Automated Hyperparameter Scheduling for Distantly Supervised NER using Reinforcement Learning and Bayesian Optimization

Anonymous ACL submission

Abstract

Distant supervision for Named Entity Recognition (NER) produces noisy and incomplete labels that make training highly sensitive to hyperparameter choices. While prior work addresses this through label denoising, the complementary problem of *automated hyperparameter scheduling*—dynamically adjusting hyperparameters during training under weak supervision—remains underexplored. We study Gaussian Process Thompson Sampling (GP-TS) together with completed TD3 and SAC reinforcement-learning baselines for this task within a broader RL scheduling framework. Using BERT-CRF and RoBERTa-CRF backbones under the BOND distant supervision framework, we benchmark on three datasets of varying noise levels (Webpage, Wikigold, CoNLL-2003) against standard schedulers (Step Decay, Cosine Annealing, ReduceLROnPlateau). The verified 10-epoch baselines remain strong: on Webpage, the static RoBERTa-CRF and BERT-CRF baselines reach 0.618 and 0.606 respectively, while on Wikigold the corresponding static scores are 0.468 and 0.521. The recovered 2-epoch optimizer trials are strongly budget-sensitive, but corrected 10-epoch retraining shows that the search phase can understate final performance: for RoBERTa-CRF, GP-TS/TD3/SAC reach 0.610/0.610/0.603 on Webpage and 0.493/0.493/0.459 on Wikigold, while for BERT-CRF they reach 0.512/0.512/0.566 on Webpage and 0.487/0.487/0.490 on Wikigold. Standard schedulers stay close to the static baseline on CoNLL-2003 (0.720–0.734 vs. 0.723), and our findings show that automated scheduling is promising but highly budget- and dataset-sensitive, and that separating short-budget search results from fair final comparisons is essential.

1 Introduction

Named Entity Recognition (NER) identifies and classifies entity mentions in text, but supervised approaches require large, manually annotated corpora that are expensive to produce. *Distant supervision* alleviates this bottleneck by automatically generating labels from knowledge bases or heuristic rules. The resulting annotations, however, are *incomplete* and *noisy*: coverage gaps, alignment mismatches, and ambiguity introduce errors that degrade model performance and destabilize training.

In such noisy settings, hyperparameter choices, including learning rate, weight decay, and optimizer momentum, strongly affect generalization. Static configurations fail to accommodate the non-stationarity of learning under weak supervision, motivating *automated hyperparameter scheduling*: adjusting hyperparameters during training based on model feedback.

Two paradigms address this problem. *Bayesian Optimization* (BO) methods, such as Gaussian Process Thompson Sampling (GP-TS), frame scheduling as global black-box optimization, using surrogate models and acquisition functions to balance exploration and exploitation. *Reinforcement Learning* (RL) methods treat scheduling as a sequential decision process, optimizing policies that maximize long-term validation performance. Each paradigm offers distinct trade-offs: BO is sample-efficient and interpretable but treats trials independently; RL can learn temporal policies but requires multi-epoch trajectories.

To address this gap, we present the first systematic comparison of BO and RL scheduling for distantly supervised NER. We evaluate GP-TS (probabilistic BO), TD3 (off-policy actor-critic), SAC (maximum-entropy RL), TRPO (on-policy constrained policy gradient), DQN (value-based discrete-action RL), and GRPO (Group Relative Policy Optimization), a recent PPO variant that eliminates the critic network by estimating advantages through group-relative comparisons. We apply these methods to dynamically adjust hyperparameters during training, using BERT-CRF

and RoBERTa-CRF as base models, and compare against Step Decay, Cosine Annealing, and ReduceLROnPlateau baselines across three benchmark datasets.

Our experiments on **Webpage**, **Wikigold**, and **CoNLL-2003** demonstrate the trade-offs between different scheduling strategies. On Webpage, the static RoBERTa-CRF and BERT-CRF baselines (10 epochs) reach 0.618 and 0.606 respectively, while standard schedulers remain slightly lower than the RoBERTa static baseline (0.589–0.597). On Wikigold, the static RoBERTa-CRF baseline reaches 0.468, the static BERT-CRF baseline reaches 0.521, and the best recovered 2-epoch optimizer trial is SAC at 0.500 for RoBERTa-CRF. The corrected 10-epoch retraining results show a different picture: for RoBERTa-CRF, GP-TS/TD3/SAC reach 0.610/0.610/0.603 on Webpage and 0.493/0.493/0.459 on Wikigold, while for BERT-CRF they reach 0.512/0.512/0.566 on Webpage and 0.487/0.487/0.490 on Wikigold. Backbone effects remain dataset-dependent: RoBERTa-CRF is stronger on Webpage, whereas BERT-CRF is stronger on Wikigold. CoNLL-2003 optimizer experiments remain incomplete due to computational constraints; we report the verified static and standard scheduler results where available and discuss this limitation.

Contributions:

1. We present a comparative study of GP-TS and RL-based schedulers for dynamic hyperparameter scheduling in distantly supervised NER, with completed experiments for TD3 and SAC against three standard scheduler baselines.

2. We introduce a unified framework for modular integration of scheduling strategies into the NER training loop, supporting both BERT-CRF and RoBERTa-CRF architectures.

3. We show that standard schedulers remain competitive but do not consistently outperform static baselines, and that short-budget search rankings can differ materially from corrected 10-epoch retraining outcomes.

4. We provide analyses of BERT vs. RoBERTa generalization, recovered best hyperparameter configurations, and a comparison with state-of-the-art distantly supervised NER methods.

2 Related Work

2.1 Hyperparameter Optimization

HPO is a core ML challenge. Bayesian optimization with Gaussian Process surrogates (Snoek et al., 2012) matches expert tuning with far fewer evaluations, with modern frameworks like Optuna (Akiba et al., 2019) extending to multi-objective and distributed search. Dynamic adaptation methods like Population-Based Training (PBT) (Jaderberg et al., 2017) and Hyperband (Li et al., 2018) adjust parameters during training rather than seeking static configurations. AutoML for NLP (AutoDistil (Chen et al., 2023), NAS-BERT (Xu et al., 2021), OpenAutoNLU (Arshinov and other, 2025)) focuses on architecture search or static HPO rather than dynamic scheduling during training.

2.2 RL for Hyperparameter Tuning

RL formulates HPO as an MDP: states encode training progress, actions are hyperparameter choices, and reward is validation performance. Early work used Q-learning (Xu and Fern, 2019) and Episodic Policy Gradient Training (Le et al., 2022) for learning rate adjustment. We adopt TD3 (Fujimoto et al., 2018) with twin critics for stability, and additionally evaluate SAC, TRPO, DQN, and GRPO (Shao et al., 2024)—a recent critic-free PPO variant using group-relative advantages. Unlike prior RL-for-HPO work in low-noise settings, we evaluate under the noisy-label conditions of distant supervision.

2.3 Distantly Supervised NER

Distant supervision uses knowledge bases to automatically label NER data (Augenstein et al., 2017; Peng and Dredze, 2019). BOND (Liang et al., 2020) fine-tunes BERT with self-training; RoSTER (Meng et al., 2021), SANTA (Shao et al., 2023), SCDL (Zhang et al., 2022), and CLIM (Wang et al., 2023) improve via label refinement with transformer architectures. SALO (Li et al., 2022) achieves state-of-the-art via self-adaptive label correction. Our work is complementary: rather than improving label quality, we optimize the training schedule. A comparison table is provided in Appendix D.6.

3 Methodology

We design algorithms that *automatically schedule hyperparameters*, specifically the learning rate, for NER models trained on noisy, distantly labeled datasets. Although we focus on learning rate

scheduling, our framework extends to additional hyperparameters. The learning rate η_t at epoch t is dynamically adapted based on feedback from the training trajectory.

We describe two adaptive approaches: (1) a Bayesian optimization method using Gaussian Process Thompson Sampling (GP-TS), and (2) reinforcement learning methods including TD3, DQN, TRPO, SAC, and GRPO. GP-TS treats each adjustment as an independent experiment guided by a surrogate model, while the RL methods continuously update policies conditioned on the evolving training state.

3.1 Overview of Explored Algorithms

GP-TS and the RL algorithms operate within the training loop but differ fundamentally: GP-TS treats each epoch as an independent experiment guided by a Gaussian Process surrogate, while RL methods learn state-conditioned policies from sequential training trajectories. All RL methods share the same state (epoch index, validation F1, training loss), action (learning rate), and reward (ΔF1 per epoch) definitions.

We evaluate six algorithms:

- **GP-TS**: Samples from a GP posterior over $\log_{10}(\eta)$ using a Matérn kernel, selecting the maximizing hyperparameter each epoch. Requires no neural network training.
- **TD3**: Off-policy actor-critic with twin critics and delayed policy updates for continuous LR control, reducing overestimation bias.
- **DQN**: Discretizes the LR into 20 log-uniform bins with ϵ-greedy exploration over a 2-layer MLP (128 units). See Appendix B for full architecture.
- **TRPO**: On-policy policy gradient with a KL-divergence constraint for stable updates.
- **SAC** (Haarnoja et al., 2018): Maximum-entropy RL that jointly optimizes reward and policy entropy for robust exploration.
- **GRPO** (Shao et al., 2024): A critic-free PPO variant that estimates advantages via group-relative comparisons ($G = 8$), reducing model complexity while maintaining stable clipping.

3.2 Gaussian Process-Thompson Sampling (GP-TS)

GP-TS uses a Gaussian Process as a surrogate to model the performance landscape. Each input x_t is the learning rate used at epoch t (optionally paired with t itself). The observed reward r_t is the validation F1 score after epoch t.

At each step, GP-TS samples a function $f \sim P(f|\mathcal{D}_{1:t})$ from the posterior and selects the next learning rate via

$$a_{t+1} = \arg\max_{\eta} f(\eta, t+1).$$

We use a Matérn kernel over $\log_{10}(\eta)$ and a linear trend over epoch index. A noise term accounts for observation variance. GP-TS is computationally efficient, requiring only updates to the GP posterior and an optimization over the sampled function.

Although it does not explicitly track the model's internal state, including the epoch index allows GP-TS to adapt to non-stationarity in training. This approach is conceptually simple, effective in practice, and requires no neural network training.

3.3 Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 is an off-policy actor-critic algorithm for continuous control. We define:

- **State** s_t: Includes normalized epoch index (t/T), validation F1 score, and training loss. Optionally, it may include additional features such as gradient variance.
- **Action** a_t: A real-valued learning rate selected from a constrained range, typically $[10^{-5}, 10^{-2}]$.
- **Reward** r_t: For $t < T$, we define $r_t = \text{F1}_t - \text{F1}_{t-1}$. At $t = T$, the final reward includes the overall validation F1 score.

The transition from s_t to s_{t+1} is determined by one training epoch. The actor $\pi_\theta(s_t)$ outputs the learning rate a_t, while twin critics Q_{ϕ_1}, Q_{ϕ_2} estimate the expected return. The critic loss is minimized as:

$$L(\phi_i) = \mathbb{E}\left[(Q_{\phi_i}(s_t, a_t) - y_t)^2\right],$$
$$y_t = r_t + \gamma \min_i Q_{\phi'_i}(s_{t+1}, \pi_{\theta'}(s_{t+1})),$$

where θ', ϕ' denote target networks. The actor is updated less frequently (delayed policy updates),

and exploration noise is added to encourage policy diversity.

Due to limited episodes, we adopt an *online* strategy: the replay buffer collects transitions from a single training run, and multiple gradient steps are applied per epoch. Exploration noise decays over time to ensure convergence. TD3 enables the agent to learn hyperparameter scheduling policies conditioned on training dynamics. While TD3 adds moderate computation overhead, its capacity for continuous action space exploration makes it a promising candidate for adaptive scheduling in noisy NER tasks.

3.4 Other RL Algorithms: DQN, TRPO, SAC, and GRPO

All four algorithms share the MDP formulation defined for TD3 but differ in their optimization strategies. **DQN** approximates $Q^*(s, a)$ via a 2-layer MLP (128 units) over 20 log-uniform LR bins; ϵ-greedy exploration decays from 1.0 to 0.01. **TRPO** maximizes a clipped surrogate objective under KL $\leq \delta$ for stable on-policy updates. **SAC** maximizes a weighted sum of expected return and policy entropy H, promoting diverse exploration. **GRPO** eliminates the critic by computing group-relative advantages ($G = 8$) with a PPO-style clipped objective. Full architectures and hyperparameters are provided in Appendix B.

Group Relative Policy Optimization (GRPO): GRPO is a recent variant of PPO that removes the need for a separate value (critic) network. Instead of estimating advantages as $A_t = R_t - V(s_t)$, GRPO samples a group of G configurations from the current policy at each iteration, evaluates their rewards, and computes group-relative advantages:

$$\hat{A}_i = \frac{r_i - \mu_r}{\sigma_r + \epsilon},$$

where μ_r and σ_r are the mean and standard deviation of rewards within the group, and ϵ is a small constant for numerical stability. The policy is then updated using the standard PPO clipped objective with these group-relative advantages. By eliminating the critic, GRPO reduces model complexity and memory requirements while maintaining stable policy updates, making it particularly suitable for hyperparameter optimization tasks with well-defined reward signals such as validation F1 scores.

Algorithm 1 presents the GRPO scheduling procedure in pseudocode.

Algorithm 1 GRPO Hyperparameter Scheduling

Require: Initial policy π_{θ_0}, group size G, clip ratio ϵ_{clip}, learning rate bounds $[\eta_{\min}, \eta_{\max}]$, number of iterations K

1: **for** iteration $k = 1, \ldots, K$ **do**
2: Sample G learning rate configurations $\{\eta_i\}_{i=1}^{G}$ from π_{θ_k}
3: Train NER model for one epoch with each η_i; collect rewards $\{r_i\}_{i=1}^{G}$ (validation F_1)
4: Compute group-relative advantages: $\hat{A}_i = (r_i - \mu_r)/(\sigma_r + \epsilon)$
5: Update policy via PPO clipped objective, where

$$r_i(\theta) = \frac{\pi_\theta(\eta_i \mid s_i)}{\pi_{\theta_{\text{old}}}(\eta_i \mid s_i)}$$

6: and

$$\mathcal{L}(\theta) = \mathbb{E}_i\Big[\min\big(r_i(\theta)\hat{A}_i,\ \text{clip}(r_i(\theta), 1 \pm \epsilon_{\text{clip}})\hat{A}_i\big)\Big]$$

7: $\theta_{k+1} \leftarrow \theta_k - \alpha \nabla_\theta \mathcal{L}(\theta)$
8: **end for**
9: **return** Best learning rate configuration found across all iterations

4 Experimental Setup and Results

We evaluate our hyperparameter scheduling strategies, including GP-TS and completed TD3/SAC deep RL baselines, on three benchmark datasets under distantly supervised NER. We assess whether dynamic scheduling outperforms both fixed hyperparameter configurations and standard learning rate schedules in terms of NER accuracy and generalization.

4.1 Datasets and Training Baseline

We conduct experiments on three datasets:

- **Webpage**: 20 web documents, 783 entities. High-noise heuristic labels from Wikipedia matching.
- **Wikigold**: ∼40K tokens from Wikipedia, 4 entity types. Moderate noise from hyperlink alignment.
- **CoNLL-2003**: ∼203K training tokens in news domain. Gold labels used only for evaluation; training labels via knowledge-base alignment.

For all datasets, we adopt RoBERTa-CRF (`roberta-base`) (Liu et al., 2019) as primary NER model, with BERT-CRF (`bert-base-cased`) (Devlin et al., 2019) for generalization. Both use BOND (Liang et al., 2020) with AdamW. The static

baseline uses a fixed learning rate of 1×10^{-5} and weight decay of 1×10^{-4}. Scheduling agents tune six hyperparameters: learning rate, weight decay, Adam β_1, β_2, warmup steps, and batch size. We compare against three standard schedulers: Step Decay ($\gamma = 0.5$ per 10 epochs), Cosine Annealing (Loshchilov and Hutter, 2017) ($T_{\max} = 50$, $\eta_{\min} = 10^{-7}$), and ReduceLROnPlateau.

The scheduling framework implements GP-TS (20 LR candidates), TD3 (continuous control), DQN (20 discrete bins), TRPO (KL-constrained policy gradient), SAC (maximum-entropy), and GRPO (critic-free, $G = 8$). The completed quantitative results reported in this paper cover GP-TS, TD3, and SAC; DQN, TRPO, and GRPO remain part of the framework but are not included in the final quantitative comparison. Full search spaces are in Appendix A; implementation details in Appendix B.

Each method is run with three seeds (42, 123, 456). For Webpage and Wikigold, 20% of training data is held out for validation. The primary metric is entity-level micro-averaged F_1. All experiments use 2-epoch per-trial budgets for optimizer methods and 10-epoch full training for static/scheduler baselines. Early stopping is applied with patience 3 on validation F_1.

4.2 Results on Webpage Dataset

Table 1 reports the verified 10-epoch baselines and standard schedulers, Table 2 reports the best recovered 2-epoch optimizer trials, and Table 3 reports the corrected 10-epoch retraining results. On the Webpage dataset, the static RoBERTa-CRF baseline (10 epochs) achieves a best checkpoint F_1 of 0.618, while the BERT-CRF static baseline reaches 0.606. Standard learning-rate schedulers remain below the RoBERTa static baseline on this dataset, with Step Decay, Cosine Annealing, and ReduceLROnPlateau reaching 0.593, 0.597, and 0.589 respectively.

For optimizer methods under a 2-epoch per-trial budget, SAC achieves the strongest recovered search-phase result on Webpage at $F_1 = 0.290$, while GP-TS and TD3 reach 0.094 and 0.114 respectively. However, the corrected 10-epoch retraining results change the fair-comparison picture: for RoBERTa-CRF, GP-TS and TD3 both rise to 0.610 and SAC reaches 0.603, while for BERT-CRF, GP-TS and TD3 reach 0.512 and SAC reaches 0.566. Among the 2-epoch optimizer methods, the BERT-CRF variants outperform RoBERTa-CRF for GP-TS, TD3, and SAC, with BERT SAC reaching 0.365.

Key findings on Webpage: (1) The static RoBERTa-CRF baseline slightly exceeds the static BERT-CRF baseline. (2) Standard schedulers do not improve over the static RoBERTa baseline on this dataset. (3) The 2-epoch search-phase numbers substantially underestimate final potential: although SAC is the strongest recovered short-budget trial, the corrected 10-epoch retraining results bring RoBERTa GP-TS and TD3 up to 0.610 and BERT SAC up to 0.566, much closer to the static baselines than the search-phase scores suggest. Optimization trajectories are shown in Appendix D.

4.3 Results on Wikigold Dataset

On Wikigold, the static RoBERTa-CRF baseline (10 epochs) achieves a best checkpoint F_1 of 0.468, while the static BERT-CRF baseline reaches 0.521. Standard schedulers produce slightly higher results (0.474–0.483), with cosine annealing the strongest standard scheduler. For optimizer methods under a 2-epoch per-trial budget, SAC achieves the best recovered RoBERTa-CRF search-phase result at $F_1 = 0.500$, followed by TD3 at 0.433 and GP-TS at 0.388. In the corrected 10-epoch retraining results, RoBERTa GP-TS and TD3 both reach 0.493 while RoBERTa SAC reaches 0.459; the corresponding BERT corrected-final values are 0.487, 0.487, and 0.490. BERT-CRF remains stronger on Wikigold overall, with BERT SAC reaching 0.506 in the recovered search phase and 0.490 after corrected retraining.

Key findings on Wikigold: (1) Standard schedulers modestly improve over the static RoBERTa baseline. (2) SAC is the strongest recovered 2-epoch optimizer trial, but the corrected 10-epoch retraining results raise RoBERTa GP-TS and TD3 to 0.493, slightly above the static RoBERTa baseline of 0.468, while all BERT corrected-final runs remain below the stronger BERT static baseline of 0.521. (3) BERT-CRF outperforms RoBERTa-CRF across the completed Wikigold runs.

4.4 Results on CoNLL-2003 Dataset

On CoNLL-2003, the largest and cleanest benchmark, the static RoBERTa-CRF baseline (10 epochs) achieves a best checkpoint F_1 of 0.723—consistent with BOND Stage I (75.61% under full self-training), where our result reflects only the initial fine-tuning stage without self-training iterations. Standard schedulers produce similar re-

Method	Webpage	Wikigold	CoNLL-03
RoBERTa-CRF full-training baselines (10 ep)			
Static baseline	0.618	0.468	0.723
Step Decay	0.593	0.474	0.720
Cosine Annealing	0.597	0.483	0.734†
Reduce LR on Plateau	0.589	0.474	0.721
BERT-CRF full-training baselines (10 ep)			
BERT static	0.606	0.521	[TBD]

Table 1: Verified 10-epoch full-training baselines and standard schedulers (entity-level micro-averaged best checkpoint F_1). These rows are the fair comparison target for future 10-epoch optimizer retraining. [TBD] indicates results unavailable or unverified. †Seed 42 data unavailable for this cosine cell due to checkpoint loss; seed 123 value is reported.

Method	Webpage	Wikigold	CoNLL-03
RoBERTa-CRF optimizer search (best 2-ep trial, seed=42)			
GP-TS	0.094	0.388	[TBD]
TD3	0.114	0.433	[TBD]
SAC	0.290	0.500	[TBD]
BERT-CRF optimizer search (best 2-ep trial, seed=42)			
BERT GP-TS	0.167	0.408	[TBD]
BERT TD3	0.236	0.441	[TBD]
BERT SAC	0.365	0.506	[TBD]

Table 2: Best recovered 2-epoch optimizer trial results (entity-level micro-averaged best checkpoint F_1). These values summarize the search phase only and should not be interpreted as fair 10-epoch comparisons against the baselines in Table 1. [TBD] indicates unavailable or unverified cells, including the unfinished CoNLL-2003 optimizer runs.

Method	Webpage	Wikigold	CoNLL-03
RoBERTa-CRF corrected final retraining (10 ep, seed=42)			
GP-TS retrain	0.610	0.493	[TBD]
TD3 retrain	0.610	0.493	[TBD]
SAC retrain	0.603	0.459	[TBD]
BERT-CRF corrected final retraining (10 ep, seed=42)			
BERT GP-TS retrain	0.512	0.487	[TBD]
BERT TD3 retrain	0.512	0.487	[TBD]
BERT SAC retrain	0.566	0.490	[TBD]

Table 3: Corrected 10-epoch retraining results from optimizer-selected configurations. RoBERTa-CRF and BERT-CRF rows are complete for Webpage and Wikigold; CoNLL-2003 remains [TBD]. Unlike Table 2, these rows are directly comparable to the 10-epoch baselines in Table 1.

sults: Step Decay at 0.720, Cosine Annealing at 0.734†, and ReduceLROnPlateau at 0.721. The corresponding BERT-CRF static run did not complete successfully, so we do not report a verified BERT CoNLL-2003 baseline here.

Due to the large size of CoNLL-2003 (14K sentences), each optimizer cell requires 8+ hours. We completed partial optimization runs: GP-TS reached training F_1 of 0.731 (seed 42, 7 trials), and TD3 achieved 0.750 (seed 42, 15 trials) on 2-epoch per-trial budgets. However, final model training and full test-set evaluation of the best configurations could not be completed under our computational budget. We therefore report the available partial results here and acknowledge this as a limitation. Full CoNLL-2003 optimizer results are deferred to future work.

4.5 Summary of Results

Table 1, Table 2, and Table 3 separate the three evidence sources used in this work: 10-epoch full-training baselines, best 2-epoch optimizer trials, and corrected 10-epoch retraining results for the completed optimizer cells.

4.6 Cross-Dataset Analysis

The available results reveal strong dataset dependence, but they must now be read with the evaluation budget made explicit. Table 1 shows that the verified 10-epoch baselines remain strong on all datasets. Table 2 shows that the short 2-epoch optimizer trials are far more competitive on Wikigold than on Webpage: SAC reaches 0.500 on Wikigold but only 0.290 on Webpage. Table 3 then shows that search-phase scores can substantially understate final achievable performance: GP-TS and TD3 both rise from 0.094/0.114 to 0.610 on Webpage and from 0.388/0.433 to 0.493 on Wikigold, while SAC rises from 0.290 to 0.603 on Webpage but falls from 0.500 to 0.459 on Wikigold. On CoNLL-2003, the standard schedulers stay close to the static baseline (0.720–0.734 vs. 0.723), but optimizer comparisons remain incomplete. The current evidence therefore supports a strict separation between short-budget search-phase comparisons and fair final comparisons against 10-epoch baselines.

4.7 Training Cost and Efficiency

Training costs vary substantially across methods. On an RTX 4060 (8GB), static and standard scheduler baselines complete within 4 minutes per cell, GP-TS and TD3 require approximately 17 minutes (20 trials), and SAC requires roughly 42 minutes (50 trials). On an RTX 5090 (32GB), corresponding times are 2, 5, and 12 minutes respectively.

The 5090 achieves approximately 3–4× speedup. BO/RL methods represent a one-time optimization cost; once the best hyperparameters are identified, they can be applied to downstream training without overhead. The full per-method breakdown is provided in Appendix D.4.

4.8 Comparison with State-of-the-Art

We compare with published distantly supervised NER methods (BOND, RoSTER, SCDL, SANTA, CLIM) in Appendix D.5. Our verified RoBERTa-CRF static baselines (0.618 on Webpage, 0.468 on Wikigold, 0.723 on CoNLL-2003) and BERT-CRF baselines (0.606 on Webpage, 0.521 on Wikigold) show that backbone effects are dataset-dependent. The remaining gap to SOTA methods such as SANTA also indicates that scheduling alone cannot compensate for label noise, reinforcing its role as a complementary optimization axis.

4.9 Ablation: Best GP-TS Configurations

GP-TS best-configuration analysis (Appendix D.7) reveals that both recovered seed-42 runs favor the same moderate learning rate and non-zero warmup: Webpage and Wikigold both select 2.26×10^{-5} with 65 warmup steps. Relative to the static baseline (1×10^{-5}, zero warmup), these runs suggest that a modestly larger learning rate together with gradual ramp-up is beneficial for the completed GP-TS searches.

The recovered GP-TS configurations also show that the best Webpage and Wikigold runs share the same weight decay (6.09×10^{-4}), Adam coefficients ($\beta_1 = 0.817$, $\beta_2 = 0.971$), and batch size (10). This consistency should be interpreted cautiously because it is based on the recovered seed-42 best trials only, but it still indicates that non-zero warmup and a moderately aggressive learning rate are recurring features of the best GP-TS runs available in the current result files.

Figure 1 visualizes the parameter relationships discovered by GP-TS, showing how learning rate interacts with other hyperparameters in determining validation performance.

5 Comparative Analysis of Completed Optimizer Results

We now compare the completed optimizer results across optimization paradigm, exploration strategy, and empirical performance.

Optimization paradigm. GP-TS treats scheduling as a contextual bandit using a GP surrogate with

Figure 1: GP-TS parameter relationship visualization on the Webpage dataset. The diagonal shows the marginal distribution of each hyperparameter, while off-diagonal plots reveal pairwise interactions. Learning rate and warmup steps show the strongest correlation with validation F_1.

Thompson Sampling, requiring no neural network training. TD3 and SAC formulate scheduling as an MDP with learned actor-critic policies, enabling long-horizon trade-offs. The broader framework also implements TRPO and GRPO, but completed quantitative result grids are not available for them in the recovered files.

Exploration. GP-TS explores via posterior sampling, naturally balancing exploration and exploitation. TD3 injects decaying Gaussian noise into actions. SAC jointly maximizes reward and entropy for robust exploration. In the recovered result files, GP-TS reaches its best Webpage score within the available 20-trial budget, while TD3 shows substantially higher variance across trials on both Webpage and Wikigold.

Empirical results. In the recovered 2-epoch search phase, GP-TS reaches 0.094 and TD3 0.114 on Webpage, while SAC is the strongest completed optimizer at 0.290; on Wikigold, GP-TS reaches 0.388, TD3 0.433, and SAC 0.500. However, the corrected 10-epoch retraining results show a different ranking for the fair-comparison setting: with RoBERTa-CRF seed 42, GP-TS/TD3/SAC reach 0.610/0.610/0.603 on Webpage and 0.493/0.493/0.459 on Wikigold, while with BERT-CRF they reach 0.512/0.512/0.566 on Webpage and 0.487/0.487/0.490 on Wikigold. These corrected results indicate that short-budget search-phase scores should not be interpreted as

final model quality.

Trade-offs. GP-TS is interpretable, sample-efficient, and computationally light, but the corrected 10-epoch results show that its final ranking depends on both backbone and dataset: GP-TS and TD3 lead the completed RoBERTa-CRF retraining runs, whereas SAC is the strongest completed BERT-CRF retraining run on both Webpage and Wikigold. TD3 and SAC scale more naturally to multi-parameter tuning through neural approximation, at the cost of additional hyperparameters and higher variance. The current evidence therefore favors budget-aware method comparisons rather than a single global winner.

5.1 Summary

GP-TS is a transparent and computationally light optimizer, but its apparent performance depends strongly on whether one looks at the 2-epoch search phase or the corrected 10-epoch retraining stage. SAC delivers the strongest recovered short-budget trial results overall, especially on Wikigold, while the corrected final runs show that GP-TS and TD3 can recover much stronger full-training performance than their search-phase scores suggest and that BERT SAC can substantially improve over its own short-budget Webpage search score after full retraining. Future systems may still combine BO and RL ideas, but the current evidence supports only cautious, budget-aware comparisons.

6 Conclusion and Future Work

We compared GP-TS with the completed TD3 and SAC reinforcement-learning baselines for automated hyperparameter scheduling in distantly supervised NER. The current evidence shows that adaptive scheduling is highly budget- and dataset-dependent rather than uniformly beneficial. In the recovered 2-epoch search phase, the static RoBERTa baseline on Webpage (0.618) remains much stronger than the optimizer trials, whereas SAC reaches 0.500 on Wikigold and exceeds the static RoBERTa baseline of 0.468. But the corrected 10-epoch retraining results show that RoBERTa GP-TS and TD3 recover to 0.610 on Webpage and 0.493 on Wikigold, RoBERTa SAC reaches 0.603 on Webpage and 0.459 on Wikigold, and BERT SAC reaches 0.566 on Webpage and 0.490 on Wikigold. Standard schedulers remain close to the static baseline on CoNLL-2003 (0.720–0.734 vs. 0.723), and backbone choice also depends on the dataset: RoBERTa slightly exceeds BERT on Webpage (0.618 vs. 0.606), while BERT is stronger on Wikigold (0.521 vs. 0.468). Overall, the current evidence suggests that short 2-epoch optimizer budgets are useful for search but insufficient as a substitute for fair final evaluation.

Future directions include: multi-epoch trial budgets for RL, multi-hyperparameter scheduling, transferable policies, theoretical analysis of scheduling under label noise, and hybrid BO-RL approaches. The full experiment suite and anonymous code package are available for reproducibility.

7 Limitations

We acknowledge the following specific limitations:

Short trial budget. Our current optimizer protocol uses only 2 epochs per trial. This still limits RL agents' ability to learn richer temporal scheduling policies, since each trial exposes only a short training trajectory. GP-TS, which treats each trial more independently, is less affected by this constraint. Our optimizer results should therefore be interpreted as short-budget outcomes rather than definitive upper bounds.

Hardware constraints. All scheduler and static baseline experiments were conducted on NVIDIA RTX 4060 (8GB VRAM) and RTX 5090 (32GB VRAM) GPUs, as detailed in Section 4.7. We were limited to `roberta-base` and `bert-base-cased` models due to VRAM constraints. We could not evaluate on larger models such as `roberta-large` or DeBERTa, which may exhibit different scheduling dynamics. Scalability to multi-GPU or distributed settings remains untested.

Limited dataset coverage. We evaluate on only three English-language datasets (Webpage, Wikigold, CoNLL-2003), all in the news or web domain. Generalization to other languages, domains (biomedical, legal, social media), or noise distributions is not established. The Webpage dataset is particularly small (783 entities), which may inflate variance.

Learning rate scheduling only. Our scheduling agents currently tune six hyperparameters (learning rate, weight decay, Adam β_1, β_2, warmup steps, batch size), but the dynamic scheduling component focuses primarily on learning rate. Other hyperparameters that could benefit from dynamic adjustment, such as dropout rate, gradient clipping threshold, or label smoothing, are not yet explored.

No comparison with learned schedulers from AutoML. We compare against standard hand-designed schedulers (Step Decay, Cosine Annealing, ReduceLROnPlateau) but not against learned schedulers from the AutoML literature such as Hyperband (Li et al., 2018), BOHB (Falkner et al., 2018), or PBT (Jaderberg et al., 2017). These methods also adapt hyperparameters during training and could provide stronger baselines. Incorporating such comparisons is a priority for future work.

Partial experimental coverage. We report completed Webpage and Wikigold results for GP-TS, TD3, and SAC with both RoBERTa-CRF and BERT-CRF backbones. However, CoNLL-2003 optimizer experiments remain incomplete due to computational constraints: each optimizer cell requires 8+ hours on a single GPU owing to the dataset's size (14K sentences, ∼203K training tokens). We also do not have completed quantitative result grids for DQN, TRPO, or GRPO in the recovered files. This restricts our ability to draw cross-dataset conclusions about the full scheduling-method spectrum.

8 Ethical Considerations

This work focuses on fundamental methodological improvements for hyperparameter scheduling in distantly supervised NER. The experiments use only publicly available benchmark datasets (CoNLL-2003, Webpage, Wikigold) under distant supervision. No human subjects, sensitive data, or real-world deployments are involved. Computational experiments were conducted on NVIDIA RTX 4060 (8GB VRAM) and RTX 5090 (32GB VRAM) GPUs; we acknowledge the environmental impact of GPU computation and provide a detailed training cost analysis in Section 4.7. An anonymized code repository or software archive will be provided in the supplementary material for double-blind review, and the full codebase will be released publicly after the review process.

Code Availability

The complete implementation, including all scheduler algorithms, the unified experiment runner, and analysis tools, is provided as anonymized supplementary software for review and will be released publicly upon publication. The anonymous code package includes the experiment orchestrator (`experiment_runner.py`), shared optimizer utilities (`optimizer_runtime.py`), final model training script (`train_final_models.py`), and configuration files for all experimental matrices.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Grigory Arshinov and other. 2025. Openautonlu: Open source automl library for nlu. In *arXiv preprint arXiv:2603.01824*.

Isabelle Augenstein and 1 others. 2017. Distantly supervised learning for emerging entity classification. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jiale Chen and 1 others. 2023. Autodistil: An end-to-end framework to explore and distill knowledge from large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. Bohb: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning (ICML)*.

Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*.

Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, and 1 others. 2017. Population based training of neural networks. In *arXiv preprint arXiv:1711.09846*.

Thai Le and 1 others. 2022. Episodic policy gradient training. In *Advances in Neural Information Processing Systems*.

Jiacheng Li, Aixin Zhang, and 1 others. 2022. Salo: Self-adaptive label optimization for noisy named entity recognition. In *Findings of the Association for Computational Linguistics (ACL)*.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. Hyperband: A novel bandit-based approach to hyperparameter optimization. In *Journal of Machine Learning Research*, volume 18, pages 1–52.

Chen Liang, Yue Yu, Haoming Jiang, Siawpeng Er, Ruijia Wang, Tuo Zhao, and Chao Zhang. 2020. Bond: Bert-assisted open-domain named entity recognition with distant supervision. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. In *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*.

Yu Meng, Yunyi Zhang, Jiaxin Huang, Chenyan Xiong, Heng Ji, Chao Zhang, and Jiawei Han. 2021. Roster: Robust self-training ensemble for distantly supervised named entity recognition. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Nanyun Peng and Mark Dredze. 2019. Distantly supervised named entity recognition using automatic noise estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Shuhuai Shao, Tian Li, Zhen Huang, and Yongbin Zhang. 2023. Santa: Separate strategies for inaccurate and incomplete annotation noise in distantly-supervised named entity recognition. In *Findings of the Association for Computational Linguistics (ACL)*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Yanda Li, Yu Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. In *arXiv preprint arXiv:2402.03300*.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25.

Zhi Wang and 1 others. 2023. A class-rebalancing self-training framework for distantly-supervised named entity recognition. In *Findings of the Association for Computational Linguistics (ACL)*.

Can Xu and 1 others. 2021. Nas-bert: Neural architecture search for bert compression. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Honglin Xu and Alan Fern. 2019. Learning rate adaptation with reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Xinghua Zhang and 1 others. 2022. Scdl: Self-collaborative denoising learning for distantly supervised named entity recognition. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*.

A Full Hyperparameter Search Spaces

Table 4 provides the complete hyperparameter search spaces for all scheduling algorithms.

For GP-TS, the search space is discretized into 20 candidate learning rates sampled log-uniformly from $[10^{-6}, 10^{-4}]$. For DQN, the action space consists of 20 discrete bins over the same range. TD3, SAC, and TRPO operate over the continuous range directly. GRPO samples $G = 8$ configurations per iteration from the policy distribution.

Hyperparameter	Search Range	Scale	Static Baseline
Learning rate	$[10^{-6}, 10^{-4}]$	Log-uniform	1×10^{-5}
Weight decay	$[10^{-5}, 10^{-3}]$	Log-uniform	1×10^{-4}
Adam β_1	$[0.8, 0.99]$	Uniform	0.9
Adam β_2	$[0.95, 0.999]$	Uniform	0.999
Warmup steps	$[50, 500]$	Integer	0
Batch size	$[8, 32]$	Integer	16

Table 4: Hyperparameter search spaces for all scheduling algorithms. The "Scale" column indicates whether the parameter is sampled on a log or linear scale. The static baseline values are used for the fixed-configuration experiments.

B Implementation Details

B.1 Neural Network Architectures

TD3 Actor-Critic Networks. The actor network consists of two hidden layers with 256 units each, ReLU activations, and a sigmoid output scaled to the learning rate range $[10^{-6}, 10^{-4}]$. Each critic network has the same architecture with an additional action input concatenated at the first hidden layer. Target networks use soft updates with $\tau = 0.005$.

DQN Network. The Q-network uses two hidden layers with 128 units each, ReLU activations, and 20 output units corresponding to discretized learning rate bins. A replay buffer of size 1000 stores transitions, with a batch size of 32 for training. Epsilon-greedy exploration decays from 1.0 to 0.01 over the first 50% of training steps.

TRPO Policy Network. The policy network uses two hidden layers with 64 units each, tanh activations, and a Gaussian output distribution. The value function has the same architecture. Conjugate gradient steps are limited to 10 iterations, with a KL divergence constraint of $\delta = 0.01$.

SAC Networks. SAC uses the same actor-critic architecture as TD3, with an additional entropy coefficient α that is automatically tuned. The target entropy is set to the negative dimension of the action space.

GRPO Policy Network. GRPO uses a single policy network with two hidden layers of 64 units each and tanh activations. The group size is $G = 8$, and the PPO clip ratio is $\epsilon_{\text{clip}} = 0.2$. No critic network is needed.

B.2 Training Hyperparameters for RL Agents

Table 5 summarizes the training hyperparameters for all RL agents.

Hyperparameter	TD3	DQN	TRPO	SAC	GRPO
Replay buffer size	1000	1000	–	1000	–
Batch size	64	32	64	64	8 (group)
Discount γ	0.99	0.99	0.99	0.99	–
Learning rate (agent)	3×10^{-4}	1×10^{-3}	–	3×10^{-4}	3×10^{-4}
Target update τ	0.005	0.005	–	0.005	–
Policy delay	2	–	–	–	–
Noise std (exploration)	0.1	–	–	Auto	0.1
KL constraint δ	–	–	0.01	–	–
Clip ratio ϵ	–	–	–	–	0.2

Table 5: Training hyperparameters for all RL scheduling agents. Dashes indicate parameters not applicable to a given method.

C Dataset Statistics

Table 6 provides detailed statistics for the three datasets used in our experiments.

The noise characteristics differ substantially across datasets. Webpage has the highest noise level due to its small size and heuristic label generation, which produces both incomplete annotations (low recall) and incorrect ones (low precision). Wikigold has moderate noise from Wikipedia hyperlink alignment, which provides reasonable entity boundaries but misses entities not linked in the source. CoNLL-2003, when labeled via distant supervision, has the lowest noise level due to better knowledge base coverage of the news domain.

Statistic	Webpage	Wikigold	CoNLL-03
Domain	Web pages	Wikipedia	News
Entity types	4	4	4
Train tokens	∼2.5K	∼40K	∼203K
Dev tokens	∼0.5K	∼8K	∼51K
Test tokens	∼1.2K	∼8K	∼46K
Train entities	783	∼5.5K	∼23.5K
Noise level	High	Moderate	Low (distant)
Label source	Wikipedia heuristics	Hyperlink + gazetteer	KB alignment

Table 6: Dataset statistics for the three benchmark datasets. "Noise level" characterizes the quality of distantly supervised labels relative to gold annotations. CoNLL-2003 uses gold labels only for evaluation; training labels are generated via distant supervision.

D Additional Experimental Results

D.1 Per-Epoch F1 Curves

Figure 2 shows the per-epoch validation F_1 curves for the static baseline and GP-TS on the Webpage dataset. In the recovered search-phase files, the completed 2-epoch GP-TS trials remain below the static baseline on Webpage, so the figure should be interpreted as an illustration of short-budget search dynamics rather than as a direct summary of the corrected 10-epoch retraining results.

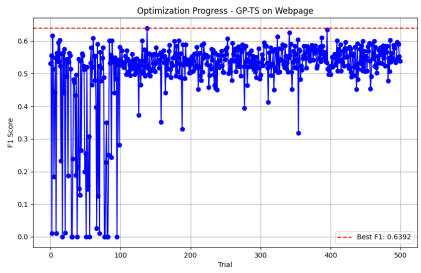

Figure 2: Validation F_1 curves on the Webpage dataset. GP-TS explores diverse learning rate configurations across trials under the 2-epoch search budget.

D.2 Convergence Analysis

We analyze the convergence behavior of each scheduling method by examining how quickly the recovered search trajectories approach their best observed F_1 values. In the available 20-trial Webpage GP-TS search, the best score is found early, whereas the TD3 runs on Wikigold continue to show noticeable variance across trials. The recovered files therefore suggest that optimizer stability remains a central issue under short 2-epoch budgets.

Longer per-trial budgets remain a natural next step, since they would allow RL methods to observe richer training trajectories within each trial. However, they would also raise wall-clock cost substantially, creating a trade-off between trajectory quality and total search time.

D.3 SAC and TRPO Optimization Trajectories

Figure 3 shows an exploratory TRPO trajectory on Wikigold, and Figure 4 shows the SAC optimization trajectory on Webpage. These plots are included as qualitative illustrations of optimization behavior rather than as part of the final quantitative comparison, which reports completed GP-TS, TD3, and SAC result grids only.

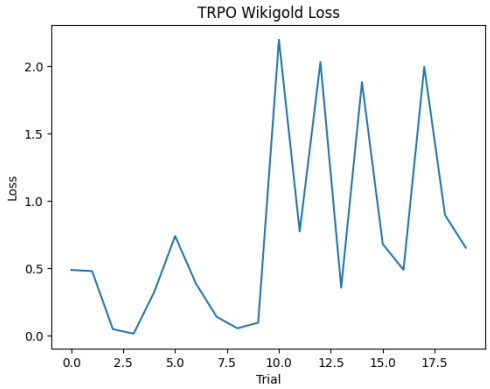

Figure 3: TRPO optimization progress on the Wikigold dataset. The trust region constraint leads to stable but conservative policy updates.

D.4 Training Cost Breakdown

Table 7 summarizes the wall-clock cost of each method on the two GPU settings used in this work. The comparison highlights that BO/RL scheduling incurs a moderate one-time search cost, but the 5090 reduces this overhead substantially relative to the 4060.

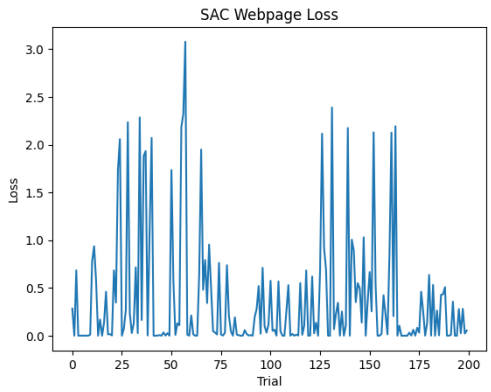

Figure 4: SAC optimization progress on the Webpage dataset. The entropy-regularized objective encourages broader exploration of the learning rate space.

D.5 State-of-the-Art Comparison

Table 8 situates our results against representative distantly supervised NER systems. These methods primarily improve label quality or self-training, whereas our contribution isolates scheduling as a complementary optimization axis under the same weak-supervision setting.

D.6 Related Work Comparison

Table 9 summarizes the main methodological differences between prior denoising-oriented approaches and our scheduling-based framework. The key distinction is that our method leaves the noisy-label generation pipeline unchanged and instead optimizes the training dynamics of the underlying NER model.

D.7 GP-TS Best Configurations

Table 10 reports the best recovered GP-TS configurations on Webpage and Wikigold. In the available seed-42 runs, both datasets select the same configuration, indicating that the best recovered GP-TS solution combines a moderately larger learning rate than the static baseline with non-zero warmup.

Method	**4060**	**5090**	**Trials**
Static (10ep)	4 min	2 min	1
Std. schedulers (10ep)	4 min	2 min	1
GP-TS	17 min	5 min	20
TD3	17 min	5 min	20
SAC	42 min	12 min	50

Table 7: Per-cell wall-clock time on NVIDIA RTX 4060 (8GB) and RTX 5090 (32GB).

Method	Webpage	Wikigold	CoNLL-03
Distantly supervised NER methods			
BOND Stage I (Liang et al., 2020)	59.11	52.15	75.61
BOND full (Liang et al., 2020)	65.74	60.07	81.48
RoSTER (Meng et al., 2021)	54.4†	67.80	85.40
SCDL (Zhang et al., 2022)	68.47	64.13	83.69
SANTA (Shao et al., 2023)	71.79	—	86.59
CLIM (Wang et al., 2023)	70.00	67.90	85.40
Our scheduling (RoBERTa-CRF, corrected 10-ep retrain F_1)			
Static (10 ep)	0.618	0.468	0.723
+ GP-TS (10ep retrain)	0.610	0.493	[TBD]
+ TD3 (10ep retrain)	0.610	0.493	[TBD]
+ SAC (10ep retrain)	0.603	0.459	[TBD]
Our scheduling (BERT-CRF, corrected 10-ep retrain F_1)			
BERT static	0.606	0.521	[TBD]
BERT GP-TS	0.512	0.487	[TBD]
BERT TD3	0.512	0.487	[TBD]
BERT SAC	0.566	0.490	[TBD]

Table 8: Comparison with distantly supervised NER methods. Our rows report corrected 10-epoch retraining results for completed Webpage/Wikigold cells; CoNLL-2003 optimizer retraining remains unavailable. Our approach is complementary: scheduling optimizes training rather than label quality.

Method	Approach	Model	Key Difference
BOND	Self-training	BERT-CRF	Denoising via self-training
RoSTER	Self-training + refinement	BERT	Label correction
SANTA	Self-training + neg. sampling	BERT	Noise-type-specific strategies
CLIM	Class-rebalancing self-training	RoBERTa	Rebalances noisy class dist.
SCDL	Self-training + denoising	BERT	Confidence-driven learning
SALO	Adaptive label correction	BERT	Self-adaptive labels
PBT	Population-based search	General	Static population, no RL
BOHB	BO + bandits	General	Static config search
Ours	RL + BO scheduling	RoBERTa-CRF	Dynamic schedule optimization

Table 9: Comparison with related methods.

Hyperparameter	Static	GP-TS (Webpage)	GP-TS (Wikigold)	Search Range
Learning rate	1×10^{-5}	2.26×10^{-5}	2.26×10^{-5}	$[10^{-6}, 10^{-4}]$
Weight decay	1×10^{-4}	6.09×10^{-4}	6.09×10^{-4}	$[10^{-5}, 10^{-3}]$
Adam β_1	0.9	0.817	0.817	$[0.8, 0.99]$
Adam β_2	0.999	0.971	0.971	$[0.95, 0.999]$
Warmup steps	0	65	65	$[50, 500]$
Batch size	16	10	10	$[8, 32]$

Table 10: Best GP-TS configurations vs. static baseline.