.org/10.18653/v1/d19-1279

[24] Dozat, T. and Manning, C.D. (2017) Deep Biaffine Attention for Neural Dependency Parsing. https://openreview.net/forum?id=Hk95PK9le

[25] Beesley, K.R. and Karttunen, L. (2003) Finite-State Morphology. CSLI Publications.

[26] Pimentel, T., *et al.* (2021) Findings of the SIGMORPHON 2021 Shared Task on Morphological Reinflection. *Proceedings of SIGMORPHON* 2021, Bangkok, 5-6 August 2021, 1-16.

[27] Habash, N. and Rambow, O. (2005) Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. *Proceedings of the* 43*rd Annual Meeting on Association for Computational Linguistics*, Ann Arbor, 25-30 June 2005, 573-580. https://doi.org/10.3115/1219840.1219911

[28] Devlin, J., Chang, M., Lee, K. and Toutanova, K. (2019) BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the* 2019 *Conference of the North*, Minneapolis, 2 June-7 June 2019, 4171-4186. https://doi.org/10.18653/v1/n19-1423