

NLPBeasts at AraSeg Shared Task 2026: A Character-Word BiLSTM-CRF for Paragraph-Aware Arabic Sentence Segmentation

Michael Salieb, Andrew Aziz, Kyrollos Soliman, Mina Shaker

American University in Cairo (AUC)

michaell12@aucegypt.edu, andrewwassem@aucegypt.edu,
kyrolloszakaria@aucegypt.edu, menanady@aucegypt.edu

Abstract

We describe NLPBeasts' submission to the Paragraph-Aware (PA) Closed track of the AraSeg 2026 shared task, which frames Arabic sentence segmentation as per-token binary boundary classification. We compare two closed-track-compliant systems trained without any pretrained embeddings or language models: (1) a feature-based CRF using hand-crafted punctuation, bracket, and paragraph-distance features, and (2) a character- and word-level BiLSTM with a linear-chain CRF decoder, both trained entirely from scratch on the shared task's training and development splits. The feature-based CRF plateaus at a hard recall ceiling ($\approx 82\text{--}83\%$) regardless of regularization strength, while the BiLSTM-CRF captures substantially more boundary cases through learned context. Our BiLSTM-CRF system, using only permitted training data with the development split reserved strictly for model selection, achieves $P=86.37$, $R=90.51$, $F1=87.60$ on the blind test set. We additionally report ablations on chunk length and training hardware (CPU vs. GPU), neither of which meaningfully changes development F1, suggesting our system's performance ceiling on this closed-track data size is data-limited rather than compute- or context-length-limited.

1 Introduction

Sentence segmentation is a foundational preprocessing step for most downstream Arabic NLP pipelines, yet Arabic punctuation is used inconsistently across genres, and paragraph structure interacts with sentence boundaries in ways that simple punctuation-based splitting cannot capture. The AraSeg 2026 shared task frames this as supervised sequence labeling: given a tokenized document, predict for every token whether a sentence boundary follows it. The Paragraph-Aware (PA) closed track we participate in additionally provides paragraph-break tokens as input and restricts sys-

tems to training only on the provided corpus, without pretrained embeddings or language models. This closed-track restriction makes the task interesting from a modeling perspective: without access to large pretrained representations, systems must rely either on strong hand-engineered priors or on what a small amount of task-specific data can teach a model from scratch.

Our initial hypothesis was that, given the small labeled corpus (174 training documents), a carefully feature-engineered CRF would be competitive with or superior to a neural sequence tagger trained from scratch. We test this hypothesis directly by building both systems under identical data conditions and comparing them on the same development set. We find the opposite: the CRF's engineered features impose a recall ceiling that regularization tuning cannot overcome, while a from-scratch character-word BiLSTM-CRF learns to recover a meaningfully larger share of true boundaries. We submit the BiLSTM-CRF as our final system and report both approaches, along with ablations that helped us understand where our system's remaining error is coming from.

2 Background

We participate in the **Paragraph-Aware (PA) Closed** track of AraSeg 2026 (Elkholly et al., 2026b,a). Each document is provided as a list of whitespace-delimited tokens, with paragraph breaks represented as literal newline tokens (" $\backslash n$ ") inside the token sequence. The label for each token is binary: 1 if a sentence boundary immediately follows that token, else 0. The closed track prohibits pretrained embeddings, pretrained language models, and any data outside the released AraSeg corpus; all representations must be learned from the training split itself. Systems are scored with per-document boundary precision, recall, and F1 on the 1 class, macro-averaged across documents

in a split.

The public data release provides 174 training documents and 222 development documents, ranging from 38 to over 21,000 tokens (mean 737), i.e. documents span short passages to full book chapters. A near-universal regularity in the training data is that documents end in a sentence boundary: 174/174 (100%) of training documents have a positive label on their final token, which motivates a simple postprocessing heuristic described in Section 3.

Sentence boundary detection has a long history outside Arabic: Kiss and Strunk (2006) treat it as unsupervised abbreviation detection followed by rule-based disambiguation, while Gillick (2009) show that a fully supervised, feature-based statistical classifier substantially outperforms such unsupervised heuristics on English news text, particularly on hard cases like abbreviation-final periods. Our CRF baseline follows this feature-based tradition, adapted to Arabic-specific punctuation and paragraph structure. On the Arabic side, general-purpose toolkits such as CAMEL Tools (Obeid et al., 2020) provide preprocessing, morphological analysis, and dialect identification, but do not include a dedicated, learned sentence segmentation component, leaving punctuation-based heuristics as the default in most Arabic NLP pipelines — precisely the gap AraSeg 2026 is designed to address.

3 System Overview

We implement and directly compare two closed-track-compliant systems for boundary prediction, trained under identical data conditions so that any performance difference between them reflects modeling choices rather than data access: a feature-based linear-chain CRF (Section 3.1) that relies on hand-crafted punctuation and paragraph-structure features, and a character-word BiLSTM with a learned CRF decoder (Section 3.2) that replaces those hand-crafted features with representations learned end-to-end from the training data. Section 3.3 then describes a postprocessing heuristic shared by both systems.

3.1 Feature-based CRF baseline

Our first system is a linear-chain CRF (Lafferty et al., 2001) over hand-crafted per-token features, trained with `sklearn-crfsuite`. For each token x_i we extract: the token’s surface form; a coarse shape class (WORD, DIGIT, PUNCT, or

NEWLINE); token length; boolean flags for strong sentence-final punctuation (period, Arabic question mark, exclamation mark, ellipsis), weak punctuation (comma, Arabic comma, semicolon, Arabic semicolon, colon), opening/closing brackets, and digits; boolean flags for document-initial/final position; identity and shape of tokens in a ± 2 window around x_i ; the shape of the next non-newline token (to distinguish sentence-final punctuation from paragraph-final punctuation); and the (capped) token distance to the nearest preceding and following newline token. Given a length- n token sequence $\mathbf{x}$, the CRF defines a distribution over label sequences $\mathbf{y}$ as

$$P(\mathbf{y} | \mathbf{x}) = \frac{\exp \left(\sum_{i=1}^n \psi(y_{i-1}, y_i, \mathbf{x}) \right)}{Z(\mathbf{x})} \quad (1)$$

where $\psi(y_{i-1}, y_i, \mathbf{x})$ linearly combines the emission features above (evaluated at position i) with a pairwise label-transition weight, and

$$Z(\mathbf{x}) = \sum_{\mathbf{y}'} \exp \left(\sum_{i=1}^n \psi(y'_{i-1}, y'_i, \mathbf{x}) \right) \quad (2)$$

normalizes over all label sequences. We tune the L1/L2 regularization coefficients c_1, c_2 over a 6×6 grid ($\{0.01, 0.05, 0.1, 0.3, 0.5, 1.0\}^2$) using `lbfgs` optimization with `all_possible_transitions=True`; as Section 5 shows, precision and recall respond very asymmetrically to this tuning, which we return to in the comparative error analysis of Section 5.

3.2 Character-word BiLSTM-CRF

Our submitted system replaces hand-crafted features with representations learned end-to-end. Each token x_i is encoded as the concatenation of (a) a learned word embedding ($d_w=100$) and (b) a character-level representation: characters of x_i are embedded ($d_c=30$) and passed through a bidirectional character LSTM (hidden size 50 per direction), whose final forward and backward hidden states are concatenated. This 200-dimensional token representation is passed through a 2-layer bidirectional word-level LSTM (hidden size 128 per direction, dropout 0.4), producing per-token emission scores via a linear layer to 2 classes, which are decoded with a linear-chain CRF layer (Huang et al., 2015) identical in form to Equations 1–2, but

with ψ now a learned emission score plus a learned 2×2 transition matrix rather than hand-built features. The model is trained by minimizing the negative log conditional likelihood (CRF loss) with Adam (lr = 10^{-3} , weight decay 10^{-5}), gradient norm clipping at 5.0, and dropout 0.4. Word and character vocabularies are built exclusively from the training split (28,628 word types, 172 character types), keeping the system closed-track compliant.

Since some documents run to tens of thousands of tokens, we split each document into non-overlapping chunks of at most `max_len` tokens for both training and inference, and stitch chunk-level predictions back into full-document sequences before scoring or writing submissions. This avoids out-of-memory failures on the longest documents at the cost of losing cross-chunk context at chunk boundaries.

3.3 Postprocessing

Motivated by the near-100% boundary rate on final tokens (Section 3), both systems apply a *force-last-token* heuristic: the final token’s predicted label is set to 1 if this improves development F1, which we verify empirically before applying it to blind test predictions.

4 Experimental Setup

Both systems are trained exclusively on the 174-document training split. The 222-document development split is used only for model selection: the CRF’s regularization coefficients are chosen by grid search against development F1, and the BiLSTM-CRF’s checkpoint is chosen by early stopping (patience of 6 epochs without development F1 improvement). Neither system updates its parameters using development labels at any point; blind test predictions (100 documents) are generated directly from the resulting model, with no additional training or fine-tuning on the development split. We use `sklearn-crfsuite` for the CRF baseline and `PyTorch` with `pytorch-crf` for the BiLSTM-CRF. Evaluation follows the shared task metric: per-document P/R/F1 on the boundary class, macro-averaged over documents.

5 Results

CRF regularization sweep. Table 1 shows representative points from our 6×6 grid search over (c_1, c_2) . Precision remains consistently in the low-to-mid 90s across the entire grid, while recall never

exceeds 83%, regardless of regularization strength. This pattern — a fixed gap that tuning cannot close — indicates the hand-crafted feature set itself is missing cues for a meaningful share of true boundaries, rather than the model being over- or under-regularized.

c_1	c_2	P	R	F1
0.01	1.0	92.56	81.48	84.56
0.1	0.01	91.21	82.30	84.53
0.3	0.1	91.62	82.59	84.82
0.5	0.01	91.44	82.79	84.84
1.0	1.0	92.63	81.36	84.35

Table 1: Representative CRF development results across the regularization grid (best in bold). Full grid: $P \in [91.2, 92.6]$, $R \in [81.4, 82.8]$, $F1 \in [84.3, 84.8]$.

BiLSTM-CRF and ablations. Our configuration (chunk length 300, patience 6) reaches development F1 88.80 (88.89 with the force-last heuristic), clearly ahead of the best CRF result. On the blind test set this system scores **P=86.37**, **R=90.51**, **F1=87.60**. Relative to development performance, blind test recall is higher and precision lower; since the blind and development sets are drawn from the same collection process, we take this as normal generalization variance rather than a systematic effect, though we note it rather than smoothing over it. To understand whether more context per chunk or more compute would help, we ran two ablations with chunk length increased to 500 and patience increased to 10: one on CPU, one on an NVIDIA RTX 3050 Ti laptop GPU. Table 2 shows neither ablation improves on the submitted configuration, and CPU and GPU training converge to statistically indistinguishable development F1. This suggests that, at our current corpus size (~ 400 total labeled documents under closed-track constraints), our system’s performance ceiling is governed by data availability rather than chunk context length or training compute.

Comparative error analysis. We do not have token-level error annotations, but the aggregate precision/recall profiles of the two systems are informative on their own. The CRF’s errors are heavily concentrated on one side: precision stays in the low-to-mid 90s across every point in our regularization grid (Table 1) while recall never exceeds 83%, meaning the CRF is rarely wrong when it predicts a boundary, but consistently misses roughly one in five true boundaries regardless of tuning —

a false-negative-dominated error profile consistent with a feature set that captures strong, unambiguous cues (sentence-final punctuation, brackets) but not weaker or more implicit ones. The BiLSTM-CRF’s precision and recall are far more balanced (development P=89.68, R=89.34 at the selected checkpoint; blind P=86.37, R=90.51), indicating it recovers many of the boundaries the CRF’s feature set cannot represent, at the cost of a small increase in false positives. This asymmetry suggests a union-style ensemble (predicting a boundary wherever either system does) could recover some of the CRF’s confidently-correct boundaries that the BiLSTM-CRF misses, at a small precision cost from the CRF’s comparatively rare false positives; whether the net F1 effect is positive depends on how much the two systems’ errors overlap, which we cannot determine without token-level alignment of their predictions. We were unable to test this within the shared task’s timeline and leave it, along with a full category-level breakdown by boundary type (punctuation-adjacent, paragraph-adjacent, neither), to future work.

Configuration	Best F1	+Force-last F1
Submitted (len 300, CPU)	88.80	88.89
len 500, patience 10, CPU	88.75	88.78
len 500, patience 10, GPU	88.81	88.84

Table 2: BiLSTM-CRF development F1 across chunk length, early-stopping patience, and hardware. All rows use identical architecture and optimizer settings.

6 Conclusion

We compared a feature-based CRF and a from-scratch character-word BiLSTM-CRF for closed-track paragraph-aware Arabic sentence segmentation. Contrary to our initial expectation that hand-engineered features would be competitive on a small labeled corpus, we found the CRF’s features impose a hard recall ceiling that no amount of regularization tuning removes, while the BiLSTM-CRF learns to recover substantially more boundaries from the same data. Our BiLSTM-CRF reaches F1=87.60 on the blind test set when trained strictly within the closed-track’s data constraints. Ablations over chunk length and training hardware show neither materially affects development performance, indicating that further gains at this corpus size are more likely to come from better architectures or feature integration (e.g., combining learned representations with the CRF’s punctuation/bracket

priors) than from longer context windows or additional compute.

Limitations

Our chunking strategy discards cross-chunk context at chunk boundaries, which may disproportionately affect the longest documents in the corpus. We did not have time, under the shared task’s testing-phase deadline, to run an ensemble of the CRF and BiLSTM-CRF systems. Section 5 reasons from the two systems’ error asymmetry that a union-style ensemble could plausibly help recall at modest precision cost, but this remains untested. We also did not perform a systematic hyperparameter search over the BiLSTM-CRF’s architecture (embedding sizes, LSTM depth, dropout), instead selecting reasonable defaults and comparing only chunk length, early-stopping patience, and hardware. Our error analysis is limited to aggregate P/R/F1; a category-level analysis of which boundary types (e.g., punctuation-adjacent vs. paragraph-adjacent vs. neither) the system still misses would likely clarify what further modeling changes would help most.

Acknowledgments

We thank the AraSeg 2026 organizers for the shared task and corpus.

References

- Mohammed Elkholy, Khalid N. Elmadani, Nizar Habash, and Bashar Alhafni. 2026a. [Arabic sentence segmentation across genres and punctuation conditions](#). *Preprint*, arXiv:2606.08025.
- Mohammed Elkholy, Khalid N. Elmadani, Nizar Habash, and Bashar Alhafni. 2026b. Arabic sentence segmentation shared task 2026. In *Proceedings of the Fourth Arabic Natural Language Processing Conference*, Budapest, Hungary. Association for Computational Linguistics.
- Dan Gillick. 2009. [Sentence boundary detection and the problem with the U.S.](#) In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 241–244, Boulder, Colorado. Association for Computational Linguistics.
- Zhiheng Huang, Wei Xu, and Kai Yu. 2015. [Bidirectional LSTM-CRF models for sequence tagging](#). *arXiv preprint arXiv:1508.01991*.

- Tibor Kiss and Jan Strunk. 2006. [Unsupervised multilingual sentence boundary detection](#). *Computational Linguistics*, 32(4):485–525.
- John Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289, Williamstown, MA, USA. Morgan Kaufmann.
- Ossama Obeid, Nasser Zalmout, Salam Khalifa, Dima Taji, Mai Oudah, Bashar Alhafni, Go Inoue, Fadhl Eryani, Alexander Erdmann, and Nizar Habash. 2020. [CAMEL tools: An open source python toolkit for Arabic natural language processing](#). In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 7022–7032, Marseille, France. European Language Resources Association.