Journal of Artificial Intelligence and Capsule Networks (ISSN: 2582-2012)
www.irojournals.com/aicn/

Adaptive Metaheuristic Optimization of AraBERT for Arabic NER: A Comparative Study of PSO and TLBO

Belal K. ELFarra[1], Mohammed Alhanjouri.[2]

Islamic University of Gaza, P.O. Box 108, Gaza Strip, Palestine.

Email: [1]fbelal@iugaza.edu.ps, [2]mhanjouri@iugaza.edu.ps

Abstract

Despite the state-of-the-art results obtained with transformer-based models like AraBERT, Arabic NER still appears to be very sensitive to the setting of hyperparameters. This usually requires a lot of manual tuning; the process is inefficient, time-consuming, and often results in suboptimal performance. In the context of this paper, an automatic framework for hyperparameter optimization was suggested using metaheuristic algorithms: PSO and TLBO. This is the first comparative research on the efficiency of metaheuristic algorithms for the optimization of Arabic NER. In this regard, this paper has applied these algorithms using the Wojood dataset and the aubmindlab/bert-base-arabertv2 to optimize the main hyperparameters: learning rate, batch size, and the number of epochs. The results revealed that PSO outperforms TLBO and a traditional approach for all test Micro-F1 scores, achieving a maximum of 0.8813, compared to 0.8755 for TLBO. Furthermore, PSO outperformed TLBO in terms of mean performance: 0.8708 versus 0.8561, with better convergence stability: a standard deviation of 0.0164 versus 0.0286. Although TLBO converged slightly faster than PSO, PSO demonstrated more robust generalization and reliability across runs. Overall, the findings confirm that metaheuristic optimization can substantially improve Arabic NER, with PSO standing out as the most effective, stable, and efficient optimizer for transformer-based Arabic NLP models.

Keywords: Artificial Intelligence, Optimization, Metaheuristic, Particle Swarm Optimization, Teaching–Learning-Based Optimization, Natural Language Processing, Named Entity Recognition, Machine Learning, Deep Neural Network.

DOI: https://doi.org/10.36548/jaicn.2025.4.002
Received: 19.09.2025, received in revised form: 20.10.2025, accepted: 03.11.2025, published: 12.11.2025

1. Introduction

NER is a field in Natural Language Processing (NLP), that specializes in identifying and classifying entities and elements in a text, such as names of people, organizations, and locations. The Arabic language is a morphologically rich and complex language characterized by orthographic ambiguities and a scarcity of high-quality annotated corpora, which presents significant challenges for applying NER. However, many models based on Transformer technology, such as AraBERT and its variants, have improved NER performance; yet one of the challenges they face is their high sensitivity to hyperparameters settings which can strongly influence both convergence and generalization. Despite advances in Arabic NER, few studies address efficient hyperparameter optimization for transformer models.

Using manual search or any other traditional approaches for hyperparameter tuning, is computationally expensive, time-consuming, and often produces suboptimal configurations. To overcome these challenges, this study compared the effectiveness of two metaheuristic optimization algorithms in tuning automatically and efficiently the hyperparameters of transformer-based NER models. This research introduces a novel comparative framework for integrating population-based metaheuristic optimization (PSO and TLBO) into transformer-based Arabic NER for automatic adaptive hyperparameter tuning of AraBERTv2:

PSO: Its idea mimics the movement of swarms searching for food. It explores the search space efficiently and identifies near-optimal solutions rapidly.

TLBO, on the other hand, mimics the teacher–student learning process and focuses on convergence stability while exploiting high-quality solutions.

We integrate both algorithms with the AraBERTv2 transformer model [1] and test them on the Wojood ArabicNER dataset. Both PSO and TLBO effectively searched the hyperparameter space of learning rate, batch size, and training epochs. The results demonstrate that both PSO and TLBO substantially outperform traditional tuning methods, with PSO achieving the best overall test micro-F1 score of 0.8813, marking a significant advancement in ArabicNER optimization.

2. Background

2.1 Particle Swarm Optimization

The development of PSO traces back to studies of collective animal behavior, particularly flocking and foraging dynamics. One of the earliest inspirations was Reynolds' Boid model [2], which simulated flocking through simple local rules. In this model, birds were represented as points with initial random positions and velocities, and their movement adjusted to match their neighbors. While simplistic, adding random velocity components produced more realistic collective motion. Building upon this, Heppner's cornfield model simulated birds locating food sources. Each bird adjusted its velocity based on its personal best experience (p_{best}) and the group's best-known position (g_{best}) [3]. These simulations demonstrated how memory and social sharing could accelerate convergence toward optimal solutions. Motivated by these observations, Kennedy and Eberhart [4] abstracted the flocking process into a computational algorithm. They modeled individuals as "particles" with only position and velocity, omitting physical properties such as mass or volume. Particles navigated a multidimensional search space by combining cognitive learning (self-experience) with social learning (peer influence). This framework forms the foundation of the PSO algorithm. Figure 1 presents the PSO flowchart, illustrating the step-by-step process of the algorithm. A particle is characterized by two key components: its position vector $Xi = (x_{i1}, x_{i2}, \ldots, x_{iD})$ and its velocity vector $Vi= (v_{i1}, v_{i2}, \ldots, v_{iD})$, where D is the problem dimension and $i \in \{1, 2, \ldots, N\}$ for a swarm of size N.

Each particle tracks two best solutions [5]:

Personal best ($P_{best,i}$): the best position discovered by particle *i*.

Global best (g_{best}): the best position found by the entire swarm (or local best P_l in neighborhood-based variants).

The PSO algorithm iteratively updates particle velocities and positions according to the following rules:

$$v_{id}^{t+1} = \omega v_{id}^{t} + c_1 r_1 \left(p_{best,id}^{t} - x_{id}^{t}\right) + c_2 r_2 \left(g_{bestr,d}^{t} - x_{id}^{t}\right) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (2)$$

where:

ω is the inertia weight controlling exploration vs. exploitation,

c_1, c_2 are cognitive and social learning coefficients,

r_1, r_2 are random numbers uniformly distributed in [0,1],

$p_{best,id}^t$ is the personal best in dimension d, at time t,

$g_{best,d}^t$ is the global best in dimension d, at time t.

The motivation for PSO lies in its natural inspiration, simplicity, and adaptability. Unlike gradient-based optimization methods, PSO does not require differentiability or continuity of the objective function, making it suitable for nonlinear, high-dimensional problems. Its stochastic and parallel design enables faster convergence than many traditional approaches. Nevertheless, challenges such as premature convergence and multimodal landscapes prompted refinements, including inertia weights, constriction factors, and hybrid variants.

2.2 Teaching–Learning-Based Optimization

The TLBO algorithm, proposed by Rao et al. [5], was inspired by the educational process in classrooms. Unlike evolutionary algorithms such as the Genetic Algorithm (GA), Differential Evolution (DE), and PSO, which rely on several control parameters, TLBO was designed to be a parameter-free optimizer, requiring only the number of learners (n_L) and the maximum number of function evaluations ($maxNFEs$) [6]. This made TLBO particularly attractive for practitioners seeking simplicity and robustness. The analogy stems from two central mechanisms of education: the teacher's influence on students and the peer-to-peer learning among students. These dynamics were mapped into two sequential phases [7]:

1. Teacher Phase (global search/intensification): The best-performing learner becomes the teacher (T), shifting the population mean ($MeanL$) toward better knowledge.

$$stepsize_i = (T - TF \cdot MeanL) \quad (3)$$

$$newL = L + rand(i,j) \cdot stepsize_i \quad (4)$$

where $TF \in \{1,2\}$ is the teaching factor and $rand(i,j) \sim U(0,1)$.

2. Learner Phase (local search/diversification): Students improve by interacting with randomly selected peers (L_{rp}), either moving toward stronger learners or diverging to maintain diversity.

$$stepsize_i = Li - Lrpif\ PFiti < PFitrp \quad (5)$$

$$newL = L + rand(i,j) \cdot stepsize_i \quad (6)$$

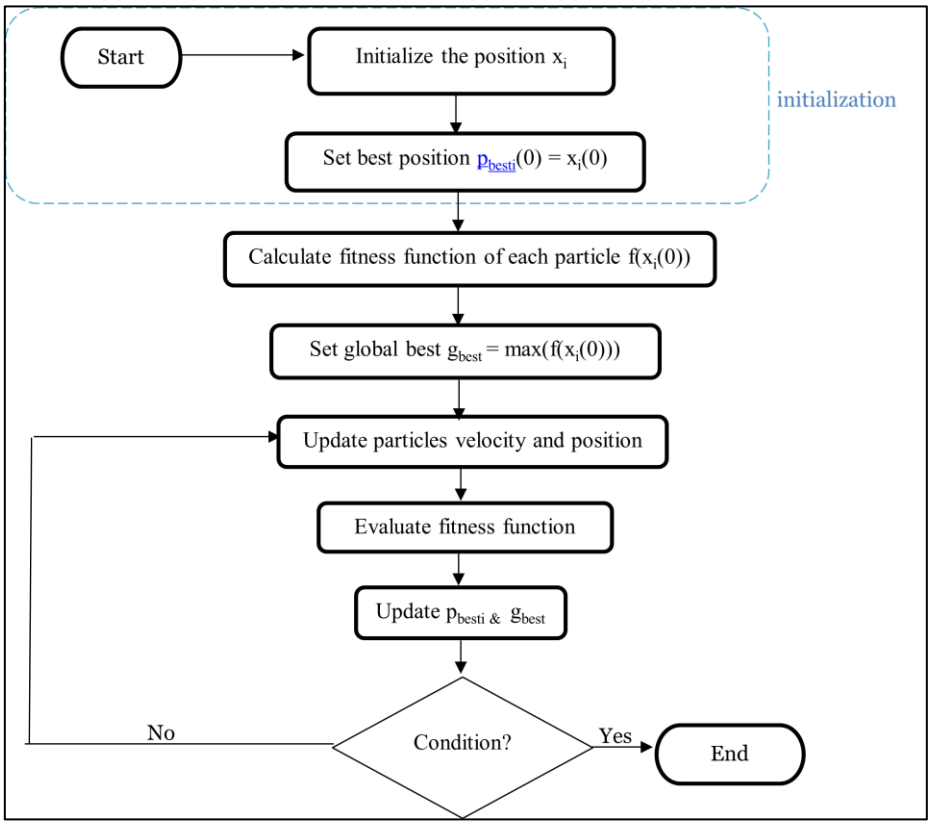

Figure 1. Flowchart Shows the Step-By-Step Process of PSO Algorithm

After both phases, a replacement strategy ensures that better solutions are retained. This iterative process continues until the stopping criterion is satisfied. The flowchart in Figure 2, illustrates the TLBO algorithm step by step [7].

2.3 Named Entity Recognition

NER is one of the core tasks in NLP that deals with identifying and labeling names of people, organizations, locations, and other entities mentioned in text. Early NER systems mainly relied on rule-based methods that used handcrafted linguistic patterns and expert-designed gazetteers. While these systems worked reasonably well for specific languages, they were difficult to scale or adapt. The move to machine learning changed that. Models like Conditional Random Fields (CRF), Support Vector Machines (SVM), and Hidden Markov Models (HMM) introduced a data-driven way of learning features such as word shapes, parts of speech, and contextual information. Later, hybrid methods that mixed rules with statistical

learning helped make NER systems more flexible. In recent years, deep learning has taken the lead. Neural networks—especially BiLSTM-CRF and transformer-based models—can automatically learn complex linguistic patterns and context without manual feature design. ArabicNER remains more difficult because of its rich morphology, lack of capitalization, and the scarcity of large, annotated datasets compared to English [8].

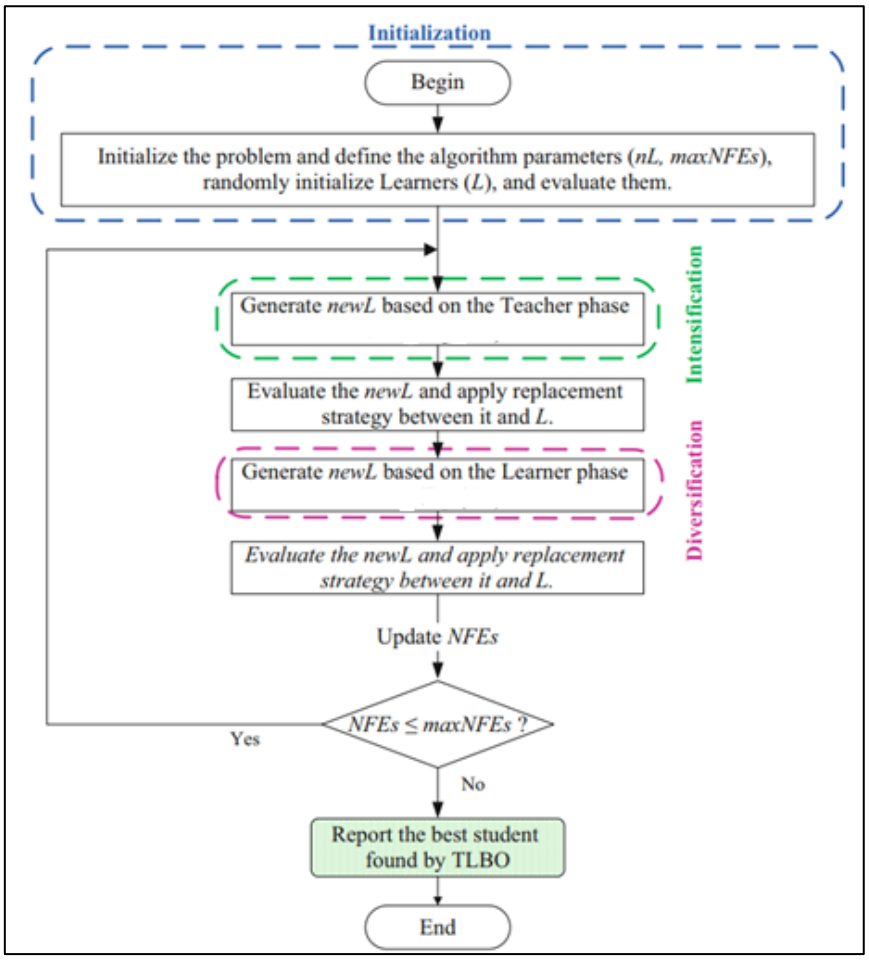

Figure 2. Flowchart of TLBO Algorithm [6]

3. Related Works

Early efforts in NER mostly relied on traditional machine learning algorithms such as CRF, Support Vector Machines (SVM), and Random Forests [9, 10, 11]. These models depended heavily on manually crafted linguistic features and rules, which made it difficult to adapt them to new domains or languages. To improve flexibility, hybrid systems that combined grammar-based heuristics with gazetteers were later proposed [11, 12, 13]. Still, these methods remained sensitive to Arabic's rich morphology and orthographic variations [14, 15].

The rise of deep learning brought a major shift in how NER systems were built. Models using recurrent or convolutional layers—such as BiLSTM-CRF and CNN architectures—helped capture context across sequences more effectively, though they often came with higher computational costs and longer training times [16, 17, 18]. The introduction of transformer-based models, especially BERT and its Arabic versions [19, 20], further pushed performance forward by learning bidirectional semantic relationships in text. Recent work has also explored specialized domains, including Arabic biomedical NER [21] and financial NER [22], showing notable gains even with limited labeled data.

When it comes to ArabicNER resources, early corpora such as ANERCorp [23], CANERCorpus [24], and ACE2005 [25] provided valuable starting points but had restricted domain and annotation coverage. More recent datasets like OntoNotes5 [26] expanded entity variety, though they still focused mainly on Modern Standard Arabic. The Wojood corpus introduced by [27] addressed these gaps by providing the largest ArabicNER dataset to date, supporting both flat and nested entity structures across dialectal and formal Arabic. This dataset has since become a benchmark for evaluating deep learning and transformer-based models in ArabicNER.

While the aforementioned studies primarily emphasize model architecture and dataset quality, hyperparameter optimization—a key determinant of transformer performance—remains underexplored. Most ArabicNER research continues to rely on manual or grid-based tuning, which is computationally expensive and non-reproducible. Metaheuristic algorithms such as PSO and TLBO have shown promise in other domains for efficiently navigating complex search spaces. For instance, PSO has been successfully applied to neural architecture search, feature selection, and text classification [28], while TLBO has demonstrated competitive results in hyperparameter tuning for sentiment analysis and topic modeling [29]. However, their integration into ArabicNER pipelines remains largely unexplored. This study addresses that gap by systematically comparing PSO and TLBO for optimizing AraBERT hyperparameters on the Wojood dataset, analyzing their convergence stability, learning dynamics, and impact on entity-level recognition performance.

4. Methodology

Figure 3 presents the flowchart of this study for the application of two population-based metaheuristic algorithms, PSO and TLBO, to automatically tune the hyperparameters-learning

rate, batch size, and number of epochs-of an ArabicNER model developed based on AraBERTv2. The implementation was developed using the open-source COMP9312 repository by Khalilia [30], in which the main goal of this optimization is to maximize the validation Micro-F1 score while minimizing both manual tuning effort and computational overhead. Each optimizer (PSO and TLBO) assessed a population of six candidate configurations across five optimization iterations, resulting in a total of 30 individual training runs per optimizer. Each run fine-tuned the AraBERTv2-based NER model for five epochs on the Wojood dataset, yielding an aggregate of approximately 150 training epochs per optimizer. This distributed training setup allowed for a diverse exploration of hyperparameters while maintaining individual runs to be computationally efficient. TLBO and PSO optimized key hyperparameters to maximize the validation Micro-F1. During the runs, convergence was generally attained after the first two to three epochs, indicating that both optimizers quickly found stable configurations without overfitting.

4.1 Objective

The optimization task consists of searching for the best set of training hyperparameters by maximizing the model's validation F1-score. The tuned hyperparameters include the following:

Learning rate (lr): continuous range [0.00005, 0.0001]

Batch size (bs): discrete set {16, 32}

Number of epochs (ep): integer range [3, 7]

Each of them has been further mapped to a normalized continuous search space, [0,1], to enable uniform optimization.

4.2 Dataset

Wojood ArabicNER [23], a manually annotated corpus that supports 21 entity types (e.g., LAW, QUANTITY, LANGUAGE, DATE, OCCUPATION, etc.), was used for the experiments. The information is split into three standard splits and supplied in the BIO tagging format:

- Training → sentences: 23124 tokens: 390900

- Validation → sentences: 3303 tokens: 57550
- Test → sentences: 6605 tokens: 113716

This distribution maintains a balanced validation and test evaluation while guaranteeing adequate training coverage. The dataset is representative for general-purpose ArabicNER evaluation since it contains Modern Standard Arabic (MSA) and minimal dialectal variation.

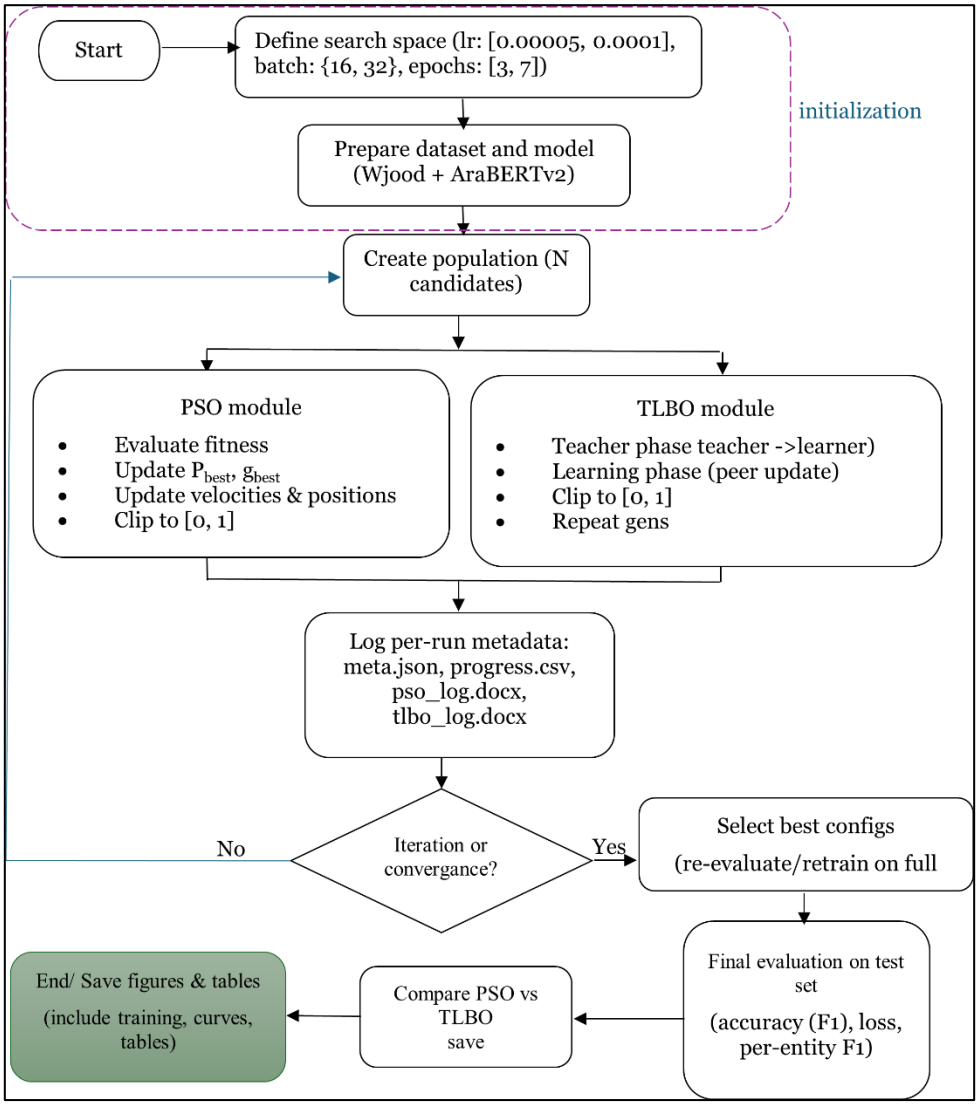

Figure 3. Flowchart Illustrating the Hyperparameter Optimization Process.

4.3 Model Evaluation

For every potential configuration produced by TLBO or PSO:

1. The Wojood dataset was used to refine the corresponding AraBERTv2 model for token classification using cross-entropy loss.

2. For reproducibility, a fixed random seed (1) was used to train each model.

3. The fitness value was calculated using the validation set's micro-averaged F1 score.

4. Learning rate, batch size, epochs, and the obtained F1 score were recorded in the results.

The top-performing configurations were then re-evaluated on the test set for final comparison.

4.4 Particle Swarm Optimization

As previously stated, each particle in PSO is a candidate solution that, when defined by its position X_i and velocity V_i in the search space, represents a possible hyperparameter set.

Algorithm Steps:

1. Initialize (N) particles randomly in [0,1].
2. Evaluate each particle's fitness (validation F1).
3. Assign each particle's best-known position $p_{best,i}$ and the global best g_{best}.
4. Update particle velocities and positions using equation (1):
5. Clip all updated positions to remain within [0,1].
6. Repeat for a fixed number of iterations until convergence or budget exhaustion.

The optimal configuration is determined by selecting the particle with the highest validation F1 following all iterations. As suggested by Clerk & Kennedy [31], we set inertia weight $\omega = 0.72$ to maintain a moderate trade-off between exploration and exploitation, and we initialized c1 and c2 to 1.49 each, balancing cognitive and social learning. Stable convergence under these values was verified by sensitivity tests.

4.5 Teaching–Learning-Based Optimization

Minimal parameter tuning is needed for TLBO. While the learner planning strategy adhered to Rao et al. [5], guaranteeing that both teacher and learner phases contribute equally to convergence, the number of generations (5) was selected to match PSO's iteration budget for fairness.

TLBO proceeds in two phases per generation: Teaching Phase and Learning Phase. Algorithm Steps:

1. Initialize *N* learners (solutions) randomly and evaluate their fitness.
2. Teacher Phase:
    - Identify the best-performing learner as the teacher.
    - Compute the mean performance of all learners.
    - Update each learner according to equations (3, 4).
3. Learning Phase:
    - Randomly pair learners (i, j)
    - If learner *i* performs worse, update towards *j*; otherwise, update *j* towards *i*.
    - Clip updated solutions to [0,1] and re-evaluate fitness.
4. Repeat for a fixed number of generations.
5. The learner with the highest validation F1 is selected as the best configuration.

TLBO's structure requires no algorithm-specific parameters, offering stable convergence and easy implementation.

4.6 Experimental Setup

All experiments were conducted in Google Colab using a single NVIDIA Tesla T4 GPU with 16 GB VRAM.

- Population size: 6 learners/particles

- Iterations (PSO) / Generations (TLBO): 5
- Random seed: 1 (fixed for reproducibility)
- Evaluation metric: Validation and Test Micro-F1
- Dataset: Wojood ArabicNER corpus, containing ~550K tokens across 21 entity types

Each optimizer ran independently under identical computational budgets to ensure a fair comparison.

4.7 Output and Evaluation

At the end of the optimization, both algorithms produced their best-performing configurations, which were retrained on the full training set and evaluated on the held-out test set. The comparison considered:

- Best Micro-F1 achieved
- Convergence rate
- Entity-level F1 improvements

5. Results and Discussion

An appropriate metaheuristic choice is based on the balance of exploration and exploitation, the complexity of parameters, stability of convergence, and computational budget. PSO provides flexible exploration through swarm velocity dynamics. TLBO is built upon parameter-free simplicity, offering smooth convergence. In TLBO, the teacher–learner model accelerates early convergence by explicitly driving the weaker learners toward the best performer. TLBO also suffers from restricted exploration due to limited swarm diversity and often converges prematurely. The stochastic update of the velocity vectors promotes a broader search in PSO for better global optima.

5.1 Overall Performance

The proposed metaheuristic optimization framework significantly enhanced the fine-tuning performance of the aubmindlab/bert-base-arabertv2 model for Arabic Wojood NER.

Both PSO and TLBO were able to effectively explore the hyperparameter space of learning rate, batch size, and number of training epochs for the maximization of the validation Micro-F1 score. Figure 4 depicts the traces of the best test F1-score during the optimization runs for PSO and TLBO. Correspondingly, one trace consists of the maximum F1-score obtained during one single run, so the run-to-run variability and stability can be depicted accordingly. PSO achieved consistently higher and more stable performance; its best-run F1-scores ranged between 0.86 and 0.88, peaking at 0.8813. TLBO reached a peak of 0.8755, but it was characterized by higher variability among runs. PSO resulted in an average F1-score of 0.8708, while TLBO's average was 0.8561, their respective standard deviations being 0.0164 and 0.0286. These results further support that PSO is better in stability and convergence reliability. In general, PSO managed to consistently findhigh-quality hyperparameter settings and reinforced its advantage over TLBO as the better optimizer for fine-tuning Arabic NER models in this study.

Figure 5 presents the distribution of F1-scores resulting from multiple runs of the optimizations by PSO and TLBO. Concretely, PSO has a higher average F1-score of 0.8708 with less dispersion, having a standard deviation of 0.0164, while TLBO has an average of 0.8561 with a standard deviation of 0.0286. In summary, smaller dispersion in the distribution of F1-scores for PSO illustrates increased stability in the optimization process, which is expected to result in better hyperparameter configurations more reliably across different runs.

Figure 6 shows a direct visual comparison, using a grouped bar chart, between PSO and TLBO on core evaluation metrics. More precisely, and as seen in detail in Table 1, against previous literature findings, PSO achieved the best test F1-score (0.8813 vs. 0.8755, +0.15%), while TLBO has a slightly higher precision (0.86 vs. 0.85, +1.2%) and recall (0.89 vs. 0.88, +1.1%). TLBO converged a little faster at 2 versus 3 epochs, reaching F1-scores above 0.87 within two epochs. On the other hand, PSO has better stability; its standard deviation is lower (0.0164 vs. 0.0286) and thus is more consistent across optimization runs. Overall, PSO stands out for maximizing both reliability and absolute F1, while TLBO is competitive in precision and recall.

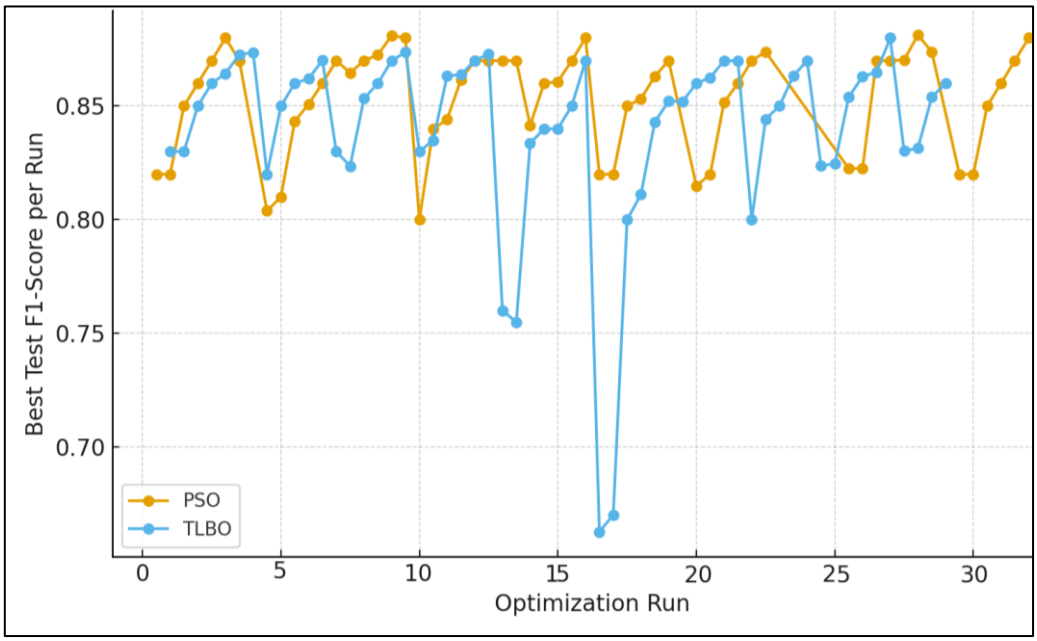

Figure 4. Test F1-Score Progression Across Optimization Runs

Table 1. The Comparison Results between TLBO and PSO

Algorithm	Best Test Micro-F1	Stability	Test Loss	Best Epoch	LR	Batch Size	Exploration	Exploitation
PSO	0.8813	0.0164	0.0352	3	5×10^{-5}	32	High	High
TLBO	0.8755	0.0286	0.0383	2	4×10^{-5}	32	Moderate	High

5.2 Entity-Level Analysis

Figure 7 compares the per-entity F1-scores of PSO and TLBO on the major entity types in the Arabic Wojood NER task. One can observe that PSO generally outperforms TLBO in both frequent and less frequent entity categories. The high-frequency entities DATE, ORG, PERS, and GPE achieved F1-scores between 0.86 and 0.94 under PSO, consistently higher than those of TLBO with small but meaningful margins. Lower-frequency or domain-specific entities, such as MONEY, LOC, and WEBSITE, were found to retain better stability and recall for PSO, while the latter two narrowed the gap due to their limited number of training instances.

TLBO produced competitive results overall but showed slightly higher variance and weaker generalization with regard to rare entities. It follows that PSO achieved higher and more consistent per-entity F1-scores, which reflect its better adaptability and robustness across dominant and low-support entity types. Table 2 lists the detailed per-entity results of each optimizer.

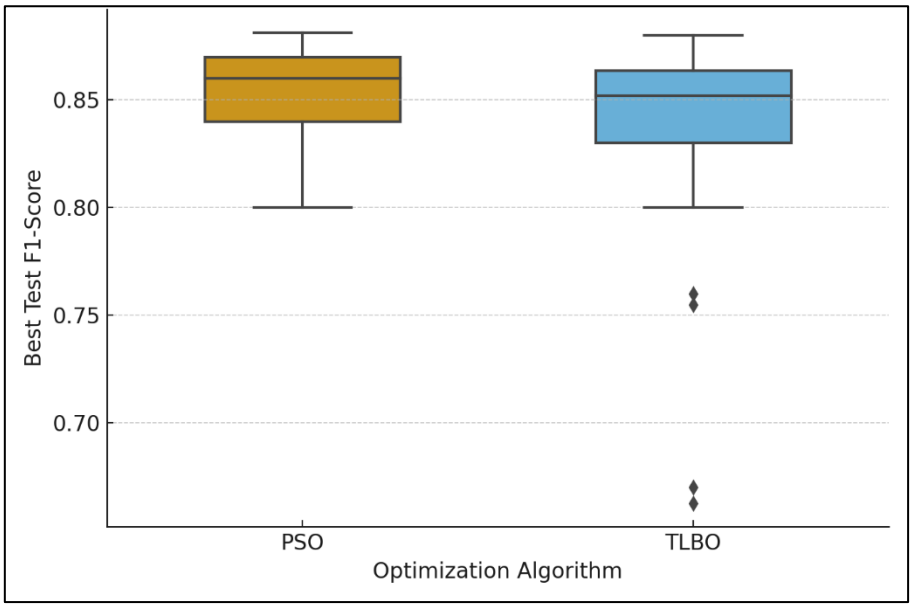

Figure 5. F1-Score Distribution Comparison Between PSO and TLBO

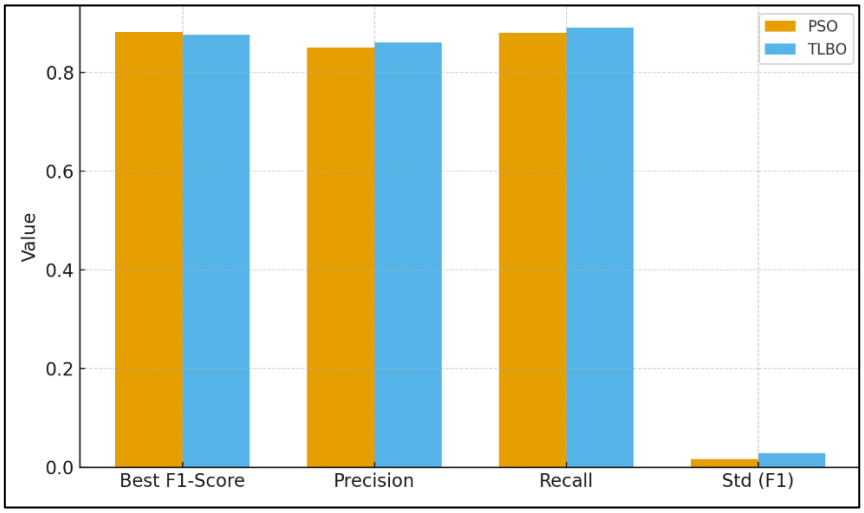

Figure 6. Summary Performance Metrics Comparison

Accordingly, frequent entities like DATE, PERS, ORG, and GPE reached F1 scores higher than 0.88, showing the remarkable language modeling of AraBERT when trained with enough examples. Similarly, mid-frequency entities like OCC, EVENT, and LAW showed

comparably high scores that benefit from the steady convergence of TLBO and stable parameter learning. On the other hand, some of the very low-frequency entities resulted in no positive predictions, including QUANTITY, UNIT, and CURR, due to the severe data imbalance in ArabicNER corpora.

Table 2. Summary of the Per-Entity F1 Scores for TLBO's Best Run

Entity	Support	PSO			TLBO		
		Precision	Recall	F1	Precision	Recall	F1
DATE	3028	0.93	0.95	0.94	0.90	0.94	0.92
PERS	1411	0.93	0.95	0.94	0.92	0.93	0.92
PERCENT	33	0.97	0.91	0.94	0.32	0.30	0.31
ORDINAL	889	0.94	0.92	0.93	0.85	0.90	0.87
ORG	3182	0.89	0.92	0.90	0.82	0.91	0.86
GPE	2254	0.89	0.91	0.90	0.87	0.86	0.87
OCC	1091	0.87	0.87	0.87	0.73	0.84	0.78
CARDINAL	341	0.81	0.87	0.84	0.70	0.70	0.70
MONEY	33	0.78	0.85	0.81	0.52	0.68	0.59
LANGUAGE	44	0.82	0.70	0.76	0.95	0.41	0.57
LAW	90	0.67	0.84	0.75	0.65	0.79	0.71
LOC	176	0.70	0.74	0.72	0.66	0.49	0.56
EVENT	550	0.67	0.77	0.71	0.67	0.71	0.69
WEBSITE	116	0.61	0.73	0.67	0.35	0.40	0.37
TIME	84	0.71	0.63	0.67	0.53	0.12	0.19
PRODUCT	17	0.47	0.47	0.47	0.00	0.00	0.00
QUANTITY	9	0.26	0.56	0.36	0.00	0.00	0.00
CURR	8	0.00	0.00	0.00	0.00	0.00	0.00
UNIT	2	0.00	0.00	0.00	0.00	0.00	0.00

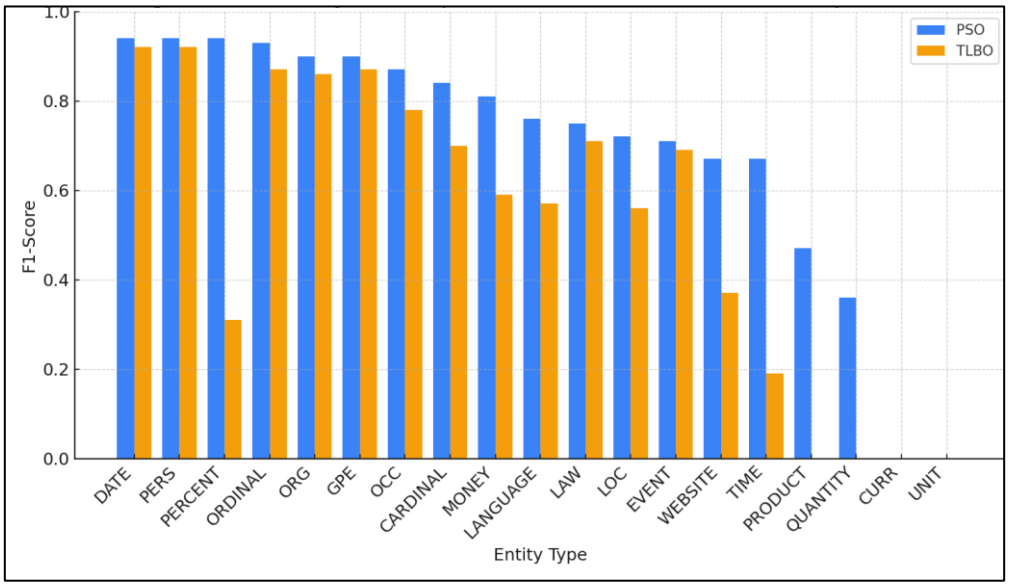

Figure 7. Entity-Level F1 Score Comparison

5.3 Convergence Behavior and Training Stability

Figure 8 represents the training and validation losses developed by the PSO and TLBO optimizers during the process of fine-tuning. Both algorithms exhibit rapid loss reduction in their early stages, but the most significant improvement happens within the first three epochs. PSO converges in a much smoother and more stable manner to reach its minimum validation loss close to Epoch 3 at about 0.035, whereas TLBO converges a bit sooner, around Epoch 2, at approximately 0.038. Though TLBO presents a somewhat faster initial drop, PSO provides overall lower loss values along with greater train-validation consistency, resulting in stronger stability in the optimization processes. Smaller and steadier train-validation gaps under PSO imply better generalization and less overfitting, thus, confirming that PSO attains better convergence reliability across runs. Figure 9 presents the training and validation accuracy curves approximated by the F1-score for both optimization methods. PSO reaches a higher peak in the validation F1 score at 0.8813 compared to TLBO's 0.8755, while it also shows a smoother and more stable improvement trend across epochs. TLBO peaks slightly earlier but exhibits larger fluctuations between epochs, reflecting less consistent convergence. The generalization gap remains modest for both optimizers, indicating that neither of them suffers from severe overfitting during fine-tuning. The conspicuous drop in validation accuracy from ≈0.87 to ≈0.80 around Epoch 6 marks the beginning of overfitting, beyond which further training will not improve generalization. That is, beyond their peaks at Epochs 2–3, both models start memorizing patterns specific to the training data rather than learning widely

transferable representations. This behavior confirms that early stopping around the third epoch provides an optimal trade-off between model fit and generalization stability for both PSO and TLBO.

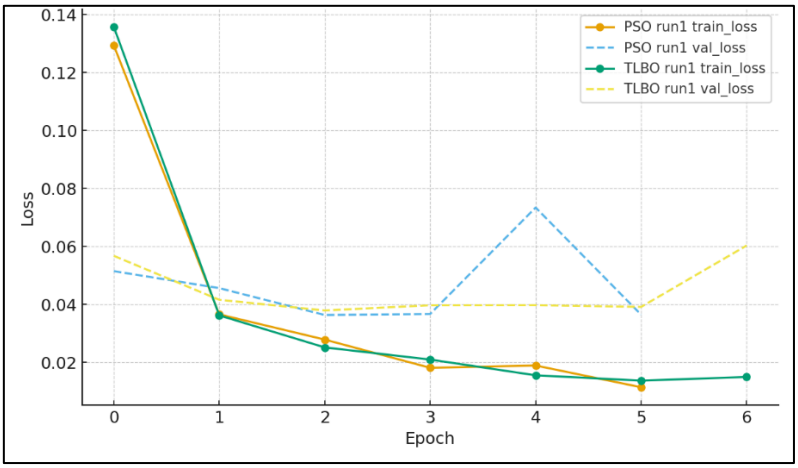

Figure 8. Training and Validation Loss Convergence

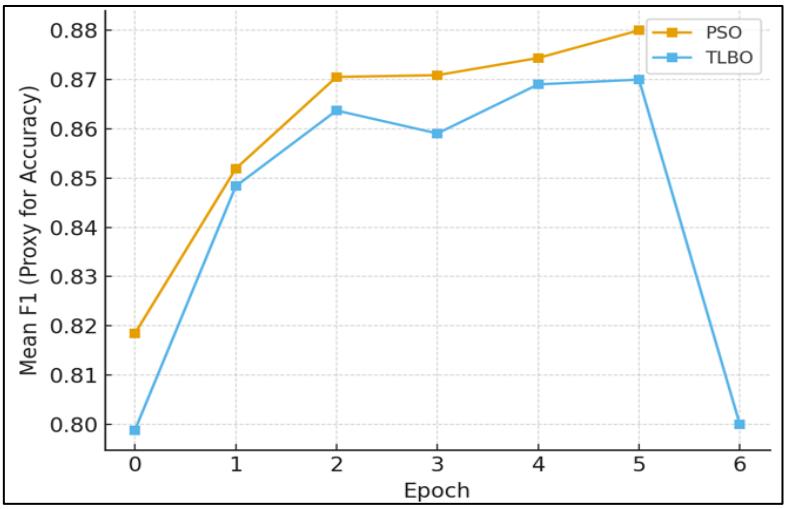

Figure 9. Training and Validation Accuracy (Approximated by F1-Score) for PSO and TLBO

5.4 Comparative Discussion: TLBO vs PSO

TLBO vs. PSO Although PSO and TLBO both attained strong performances in optimizing the NER hyperparameters, distinct characteristics were present in the optimization behavior along with convergence dynamics. PSO is characterized by faster early-stage improvements while exhibiting stronger diversity in search, which consequently allowed it to identify better-performing regions of hyperparameters. Its swarm-based exploration found the

best overall F1-score, 0.8813, slightly superior to TLBO's 0.8755. Moreover, PSO yielded a higher average F1 (0.8708 vs. 0.8561) with lower variation across different runs (std. dev. 0.0164 vs. 0.0286), involving greater stability and robustness in generalization. In contrast, TLBO had smoother and more gradual convergence, with fewer oscillations in training and validation loss. Its teacher-learner mechanism promoted a good balance in exploration and exploitation, which ensured reliable though slightly lower peak performance. The swarm dynamics of PSO encapsulate personal and collective knowledge sharing through an update mechanism of both pbest and gbest. This ensures multi-directional exploration within the search space. Such a twin-update mechanism avoids pre-mature convergence and increases the likelihood of escaping local minima; hence, it explains its superior exploration over TLBO. Although TLBO reached its best performance somewhat earlier at approximately Epoch 2, PSO maintained superior convergence reliability and generalization across later epochs up to Epoch 3, in which it reached its highest validation F1. Task-wise, PSO was always outperforming TLBO in almost all categories of entities, especially the low-support classes such as MONEY, LOC, and WEBSITE, while both performed comparably on high-frequency entities like DATE, PERS, and ORG. These observations confirm that PSO ensures higher final quality of optimization and overall robustness, whereas TLBO offers smoother and more predictable convergence; thus, it is favorable when training stability and reproducibility are a priority.

5.5 Error Patterns and Limitations

Both optimizers struggled with data-scarce entity classes, especially those with fewer than 20 training instances. Entities such as QUANTITY, UNIT, and CURR exhibited near-zero precision and recall, primarily due to extreme class imbalance, the dominance of frequent classes during loss minimization, and tokenization mismatches for rare terms. Improving performance on such categories would likely require data augmentation or class-weighted loss adjustments.

In summary, PSO is recommended as the primary optimization approach for achieving the highest and most consistent F1 performance on Arabic NER, while TLBO remains a competitive alternative for experiments prioritizing smoother convergence and reproducibility. Hyperparameter analysis further indicates that the learning rate and training epochs are the most influential factors for model accuracy, while smaller batch sizes generally yield better

generalization. All figures and tables collectively support these conclusions, clearly differentiating the optimization behavior and performance profiles of PSO and TLBO.

6. Conclusion and Future Work

This paper investigates two metaheuristic optimization algorithms, PSO and TLBO, in fine-tuning transformer-based models for Arabic NER using the Wojood dataset and the aubmindlab/bert-base-arabertv2 model. The obtained results clearly highlight that metaheuristic optimization enhances the performance of the models, with PSO yielding the best F1-scores among the compared algorithms, achieving a maximum test micro-F1 of 0.8813, compared to a score of 0.8755 from TLBO. Moreover, TLBO showed faster early convergence, while PSO was much stronger in terms of convergence stability and had less variability. Due to the unbalanced data, overall performance was weak in classes with low support but relatively good in high-frequency categories. The contribution of this work is in providing a reproducible framework for applying population-based metaheuristics for transformer tuning and outlining potential future research directions such as multi-seed evaluations, data augmentation, and applications on several Arabic dialects. These findings suggest that metaheuristic optimization might improve the effectiveness and precision of training Arabic named entity recognition. In the overall comparison, PSO performed better.

References

[1] Antoun, Wissam, Fady Baly, and Hazem Hajj. "Arabert: Transformer-based model for arabic language understanding." arXiv preprint arXiv:2003.00104 (2020).

[2] Reynolds, Craig W. "Flocks, Herds and Schools: A Distributed Behavioral Model." In Proceedings of the 14th annual conference on Computer graphics and interactive techniques, 25-34. 1987.

[3] Heppner, Frank C., and Ulf Grenander. 1990. “A Stochastic Nonlinear Model for Coordinated Bird Flocks.” In *The Ubiquity of Chaos*, edited by Stanley Krasner, 233–238. Washington, DC: AAAS Publications.

[4] Eberhart, Russell, and James Kennedy. "A New Optimizer Using Particle Swarm Theory." In MHS'95. Proceedings of the sixth international symposium on micro machine and human science, pp. 39-43. Ieee, 1995.

[5] Marini, Federico, and Beata Walczak. "Particle Swarm Optimization (PSO). A Tutorial." Chemometrics and intelligent laboratory systems 149 (2015): 153-165.

[6] Kaveh, Ali, and Taha Bakhshpoori. "Teaching-Learning-Based Optimization Algorithm." In Metaheuristics: Outlines, MATLAB codes and examples, pp. 41-49. Cham: Springer International Publishing, 2019.

[7] Rao, R. Venkata, Vimal J. Savsani, and Dipakkumar P. Vakharia. "Teaching–Learning-Based Optimization: A Novel Method for Constrained Mechanical Design Optimization Problems." Computer-aided design 43, no. 3 (2011): 303-315.

[8] Balla, Husamelddin, and Sarah Jane Delany. "Exploration of Approaches to Arabic Named Entity Recognition." (2020). 2-16.

[9] Muhammad, Marwa, Muhammad Rohaim, Alaa Hamouda, and Salah Abdel-Mageid. "A Comparison between Conditional Random Field and Structured Support Vector Machine for Arabic Named Entity Recognition." Journal of Computer Science 16, no. 1 (2020): 117-125.

[10] Hudhud, Mohammad, Hamed Abdelhaq, Fadi Mohsen, A. Rocha, L. Steels, and J. van den Herik. "ArabiaNer: A System to Extract Named Entities from Arabic Content." In ICAART (1), pp. 489-497. 2021.

[11] Alshammari, Nasser, and Saad Alanazi. "The Impact of Using Different Annotation Schemes on Named Entity Recognition." Egyptian Informatics Journal 22, no. 3 (2021): 295-302.

[12] Elgamal, Marwa, Mohamed Abou-Kreisha, Reda Abo Elezz, and Salwa Hamada. "An Ontology-Based Name Entity Recognition NER and NLP Systems in Arabic Storytelling." Al-Azhar Bulletin of Science 31, no. 2-B (2020): 31-38.

[13] Alkhatib, Manar, and Khaled Shaalan. "Boosting Arabic Named Entity Recognition Transliteration with Deep Learning." In FLAIRS, pp. 484-488. 2020.

[14] Pasha, Arfath, Mohamed Al-Badrashiny, Mona T. Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. "Madamira: A Fast, Comprehensive Tool for Morphological Analysis and Disambiguation of Arabic." In Lrec, vol. 14, no. 2014, pp. 1094-1101. 2014.

[15] Balla, Husamelddin, and Sarah Jane Delany. "Exploration of Approaches to Arabic Named Entity Recognition." (2020).

[16] Helwe, Chadi, and Shady Elbassuoni. "Arabic Named Entity Recognition Via Deep Co-Learning." Artificial Intelligence Review 52, no. 1 (2019): 197-215.

[17] Alkhatib, Manar, and Khaled Shaalan. "Boosting Arabic Named Entity Recognition Transliteration with Deep Learning." In FLAIRS, pp. 484-488. 2020.

[18] Al-Smadi, Mohammad, Saad Al-Zboon, Yaser Jararweh, and Patrick Juola. "Transfer Learning for Arabic Named Entity Recognition with Deep Neural Networks." Ieee Access 8 (2020): 37736-37745.

[19] Antoun, Wissam, Fady Baly, and Hazem Hajj. "Arabert: Transformer-Based Model for Arabic Language Understanding." arXiv preprint arXiv:2003.00104 (2020).

[20] Abdul-Mageed, Muhammad, and AbdelRahim Elmadany. "ARBERT & MARBERT: Deep Bidirectional Transformers for Arabic." In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), 7088-7105. 2021.

[21] Boudjellal, Nada, Huaping Zhang, Asif Khan, Arshad Ahmad, Rashid Naseem, Jianyun Shang, and Lin Dai. "ABioNER: A BERT-Based Model for Arabic Biomedical Named-Entity Recognition." Complexity 2021, no. 1 (2021): 1–6.

[22] Kumar, Sunisth, Davide Liu, and Alexandre Boulenger. "Cross-Lingual NER for Financial Transaction Data in Low-Resource Languages." arXiv preprint arXiv:2307.08714 (2023).

[23] Benajiba, Yassine, and Paolo Rosso. "Arabic Named Entity Recognition Using Conditional Random Fields." In Proc. of Workshop on HLT & NLP within the Arabic World, LREC, vol. 8, 143-153. 2008.

[24] Salah, Ramzi Esmail, and Lailatul Qadri Binti Zakaria. "Building the Classical Arabic named Entity Recognition Corpus (CANERCorpus)." In 2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP), pp. 1-8. IEEE, 2018.

[25] Walker, Christopher, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. 2005. "ACE 2005 Multilingual Training Corpus." *Linguistic Data Consortium*. https://catalog.ldc.upenn.edu/LDC2006T06.

[26] Weischedel, Ralph, M. Palmer, M. Marcus, E. Hovy, S. Pradhan, L. Ramshaw, N. Xue et al. OntoNotes Release 5.0 LDC2013T19. Web Download. Philadelphia: Linguistic Data Consortium, 2013. 2013.

[27] Jarrar, Mustafa. "The Arabic Ontology–An Arabic Wordnet with Ontologically Clean Content." Applied ontology 16, no. 1 (2021): 1-26.

[28] Eiben, Agoston E., and James E. Smith. Introduction to Evolutionary Computing. springer, 2015.

[29] Rajwar, Kanchan, Kusum Deep, and Swagatam Das. "An Exhaustive Review of the Metaheuristic Algorithms for Search and Optimization: Taxonomy, Applications, and Open Challenges." Artificial Intelligence Review 56, no. 11 (2023): 13187-13257.

[30] Khalilia, Mohammed. 2024. *COMP9312: Named Entity Recognition Project* [Computer software]. GitHub. https://github.com/mohammedkhalilia/COMP9312.

[31] Clerc, Maurice, and James Kennedy. "The Particle Swarm-Explosion, Stability, and Convergence in a Multidimensional Complex Space." IEEE transactions on Evolutionary Computation 6, no. 1 (2002): 58-73.