Named Entity Recognition in Arabic

BIO Tagging: HMM, BiRNN/LSTM, BiLSTM–CRF, and BERT

Yasser Hifny

Faculty of Computers and Artificial Intelligence
Capital University

1. What is Named Entity Recognition?

Task

Identify spans that refer to named entities and assign each span a type.

زارَ مُحَمَّدٌ القاهِرَةَ والتقى مُمَثِّلِي شَرِكَةِ مايكروسوفت.

- Person: مُحَمَّدٌ
- Location: القاهِرَةَ
- Organization: مايكروسوفت

2. Why Sequence Tagging?

- NER assigns a label to each token, not just one label to a whole sentence.
- A named entity may contain one token or several tokens.
- Context is important: the same word can be an entity in one context and ordinary text in another.

ذهبَ عليٌّ إلى عمّانَ.

Token	عليٌّ	عمّانَ
Tag	B-PER	B-LOC

3. BIO Tagging Scheme

- **B-X** First token of an entity of type X.
- **I-X** Subsequent token inside the same entity.
- **O** Token outside any named entity.

Token	زارَ	عبدُ	الرَّحمنِ	جامعةَ	القاهرةِ	أمسِ
Tag	O	B-PER	I-PER	B-ORG	I-ORG	O

BIO is a boundary-encoding scheme; it is not itself a model.

4. Entity Types and Tag Inventory

For three entity types—person, organization, and location—the tag set is:

$$\mathcal{Y} = \{O, B\text{-PER}, I\text{-PER}, B\text{-ORG}, I\text{-ORG}, B\text{-LOC}, I\text{-LOC}\}.$$

Type	Example	BIO tags
Person	أحمد بن طولون	B-PER, I-PER, I-PER
Organization	جامعة القاهرة	B-ORG, I-ORG
Location	المملكة العربية السعودية	B-LOC, I-LOC, I-LOC
Other	زار اليوم	O, O

The actual label inventory must be read from the supplied dataset and kept consistent across all models.

5. From Arabic Text to BIO Labels

التقى الرئيسُ دونالد ترامب في واشنطن وفدَ الأممِ المتحدةِ.

Token	Label
التقى	O
الرئيسُ	B-PER
دونالد	I-PER
ترامب	I-PER
في	O
واشنطنِ	B-LOC
وفدَ	O
الأممِ	B-ORG
المتحدةِ	I-ORG

6. Approach 1 — Hidden Markov Model

An HMM models the hidden tag sequence $y_{1:T}$ that generated observed tokens $x_{1:T}$.

$$\hat{y}_{1:T} = \arg\max_{y_{1:T}} P(y_{1:T} \mid x_{1:T})$$

Using Bayes' rule, maximizing the posterior is equivalent to maximizing $P(x_{1:T}, y_{1:T})$ for a fixed sentence.

- Hidden states: BIO tags.
- Observations: Arabic tokens.
- Parameters: transition and emission probabilities estimated from training data.

7. HMM Joint Probability Factorization

With first-order Markov and emission independence assumptions:

$$P(x_{1:T}, y_{1:T}) = P(y_1)P(x_1 \mid y_1) \prod_{t=2}^{T} P(y_t \mid y_{t-1})P(x_t \mid y_t).$$

Equivalently,

$$P(y_{1:T}, x_{1:T}) = \underbrace{P(y_1) \prod_{t=2}^{T} P(y_t \mid y_{t-1})}_{\text{tag sequence}} \underbrace{\prod_{t=1}^{T} P(x_t \mid y_t)}_{\text{token emissions}}.$$

Training counts

Estimate transitions $P(y_t \mid y_{t-1})$ and emissions $P(x_t \mid y_t)$ from tagged sentences, with smoothing for unseen events.

8. HMM Transition and Emission Probabilities

$$P(y_t \mid y_{t-1}) = \frac{C(y_{t-1}, y_t)}{C(y_{t-1})}, \qquad P(x_t \mid y_t) = \frac{C(y_t, x_t)}{C(y_t)}.$$

Illustrative counts (toy values, not dataset measurements):

Event	Count	Estimate
$C(O, B\text{-LOC})$	40	$40/100 = 0.40$
$C(B\text{-LOC}, I\text{-LOC})$	18	$18/25 = 0.72$
$C(B\text{-LOC}, x)$ for token القاهرة	12	$12/20 = 0.60$

9. HMM Decoding: Why Viterbi?

For T tokens and $|\mathcal{Y}| = K$ tags, exhaustive search considers K^T paths.

Dynamic programming

Viterbi retains only the best partial path ending in each tag. This reduces the complexity to $O(TK^2)$.

Use log probabilities to avoid numerical underflow:

$$\log P(y_1) + \log P(x_1 \mid y_1) + \cdots$$

10. Viterbi Recurrence

Define $\delta_t(s)$ as the best log-score of any path ending in tag s at position t.

$$\delta_1(s) = \log P(s) + \log P(x_1 \mid s)$$

$$\delta_t(s) = \max_{s' \in \mathcal{Y}} [\delta_{t-1}(s') + \log P(s \mid s')] + \log P(x_t \mid s).$$

Store the maximizing predecessor:

$$\psi_t(s) = \arg\max_{s'} [\delta_{t-1}(s') + \log P(s \mid s')].$$

At the end, choose $\arg\max_s \delta_T(s)$ and backtrack through ψ.

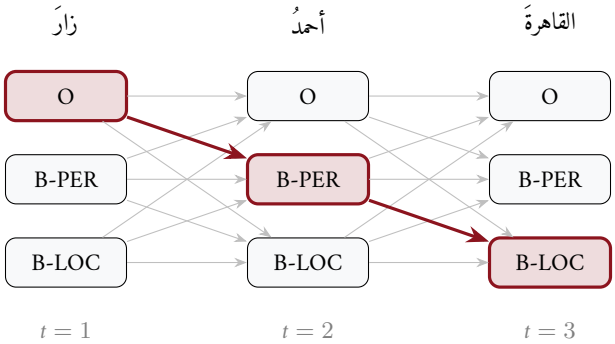
11. Viterbi Worked Example in Arabic

Sentence: زارَ أحمدُ القاهرةَ

Token	Candidate tag	Illustrative score	Interpretation
زارَ	O	$\delta_1(O) = -0.4$	Start
أحمدُ	B-PER	$\delta_2(B\text{-PER}) = -1.0$	Person begins
القاهرةَ	B-LOC	$\delta_3(B\text{-LOC}) = -1.5$	Location begins

The displayed numbers are pedagogical toy scores, not trained probabilities. In a real decoder each cell compares every predecessor tag and stores its best backpointer.

12. Viterbi Lattice and Backtracking



Red arrows illustrate one possible best path only; actual arrows are determined by trained transition and emission scores. Backtracking: start from the best final state and follow the stored backpointers from right to left.

13. HMM Strengths and Limitations

Strengths

- Fast training and decoding.
- Transparent transition/emission tables.
- Viterbi gives a globally best tag path under the model.

Limitations

- Strong independence assumptions.
- Sparse word emissions and OOV tokens.
- No rich contextual representations.
- BIO constraints need careful handling.

14. Approach 2 — Bidirectional RNN

A bidirectional RNN reads the sentence from left to right and right to left.

$$\overrightarrow{h}_t = \mathrm{RNN}_f(e_t, \overrightarrow{h}_{t-1}), \qquad \overleftarrow{h}_t = \mathrm{RNN}_b(e_t, \overleftarrow{h}_{t+1})$$

$$h_t = [\overrightarrow{h}_t; \overleftarrow{h}_t], \qquad p(y_t \mid x) = \mathrm{softmax}(Wh_t + b).$$

Every token receives a context-aware representation, and the output layer predicts one BIO label per token.

15. Arabic Example: Contextual Tagging

زارَ خالدٌ باريسَ ثمَّ عادَ إلى باريسَ.

Token	Predicted label	Contextual clue
خالدٌ	B-PER	Person name
باريسَ (first)	B-LOC	Destination of visit
باريسَ (second)	B-LOC	Destination after إلى

The same surface token can appear more than once; each occurrence is tagged using its own context.

16. Why LSTM Instead of a Basic RNN?

LSTM uses a cell state and gates to control information flow.

$$f_t = \sigma(W_f[h_{t-1}, e_t] + b_f), \quad i_t = \sigma(W_i[h_{t-1}, e_t] + b_i)$$

$$\tilde{c}_t = \tanh(W_c[h_{t-1}, e_t] + b_c), \quad c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$o_t = \sigma(W_o[h_{t-1}, e_t] + b_o), \quad h_t = o_t \odot \tanh(c_t)$$

These gates help retain useful information over longer contexts.

17. BiLSTM Token Classifier

1. Convert each Arabic token into an embedding e_t (random, pretrained, or character-aware).
2. Encode the sequence with forward and backward LSTMs.
3. Concatenate the two hidden states at every position.
4. Apply a linear layer and softmax over BIO labels.
5. Train with token-level cross-entropy, masking padding positions.

$$\mathcal{L}_{CE} = -\sum_{t=1}^{T} m_t \log p_\theta(y_t \mid x)$$

where $m_t = 0$ for padding and 1 for real tokens.

18. Approach 3 — BiLSTM + CRF

A softmax predicts each token independently conditional on the BiLSTM features. A CRF models dependencies between the output labels.

$$s_t(y) = W_y h_t + b_y$$

For a candidate sequence y, define its score:

$$S(x, y) = \sum_{t=1}^{T} s_t(y_t) + \sum_{t=2}^{T} A_{y_{t-1}, y_t}$$

$$P(y \mid x) = \frac{\exp S(x, y)}{\sum_{y'} \exp S(x, y')}$$

19. CRF Training: Conditional Probability

For an input sentence $x = (x_1, \ldots, x_T)$, a BiLSTM produces emission scores $e_t(k)$ for each BIO tag k. A linear-chain CRF scores a tag sequence $y = (y_1, \ldots, y_T)$ as:

$$S(x, y) = A_{\text{START}, y_1} + \sum_{t=1}^{T} e_t(y_t) + \sum_{t=2}^{T} A_{y_{t-1}, y_t} + A_{y_T, \text{END}},$$

where $A_{i,j}$ is the transition score from tag i to tag j.
The conditional probability is

$$P(y \mid x) = \frac{\exp S(x, y)}{Z(x)}, \qquad Z(x) = \sum_{y' \in \mathcal{Y}^T} \exp S(x, y').$$

Training objective

For the gold sequence y^*, maximize the conditional log-likelihood:

$$\log P(y^* \mid x) = S(x, y^*) - \log Z(x).$$

The challenge is computing $Z(x)$ without enumerating all K^T possible tag sequences, where $K = |\mathcal{Y}|$ is the number of tags.

20. CRF Training: Forward Algorithm

Define $\alpha_t(j)$ as the log-sum-exp of the scores of all partial tag sequences $(y_1, \ldots, y_t)$ with $y_t = j$.
Initialization: $\alpha_1(j) = e_1(j) + A_{\text{START},j}$.
Recurrence:

$$\boxed{\alpha_t(j) = e_t(j) + \text{LSE}_{i \in \mathcal{Y}} \left[\alpha_{t-1}(i) + A_{i,j}\right]} \qquad \text{LSE}(z_1, \ldots, z_K) = \log \sum_{i=1}^{K} \exp(z_i)$$

Termination:

$$\boxed{\log Z(x) = \text{LSE}_{j \in \mathcal{Y}} \left[\alpha_T(j) + A_{j,\text{END}}\right]}$$

- Log-sum-exp aggregates the scores of all possible paths.
- The computation costs $O(TK^2)$, rather than $O(K^T)$.

Arabic example: For زار محمد القاهرة ($T = 3$), the algorithm sums the scores of all K^3 possible BIO tag sequences without listing them individually.

21. CRF Training: Backward Algorithm and Marginals

The backward value $\beta_t(i)$ is the log-sum-exp of all suffix-path scores after position t, given tag i at position t.

Initialization: $\beta_T(i) = A_{i,\text{END}}$.

Backward recurrence:

$$\boxed{\beta_t(i) = \text{LSE}_{j \in \mathcal{Y}} \left[A_{i,j} + e_{t+1}(j) + \beta_{t+1}(j) \right]}$$

Tag marginal:

$$P(y_t = j \mid x) = \exp(\alpha_t(j) + \beta_t(j) - \log Z(x)) \,.$$

Transition marginal for $i \to j$:

$$P(y_{t-1} = i, y_t = j \mid x) = \exp(\alpha_{t-1}(i) + A_{i,j} + e_t(j) + \beta_t(j) - \log Z(x)) \,.$$

22. CRF Training: Gradients

For the negative log-likelihood $\mathcal{L} = -S(x, y^*) + \log Z(x)$, the gradients are:

$$\frac{\partial \mathcal{L}}{\partial e_t(j)} = P(y_t = j \mid x) - \mathbf{1}[y_t^* = j],$$

$$\frac{\partial \mathcal{L}}{\partial A_{i,j}} = \mathbb{E}[N_{i,j} \mid x] - N_{i,j}(y^*),$$

where $N_{i,j}$ counts transitions $i \to j$ (including those from START and to END), and $\mathbb{E}[N_{i,j} \mid x] = \sum_t P(y_{t-1} = i, y_t = j \mid x)$.

Training intuition

Gradient descent adjusts emission and transition scores to favor the gold sequence over competing paths. Forward-backward supplies the expected counts; the gold counts come directly from y^*.

23. CRF Decoding: Viterbi Algorithm

At inference, we need the single highest-scoring tag sequence, not the sum over all sequences. Viterbi replaces log-sum-exp with maximum.

Initialization: $\delta_1(j) = e_1(j) + A_{\text{START},j}$

Recurrence and backpointer:

$$\boxed{\delta_t(j) = e_t(j) + \max_{i \in \mathcal{Y}} [\delta_{t-1}(i) + A_{i,j}]}$$

$$\psi_t(j) = \arg\max_{i \in \mathcal{Y}} [\delta_{t-1}(i) + A_{i,j}]$$

Termination:

$$y_T^* = \arg\max_j [\delta_T(j) + A_{j,\text{END}}]$$

Backtracking: $y_{t-1}^* = \psi_t(y_t^*), \quad t = T, \ldots, 2$

Arabic example: زار محمد القاهرة $\longrightarrow$ `O B-PER B-LOC`

- Complexity: $O(TK^2)$.
- Forward-backward gives the partition function and marginals for training; Viterbi gives the best path for decoding.
- BIO constraints: give forbidden transitions a score of $-\infty$, e.g. `O`$\rightarrow$`I-PER` or `B-LOC`$\rightarrow$`I-ORG`.

20. BiLSTM–CRF vs. BiLSTM–Softmax

Property	BiLSTM + Softmax	BiLSTM + CRF
Prediction	Per-token argmax	Best complete sequence
Label dependencies	Not explicit	Learned transition scores
Training loss	Token cross-entropy	Sequence log-likelihood
Decoding	Independent	Viterbi
BIO consistency	May need repair	Can be encouraged by transitions

CRF does not automatically guarantee all BIO rules unless illegal transitions are masked or penalized appropriately.

21. Approach 4 — BERT Token Classification

BERT contextualizes each input token using bidirectional self-attention.

$$H = \mathrm{BERT}(x_1, \ldots, x_T), \quad h_t \in \mathbb{R}^d$$

$$p(y_t \mid x) = \mathrm{softmax}(W h_t + b)$$

Fine-tune the encoder and classification head jointly with masked token-level cross-entropy. For Arabic, an Arabic-pretrained encoder such as AraBERT is a natural candidate; multilingual BERT is another baseline.

22. BERT Tokenization and Label Alignment

Subword tokenizers may split one Arabic word into several pieces. Labels must be aligned deliberately.

Word	Subword pieces (illustrative)	Label strategy
والقاهرة	و + ال + قاهرة	Label first piece; mask rest
محمد	محمد	B-PER
الدولي	الدول + ي	Copy or mask subword labels consistently

Use the tokenizer's word alignment (e.g., word IDs); ignore special tokens and padding in the loss and evaluation. Do not count subwords as separate gold words during word-level scoring.

22b. Example: Label the First Subword, Mask the Rest

Sentence: زار محمد مطار القاهرة الدولي (“Mohamed visited Cairo International Airport”)

Token	[CLS]	زار	محمد	مطار	ال	قاهرة##	الدول	ي##	[SEP]
Word ID	None	0	1	2	3	3	4	4	None
Gold word label	–	O	B-PER	B-LOC	I-LOC	I-LOC	I-LOC	I-LOC	–
Training label	**-100**	**O**	**B-PER**	**B-LOC**	**I-LOC**	**-100**	**I-LOC**	**-100**	**-100**

- **Rule:** a subword gets its word’s label only if its word ID differs from the previous token’s word ID; otherwise it gets -100.
- -100 is the ignore index of the cross-entropy loss, so special tokens and non-first pieces add nothing to the loss.
- At prediction time, keep only the first-piece predictions, giving one label per word: زار/O, محمد/B-PER, مطار/B-LOC, القاهرة/I-LOC, الدولي/I-LOC.

23. BERT + Softmax Inference

1. Tokenize a sentence and preserve the mapping from subwords to original words.
2. Obtain contextual token vectors from the final BERT layer.
3. Apply a linear classifier and softmax to each evaluated word position.
4. Map predicted BIO labels back to words.
5. Optionally repair invalid BIO sequences, and report the policy used.

زارَ محمدٌ جامعةَ القاهرةِ.

24. Compare the Four Modeling Approaches

Model	Context	Sequence constraints	Main trade-off
HMM + Viterbi	Local Markov state	Yes, via transition model	Simple; sparse emissions
BiRNN/BiLSTM + Softmax	Bidirectional recurrent context	No explicit global label model	Straightforward baseline
BiLSTM + CRF	Bidirectional recurrent context	Learned transition scores	Strong structured decoder
BERT + Softmax	Transformer self-attention	No explicit CRF layer	Rich pretrained contextual features

25. Evaluation: Entity-Level Precision, Recall, F1

For NER, use exact-span entity-level scoring, not only token accuracy.

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN}$$

$$F_1 = 2\frac{PR}{P + R}$$

An entity is correct only if its type and complete start/end boundaries match the gold annotation. Report micro entity-level F1 (e.g., with `seqeval`) and state the evaluation convention.

26. Fair Comparison Protocol

- Use exactly the same train/dev/test split for all models.
- Fix random seeds and select checkpoints using development entity-level F1.
- HMM: estimate smoothed transition/emission probabilities from train only.
- BiLSTM–CRF: tune embedding size, hidden size, dropout, and learning rate on dev.
- BERT: use the same original word boundaries and label mapping; tune epochs and learning rate on dev.
- Compute the final test F1 once the configurations are fixed.

27. F1 Results: Do Not Confuse Template Values with Measurements

Model	Entity-level micro-F1	Status
HMM + Viterbi	Not measured	Requires training run
BiLSTM + CRF	Not measured	Requires training run
BERT + Softmax	Not measured	Requires fine-tuning run

Important

The dataset archive could not be downloaded in this execution environment, so no valid test F1 values were computed. Numerical scores are intentionally not fabricated. Run the assignment experiments on the linked dataset and fill this table with measured results.

28. Error Analysis

Inspect entity-level false positives, false negatives, and boundary errors.

Gold span	Prediction	Error type
جامعة القاهرة (ORG)	القاهرة (LOC)	Boundary and type
عبد الرحمن (PER)	عبد (PER)	Boundary
وزارة الصحة (ORG)	وزارة الصحة (ORG)	Correct

Questions: Are errors caused by rare names, tokenization, inconsistent annotation, or confusion between organizations and locations?

29. Assignment — Arabic NER with BIO

Dataset repository:
`https://github.com/Alaa-Shaker/Arabic_Named_Entity_Recognition_dataset`

1. Download and inspect the archive and accompanying reader notebook.
2. Document the actual entity inventory, file format, and annotation conventions.
3. Convert the data to sentence-level token/BIO sequences; validate malformed spans.
4. Create fixed train/dev/test splits (or preserve supplied splits if available).
5. Implement HMM + Viterbi, BiLSTM–CRF, and BERT + softmax.
6. Report entity-level precision, recall, micro-F1, training time, and inference time.

30. Assignment: HMM Requirements

- Estimate initial, transition, and emission probabilities from training sentences.
- Apply smoothing to avoid zero probability for unseen transitions/emissions.
- Decode with log-space Viterbi and backpointers.
- Define an unknown-token strategy (e.g., UNK or word-shape features).
- Check BIO legality and report entity-level F1 on the held-out test set.

31. Assignment: Neural Model Requirements

BiLSTM–CRF

- Word embeddings
- Bidirectional LSTM encoder
- Linear emissions + CRF
- Sequence likelihood and Viterbi decoding

BERT + Softmax

- Arabic or multilingual pretrained encoder
- Word/subword label alignment
- Masked cross-entropy
- Entity-level evaluation

32. Deliverables and Grading

1. Reproducible source code and environment requirements.
2. Data statistics, label inventory, and split counts.
3. Training curves and selected hyperparameters.
4. Test table: precision, recall, entity-level micro-F1 for all three required systems.
5. Error analysis with at least ten representative Arabic examples.
6. Short discussion of accuracy, efficiency, and model limitations.

33. Key Takeaways

- BIO tags turn entity boundaries and types into a token sequence labeling task.
- HMMs factorize joint probabilities and use Viterbi to find the best path.
- BiLSTM learns bidirectional context; CRF models label-to-label dependencies.
- BERT provides pretrained contextual representations with a token-level softmax head.
- Reliable comparison requires identical splits and exact entity-level F1 evaluation.

Thank You

شكراً لكم

Questions and discussion