When Tasks Share Structure: A Comparative Study of Training Strategies for Generative Event Extraction

Rishi Ravikumar and Riza Batista-Navarro
Department of Computer Science, University of Manchester, United Kingdom
{rishi.ravikumar, riza.batista}@manchester.ac.uk

Abstract

Event extraction requires performing two interdependent subtasks: event detection and event argument extraction. While prior work has explored pipelined and joint training approaches, the question of how best to coordinate training across these subtasks in generative LLM-based systems remains open. We present a systematic study comparing three training paradigms: disjoint, fully shared and hybrid weight allocation, instantiated as eight concrete strategies and evaluated on ACE2005 and RichERE across multiple instruction-tuned LLMs. Our findings show that training strategy has a consistent and meaningful effect on extraction accuracy, and that a clear best-performing strategy emerges across models and benchmarks. We believe that these findings could extend beyond event extraction to other information extraction tasks that decompose into interdependent subtasks.

1 Introduction

Event extraction (EE) aims to identify structured event information from text. It is typically decomposed into two subtasks: event detection (ED), which identifies event triggers and their types, and event argument extraction (EAE), which identifies event participants and their roles (LDC, 2005). A long-standing question in EE is how these interdependent subtasks should be coordinated during training: should they be learned independently in a pipeline, jointly or through some intermediate form of interaction?

Both pipeline and joint formulations have been extensively explored, each with well-known trade-offs. Pipeline approaches offer modularity but suffer from error propagation, while joint approaches can capture cross-task dependencies at the cost of increased complexity. The advent of large language models (LLMs) has introduced a new paradigm, where EE is framed as a conditional text generation problem. This shift brings the question of training strategy into a new setting, one that has received little systematic attention: what is the best strategy for fine-tuning LLMs for generative event extraction?

In this work, we systematically investigate how different training strategies affect extraction performance in generative event extraction. We study three paradigms: disjoint parameter allocation, fully shared parameters and hybrid configurations with partial parameter sharing, instantiated as eight **computationally equivalent** training strategies. Models are fine-tuned using LoRA (Low-Rank Adapters) (Hu et al., 2021) and evaluated on ACE2005 (Doddington et al., 2004) and RichERE (Song et al., 2015) across three instruction-tuned LLMs ranging from 3B to 12B parameters.

Our results show that training strategy has a consistent and meaningful effect on extraction accuracy across models and benchmarks. We find that a disjoint approach in which the ED adapters are initialised from pre-trained EAE adapters consistently outperforms both de facto approaches: fully independent training and joint modelling. Joint modelling, where both tasks are handled within a single pass, on the contrary, proves to be the weakest configuration overall. We also find that robustness to strategy choice increases with model size. Our contributions are as follows:

- We propose a taxonomy of training strategies for generative event extraction based on parameter sharing and task interaction, spanning disjoint, fully shared and hybrid paradigms.
- We conduct a controlled empirical study of eight strategies across three LLMs and two benchmarks, providing a comprehensive comparison of training strategies for generative event extraction[1].

[1]Our code is available at www.github.com/rishi-ravikumar/GenerativeEventExtractionTrainingStrategies.

Proceedings of the 9th Workshop on Event Extraction and Understanding: Challenges and Applications, pages 38–48

- We provide insights into how task decomposition, transfer and parameter sharing affect extraction accuracy across models and datasets.

2 Related Work

2.1 Event Extraction: Pipeline and Joint Approaches

Pipeline and joint approaches represent the two dominant paradigms for event extraction, each with well-established trade-offs. Pipeline approaches train separate models for ED and EAE, applying them sequentially at inference time, offering modularity but suffering from error propagation: mistakes in event detection directly degrade argument extraction (Xiang and Wang, 2019). Joint approaches model both subtasks within a unified framework, enabling cross-task interaction and interdependencies to be exploited, thereby alleviating error propagation at the cost of increased model complexity (Lin et al., 2020; Xiang and Wang, 2019). The relative merits of each paradigm remain setting-dependent, with leading systems spanning both: joint models such as OneIE (Lin et al., 2020) and DyGIE++ (Wadden et al., 2019) alongside pipeline approaches such as TagPrime (Hsu et al., 2023), as evidenced by the standardised evaluation in Huang et al. (2024).

2.2 Generative Event Extraction

The reframing of event extraction as a conditional text generation problem has gained significant traction with the advent of pre-trained language models. Early generative approaches include Du and Cardie (2020), who formulate EAE as question answering, and Paolini et al. (2021), who cast a range of structured prediction tasks, including EE, as translation between augmented natural languages. Lu et al. (2021) propose Text2Event, which directly generates structured event records from text using constrained decoding, handling ED and EAE jointly in a single pass. Hsu et al. (2022) propose DEGREE, which frames EE as prompt-based conditional generation, and they explicitly evaluate both pipeline (DEGREE-PIPE) and joint (DEGREE-E2E) configurations, finding that the joint configuration generally outperforms the pipeline configuration on ACE2005, particularly in low-resource scenarios. More recently, work on large instruction-tuned LLMs has examined the upper bound of the generative paradigm: Gao et al. (2023) evaluate an earlier version of ChatGPT on few-shot EE and find that it considerably falls short of supervised approaches, while Srivastava et al. (2025) systematically study instruction tuning strategies for event extraction. Across this body of work, the question of how training should be coordinated across ED and EAE in generative models has received no systematic attention, with most work committing to a single fixed configuration. This paper addresses this gap directly.

2.3 Multi-Task and Transfer Learning for Information Extraction

The question of how to coordinate learning across related tasks has a long history in NLP. Caruana (1997) establishes that jointly training related tasks provides inductive biases that improve generalisation, and subsequent work has explored both hard parameter sharing, where all tasks share the same parameters, and soft sharing, where tasks maintain separate parameters with regularisation encouraging similarity (Ruder, 2017). In the context of IE, multi-task learning over related subtasks, where shared encoders capture common linguistic features, has consistently outperformed single-task baselines (Wadden et al., 2019; Lin et al., 2020). Paolini et al. (2021) extend this to a broader range of structured prediction tasks within a unified generative framework, demonstrating that related IE tasks can benefit from shared representations at scale. Sequential transfer between related tasks has also proven effective, with pre-training on auxiliary tasks providing beneficial initialisations for downstream extraction objectives (Pruksachatkun et al., 2020). Our work draws directly on these insights, systematically investigating how hard parameter sharing, task-specific parameters and sequential transfer interact in the specific context of generative EE under LoRA-based fine-tuning.

3 Methodology

3.1 Task Formulation

We adopt a generative formulation of event extraction, casting ED, EAE and joint EE as conditional text generation tasks within a prompt-completion framework. In each case, the model takes a natural language sentence as input and generates a structured JSON representation of the predicted event information. We deliberately adopt a minimal prompting strategy, providing no task instructions or additional context beyond the input sentence. Since all models are fine-tuned, the training

process itself is sufficient for the model to learn the target behaviour. This choice avoids confounding effects introduced by prompt engineering and ensures consistency across all experimental conditions.

Event Detection. Given a raw input sentence, the model generates a list of all event triggers present in the sentence along with their corresponding event types, in a single generation step.

Event Argument Extraction. Given the input sentence along with a specific event trigger and its type, provided using simple delimiters, the model generates all arguments associated with that event and their semantic roles. Since argument extraction is conditioned on a single trigger, EAE is run once per trigger. This formulation reflects the dependence of EAE on ED outputs.

Joint Event Extraction. Given a raw input sentence, the model generates complete event structures in a single pass, including triggers, event types and all associated arguments.

The prompt-completion formats for all three formulations are illustrated in Table 1, with all completions structured as JSON arrays with consistent field names across tasks.

3.2 Training Strategies

We investigate eight training strategies organised into three paradigms based on how adapters are allocated across ED and EAE. Unless otherwise stated, inference follows a pipeline: ED is run first to identify triggers and event types, which are then passed as input to EAE. During training, EAE is conditioned on gold triggers and event types, whereas at inference it relies on predictions from ED.

3.2.1 Disjoint Training

Disjoint strategies assign separate sets of adapters to ED and EAE, with no parameter sharing between the two tasks at inference time.

Independent (S1). Separate adapter sets are trained for ED and EAE independently, with no interaction between tasks at any stage. This represents the standard pipeline baseline.

Forward Transfer (S2). The ED adapters are trained first and used to initialise the EAE adapters, which are then trained. This examines whether ED supervision provides a useful starting point for argument extraction.

Backward Transfer (S3). The EAE adapters are trained first and used to initialise the ED adapters, which are then trained. This examines whether exposure to argument-level supervision benefits ED.

3.2.2 Fully Shared Training

Fully shared strategies use a single set of adapters for both tasks, with no task-specific parameters.

Joint Modelling (S4). A single set of adapters is trained to generate complete event structures, including triggers, types and arguments, in a single pass. Unlike the other strategies, inference is also performed in a single pass. This represents the standard joint baseline.

Mixed Training (S5). A single set of adapters is trained on interleaved batches of ED and EAE instances. Despite sharing a single adapter set, inference remains pipelined: the model first performs ED, then EAE.

3.2.3 Hybrid Training

Hybrid strategies partition adapters into two sets: a shared set applied to the lower layers, and task-specific sets applied to the upper layers. Since lower layers of transformer-based LLMs encode general linguistic and semantic representations while upper layers encode increasingly task-specific information (Rogers et al., 2021), this design allows the model to leverage common event semantics in the shared layers while retaining task-specific capacity in the upper layers.

Partial Sharing (S6). Adapters in the lower layers are shared between ED and EAE, while the upper layers maintain separate task-specific adapter sets. This design interpolates between fully disjoint and fully shared training: at 0% sharing the strategy reduces to Independent training (S1), and at 100% sharing it reduces to Mixed Training (S5). We experiment with three configurations: sharing the lower 25% of layers (**S6.1**), 50% (**S6.2**) and 75% (**S6.3**). An overview of this architecture is shown in Figure 1.

3.3 Benchmark Datasets

We evaluate on ACE2005 and RichERE, the two most widely used benchmarks for event extraction, both focusing on the news domain. We follow the TextEE standardised benchmarking framework (Huang et al., 2024) to download, pre-process and split the data, adopting the standard train/dev/test splits. In particular, we use Split 1 of the five

Task	Example
ED	**Prompt:** A bomb exploded in the city center. **Completion:** `[{'trigger': 'exploded','event_type': 'Conflict:Attack'}]`
EAE	**Prompt:** A bomb exploded in the city center. <trigger> exploded </trigger> <type> Conflict:Attack </type> **Completion:** `[{'argument': 'city center', 'argument_role': 'Place'}]}]`
EE (ED+EAE)	**Prompt:** A bomb exploded in the city center. **Completion:** `[{'trigger': 'exploded','event_type': 'Conflict:Attack', 'arguments': [{'argument': 'city center', 'argument_role': 'Place'}]}]`

Table 1: Prompt-completion formats for different task formulations.

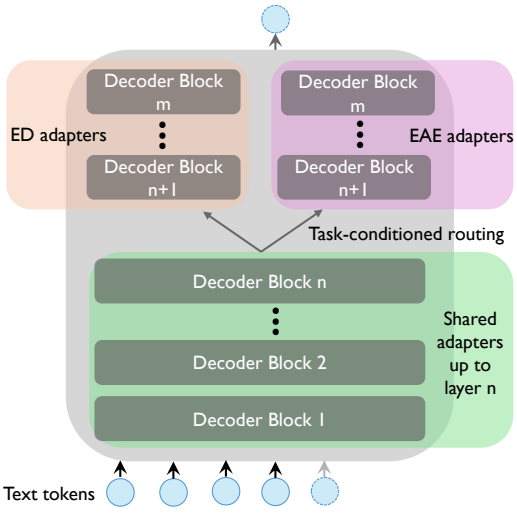

Figure 1: Partial Sharing (S6): lower layers use a shared adapter set across ED and EAE, while upper layers maintain separate task-specific adapter sets.

Dataset	# Event Types	# Argument Types	Train	Dev	Test
ACE05	33	22	3234	348	416
RichERE	38	21	2721	291	403

Table 2: Dataset statistics for ACE2005 and RichERE. Samples containing no events are excluded.

splits provided by TextEE. Additionally, *we exclude samples containing no events* due to the class imbalance in both datasets: fewer than 20% of ACE2005 samples and fewer than 30% of RichERE samples contain at least one event. Retaining null-event samples would therefore heavily skew evaluation towards trivial empty predictions, obscuring meaningful differences in extraction quality across strategies. We note that this filtering precludes direct comparisons with prior work that retain such samples; however, since all strategies are evaluated under identical conditions, cross-strategy comparisons remain fully valid. The data is also pre-processed into the prompt-completion format described in Section 3.1.

ACE2005 is a popular benchmark for event extraction, featuring a well-established ontology of event types and argument roles. RichERE extends this setting with a broader ontology, potentially presenting additional challenges due to its increased schema complexity and smaller size. Together, the two benchmarks allow us to assess the robustness of our findings across different levels of schema complexity and data availability. Key statistics are reported in Table 2.

3.4 Models

We experiment with three open-weight, instruction-tuned decoder-only LLMs: Qwen2.5-3B-Instruct (Qwen et al., 2025), Llama-3.1-8B-Instruct (Meta et al., 2024) and Mistral-Nemo-Instruct-2407 (Mistral AI, 2024), spanning 3B to 12B parameters. This selection provides diversity across both model family and scale, allowing us to assess the consistency of our findings across these dimensions.

3.5 Hardware and Hyperparameters

Hyperparameter	Value
Sampling Parameters	Greedy Sampling
Number of Epochs	2
Precision	bfloat16
Optimizer	AdamW
Learning Rate	2.00E-04
Learning Rate Scheduler	cosine with warm-up
Warm-up Ratio	0.05
Batch Size	4
LoRA Dropout	0.05
Gradient Accumulation Steps	4
Weight Decay	0.01
Gradient Clipping	1.0
Maximum Seq. Length	512
Projections Modified	'q_proj', 'k_proj', 'v_proj', 'o_proj','gate_proj', 'up_proj', 'down_proj'

Table 3: Hyperparameters used for training.

Table 3 lists the key hyperparameters used for fine-tuning. We fine-tune all models using LoRA on NVIDIA RTX A5000, A40 and A100 GPUs, with a cumulative training time of approximately 200 GPU hours. Loss is computed exclusively over completion tokens, with input tokens masked during training. We use a fixed training schedule of 2 epochs across all experiments to ensure comparable

compute across strategies. This choice is further supported by prior QLoRA results (Dettmers et al., 2023), which report stronger performance with 2-epoch fine-tuning compared to 1 and 3 epochs. Additionally, the small dataset sizes and the number of model components adapted by LoRA make 2 epochs a reasonable choice.

3.6 Computational Equivalence

All eight strategies are designed to incur identical training cost, measured as the total number of completion tokens processed under prompt loss masking. Let D_{ED} and D_{EAE} denote the ED and EAE training sets with mean completion lengths m and k respectively. In the disjoint and transfer strategies (S1-S3), two adapter sets are trained independently, one per task, for two epochs each, contributing $2(|D_{\text{ED}}| \cdot m + |D_{\text{EAE}}| \cdot k)$ loss tokens in total. In Joint Modelling (S4), ED and EAE completions are concatenated into a single sequence per sample, and a single adapter set is trained for two epochs, yielding the same total cost. In Mixed Training (S5), the two datasets are interleaved and processed by a single adapter set over two epochs, again resulting in the same total. In the hybrid strategies (S6.1-S6.3), the shared lower-layer adapters are trained on the interleaved combined dataset, while each task-specific upper-layer adapter set is trained on its respective task; the contributions sum to the same quantity. *Training cost is thus held constant across all strategies by construction.*

3.7 Evaluation

We adopt two evaluation metrics following standard practice in event extraction literature (Huang et al., 2024), all reported as micro-averaged F1. A true positive for ED requires both the trigger span and event type to exactly match the gold annotation; we report this as the **TC** (Trigger Classification') score. A true positive for *end-to-end EE* requires the trigger span, event type, argument span and argument role to all exactly match the gold annotation; we report this as the **AC** (Argument Classification') score. Since EAE models are trained on gold triggers but evaluated on predicted triggers, AC is subject to exposure bias. To address this, we additionally propose **AC|TC**, which evaluates argument extraction exclusively over correctly identified triggers; by construction, this ensures that EAE receives input consistent with its training conditions, eliminating the exposure bias gap. However, AC|TC should be interpreted with care: as it is computed over only the subset of instances where ED succeeds, it is sensitive to the composition of that subset and provides only a rough signal of EAE performance in isolation. Together, TC measures ED quality, AC captures end-to-end EE performance, and AC|TC offers a rough measure of EAE quality.

4 Results and Discussion

Table 4 presents TC, AC and AC|TC scores for all eight training strategies across three models and two benchmarks, with each configuration run over three random seeds and averaged. To facilitate comparison across strategies independently of absolute score differences between models, we rank strategies by score within each model and benchmark combination, then average these ranks across models to produce a consolidated ordering for each metric and benchmark; results are reported in Table 5. To assess the consistency of these rankings across models, we compute Kendall's coefficient of concordance (W) for each metric–benchmark combination. In 8 of 9 cases, W ranges from 0.503 to 0.677, indicating moderate to substantial agreement according to commonly used interpretation guidelines adapted from Landis and Koch (1977); The Comprehensive R Archive Network. The remaining case, AC|TC on ACE2005 (W=0.323), indicates fair agreement, although two of the three model pairs still exhibit substantial pairwise concordance. We organise our discussion around the key trends that emerge.

Performance improves consistently with model scale. Across all strategies and both benchmarks, TC and AC scores increase with model size. This trend holds regardless of training strategy, suggesting that the absolute gains from scaling generalise across different forms of task interaction.

Larger models are more robust to training strategy variation. As show in Figure 2, the TC and AC score gap (margin) between the best and worst performing training strategies generally narrows with model size across benchmarks. On ACE2005 TC, this margin is approximately 4.5 points for Qwen2.5 3B, compared to around 2.8 points for Mistral Nemo. A similar trend holds on RichERE and for AC. Training strategy remains a meaningful factor at all scales, but its effect is more pronounced in smaller models. This is likely attributable to two complementary factors: larger models bring

ACE05	Qwen2.5 3B Instruct			Llama 3.1 8B Instruct			Mistral Nemo Instruct 2407		
Strategies	TC	AC	AC|TC	TC	AC	AC|TC	TC	AC	AC|TC
S1: Independent	71.86 ± 0.41	46.15 ± 0.41	66.21 ± 0.34	75.52 ± 0.19	54.56 ± 0.17	72.93 ± 0.12	76.78 ± 0.38	54.83 ± 0.14	72.29 ± 0.23
S2: Forward Transfer	71.86 ± 0.41	47.23 ± 0.63	**67.71 ± 0.59**	75.52 ± 0.19	54.49 ± 0.35	**72.95 ± 0.50**	76.78 ± 0.38	55.55 ± 0.39	**73.15 ± 0.49**
S3: Backward Transfer	73.62 ± 0.27	47.28 ± 0.57	65.87 ± 0.40	**77.20 ± 0.43**	**55.91 ± 0.36**	72.25 ± 0.52	**78.15 ± 0.21**	**55.71 ± 0.25**	71.48 ± 0.19
S4: Joint Modelling	69.26 ± 0.24	43.74 ± 0.16	64.62 ± 0.08	75.07 ± 0.13	52.75 ± 0.61	71.02 ± 0.64	75.38 ± 0.39	54.59 ± 0.35	72.20 ± 0.59
S5: Mixed Training	72.67 ± 0.37	47.21 ± 0.80	66.07 ± 0.80	76.64 ± 0.41	53.99 ± 0.39	70.70 ± 0.24	76.89 ± 0.28	54.16 ± 1.07	70.97 ± 0.92
S6.1: Partial Sharing (25%)	**73.72 ± 0.31**	**47.65 ± 0.10**	65.95 ± 0.33	76.31 ± 0.28	54.83 ± 0.85	72.27 ± 0.47	76.67 ± 0.65	55.41 ± 0.94	72.07 ± 0.74
S6.2: Partial Sharing (50%)	73.58 ± 0.45	46.12 ± 0.55	64.45 ± 1.07	75.70 ± 0.18	53.70 ± 0.24	71.70 ± 0.03	77.34 ± 0.45	54.75 ± 0.86	70.99 ± 0.37
S6.3: Partial Sharing (75%)	72.16 ± 0.44	46.12 ± 1.08	65.42 ± 0.81	75.57 ± 0.69	53.52 ± 0.49	71.01 ± 0.10	76.95 ± 0.54	55.70 ± 0.60	72.39 ± 0.10

RichERE	Qwen2.5 3B Instruct			Llama 3.1 8B Instruct			Mistral Nemo Instruct 2407		
Strategies	TC	AC	AC|TC	TC	AC	AC|TC	TC	AC	AC|TC
S1: Independent	60.76 ± 0.87	41.81 ± 0.48	68.70 ± 0.21	64.97 ± 0.46	47.05 ± 0.82	72.37 ± 0.73	66.14 ± 0.63	49.48 ± 0.63	74.11 ± 0.40
S2: Forward Transfer	60.76 ± 0.87	41.70 ± 0.42	68.76 ± 0.36	64.97 ± 0.46	46.68 ± 0.79	71.83 ± 0.78	66.14 ± 0.63	**49.68 ± 0.67**	74.44 ± 0.65
S3: Backward Transfer	61.82 ± 0.15	**42.84 ± 0.08**	**69.13 ± 0.32**	**67.79 ± 0.30**	**49.58 ± 0.63**	72.85 ± 0.38	**67.93 ± 0.94**	49.35 ± 0.59	72.91 ± 0.32
S4: Joint Modelling	58.98 ± 0.18	38.92 ± 0.20	66.14 ± 0.07	64.86 ± 0.21	47.97 ± 0.12	**73.55 ± 0.17**	65.96 ± 0.49	49.57 ± 0.09	**74.86 ± 0.26**
S5: Mixed Training	60.65 ± 0.56	41.22 ± 0.58	67.13 ± 0.23	65.17 ± 0.56	46.59 ± 0.55	71.09 ± 0.13	66.35 ± 0.30	48.74 ± 0.07	72.72 ± 0.30
S6.1: Partial Sharing (25%)	61.97 ± 0.64	42.59 ± 0.54	68.31 ± 0.62	65.41 ± 0.30	46.30 ± 0.47	70.27 ± 0.32	66.30 ± 0.93	48.82 ± 0.83	73.25 ± 0.50
S6.1: Partial Sharing (50%)	60.89 ± 0.47	41.76 ± 0.98	68.53 ± 0.96	65.87 ± 0.52	47.22 ± 0.12	71.57 ± 0.34	66.37 ± 0.31	48.91 ± 0.42	73.18 ± 0.24
S6.2: Partial Sharing (75%)	**62.03 ± 0.60**	42.78 ± 0.88	68.65 ± 0.53	65.31 ± 0.41	46.76 ± 0.42	71.45 ± 0.57	66.37 ± 1.03	47.67 ± 1.05	71.70 ± 1.03

Table 4: TC, AC and AC|TC scores on ACE2005 (top) and RichERE (bottom), averaged over three random seeds (± standard error). For each column, the best performance is shown in bold and the second-best is underlined.

Rank	TC ACE2005	TC RichERE	AC ACE2005	AC RichERE	AC|TC ACE2005	AC|TC RichERE
1	Backward Transfer	Backward Transfer	Backward Transfer	Backward Transfer	Forward Transfer	Forward Transfer
2	Partial Sharing (50%)	Partial Sharing (50%) Partial Sharing (75%)	Partial Sharing (25%)	Independent	Independent	Independent Backward Transfer
3	Mixed Training Partial Sharing (25%)	Partial Sharing (25%)	Forward Transfer	Joint Modelling	Partial Sharing (25%)	Joint Modelling
4	Partial Sharing (75%)	Mixed Training	Independent	Forward Transfer Partial Sharing (50%)	Backward Transfer Partial Sharing (75%)	Partial Sharing (50%)
5	Independent Forward Transfer	Independent Forward Transfer	Partial Sharing (75%)	Partial Sharing (75%)	Joint Modelling	Partial Sharing (25%) Partial Sharing (75%)
6	Joint Modelling	Joint Modelling	Mixed Training	Partial Sharing (25%)	Mixed Training	Mixed Training
7			Partial Sharing (50%)	Mixed Training	Partial Sharing (50%)	
8			Joint Modelling			

Table 5: Model-averaged strategy rankings for each metric and benchmark combination. For each model and benchmark, strategies are ranked by score; ranks are then averaged across models to produce a consolidated ordering. A lower rank indicates better average performance.

stronger prior representations of linguistic and event-related semantics, and their greater parameter capacity allows them to accommodate different forms of task interaction more flexibly, learning both tasks effectively regardless of how training is coordinated. This is consistent with findings in related work showing that larger models exhibit greater robustness to task-level design choices in generative IE settings (Ravikumar et al., 2026).

4.1 Event Detection Performance (TC)

We discuss event detection performance across all eight strategies, as measured by the TC metric. Since TC evaluates only the ED component of the pipeline, strategies that differ solely in their EAE adapter—namely S1 and S2—produce identical TC scores by construction.

Backward Transfer is the strongest strategy for TC. S3 ranks first on both ACE2005 and RichERE in terms of performance based on TC, achieving the highest score in four out of six model-benchmark combinations. This consistent advantage suggests that initialising the ED adapter from a pre-trained EAE adapter provides a useful inductive bias. EAE training exposes the model to rich supervision over event participants and their semantic roles, encouraging representations that are broadly sensitive to event-bearing spans, a property that directly benefits event detection. The fact that this advantage persists after task-specific fine-tuning of the ED adapters suggests that the transferred representations provide a meaningful initialisation rather than being simply overwritten.

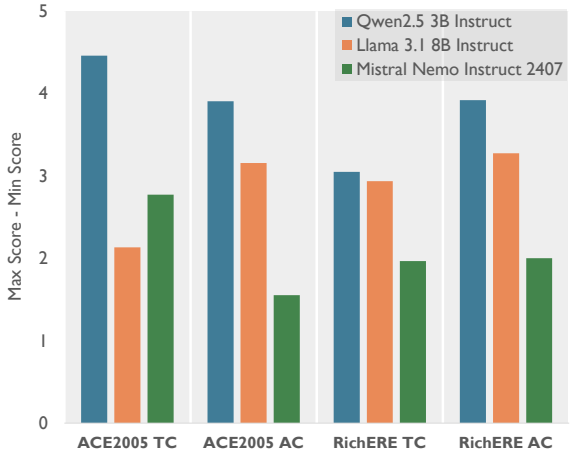

Figure 2: Score gap between the best and worst performing training strategies per model, across metrics and benchmarks.

Independent training leads to poor TC scores. S1 (and by construction S2) ranks second-last on both benchmarks in the model-averaged ranking, outperforming only Joint Modelling. The performance gap between S1 and S3 is particularly informative: the two strategies are identical except for how the ED adapter is initialised, making this a direct controlled comparison that isolates the effect of initialisation from all other experimental factors. The consistent advantage of S3 over S1 across all models and benchmarks confirms that cross-task initialisation provides measurable gains that independent training foregoes.

Joint Modelling is the weakest strategy for TC. S4 ranks last on TC across both benchmarks and all models. Unlike all other strategies, S4 trains on complete event structures and infers in a single pass, forcing a single set of adapters to jointly optimise for trigger identification and argument extraction within a unified output space. This conflation is detrimental to event detection: the competing demands of simultaneously producing triggers, event types, argument spans and roles within a single generation pass degrades the model's ability to precisely identify event-bearing spans.

Mixed Training, by contrast, is competitive for TC. S5 ranks third on ACE2005 and fourth on RichERE, outperforming Joint Modelling in both cases. Although both S4 and S5 use a single set of adapters, they differ fundamentally in both training and inference: S4 trains on complete event structures and infers in a single pass, while S5 trains on interleaved batches of ED and EAE instances and infers in a pipeline. This preservation of task boundaries—at both training and inference time—avoids the conflation that hampers S4. The competitive TC performance of S5 demonstrates that cross-task signals from interleaved training benefit event detection, and that the failure of S4 is attributable specifically to joint generation rather than parameter sharing.

Partial sharing yields strong TC performance. S6.2 (50% sharing) ranks second on both benchmarks, behind only Backward Transfer, making partial sharing the strongest TC strategy aside from S3. On ACE2005, S6.1 (25% sharing) and S6.3 (75% sharing) rank third and fourth respectively, while on RichERE, S6.3 matches S6.2 at rank two and S6.1 ranks third. These results suggest that sharing lower-layer representations between ED and EAE is broadly beneficial for event detection. Lower transformer layers encode common features useful to both subtasks, allowing ED to benefit from the additional supervision signal provided by EAE while retaining task-specific modelling capacity in higher layers.

The consistent advantage of S6.2 over both S6.1 and S6.3 on ACE2005, together with its parity with S6.3 on RichERE, suggests that moderate parameter sharing provides the most effective balance between cross-task transfer and task-specific specialisation. Sharing only 25% of layers yields weaker performance, indicating limited transfer between the subtasks, while increasing sharing to 75% does not provide further gains and can slightly reduce performance on ACE2005. Across both benchmarks, the strongest and most consistent results are obtained with 50% sharing, suggesting that moderate sharing is sufficient to transfer useful event-level representations without excessively constraining task-specific specialisation. This pattern is also reflected in the broader strategy ranking: S6.2 outperforms both fully independent training (S1; 0% sharing) and fully shared mixed training (S5; 100% sharing).

Overall TC paradigm ranking. At the strategy level, S3 and S4 represent the clear performance extremes, ranking first and last respectively. Between these extremes, the hybrid strategies cluster consistently in the upper-middle ranks, establishing partial parameter sharing as the most reliable paradigm for event detection after backward transfer. S5 performs competitively, while S1 and S2 rank in the

lower half, confirming that independent training without cross-task interaction is an ineffective strategy for event detection. Cross-task interaction with preserved task boundaries, whether through initialisation or parameter sharing, is broadly beneficial for TC performance.

4.2 Event Extraction (AC) and Event Argument Extraction (AC|TC) Performance

We discuss AC and AC|TC performance across all eight strategies. AC captures the compounded effect of both ED and EAE (end-to-end EE), while AC|TC isolates argument extraction quality by conditioning on correctly identified triggers.

Backward Transfer remains the strongest strategy for AC. S3 ranks first on AC for both benchmarks in the model-averaged ranking, continuing its strong performance from TC. This is attributable to two complementary factors: S3 achieves the highest TC scores, maximising the number of correctly identified triggers and thus the opportunities for successful argument extraction; and S3's largely strong AC|TC performance indicates that its dedicated EAE adapter, trained independently on argument extraction, extracts arguments effectively given correct triggers. Together, strong event detection and effective argument extraction account for S3's consistent advantage across models and benchmarks.

The disjoint paradigm performs strongly on AC and AC|TC. S1, S2 and S3 collectively dominate both the AC and AC|TC rankings with S2 ranking first for both benchmarks on AC|TC. This consistent superiority of the disjoint paradigm points to a clear finding: dedicated, task-specific EAE adapters are beneficial for argument extraction, providing the parameter capacity to specialise for the distinct demands of identifying event participants and their roles. Within this paradigm, S2 outperforms S1 and S3 on AC|TC across both benchmarks, demonstrating that initialising the EAE adapter from a pre-trained ED adapter improves argument extraction by providing a useful inductive bias without sacrificing dedicated task-specific capacity. This distinguishes forward transfer from other strategies that also introduce cross-task signal but at the cost of shared parameters.

Joint Modelling recovers on AC|TC despite weak TC. S4 ranks last on AC on ACE2005 but third on RichERE, despite ranking last on TC across both benchmarks. The key to this divergence lies in AC|TC: S4 ranks fifth on ACE2005 and third on RichERE, indicating that the unified output space that impairs trigger identification does not equally impair argument extraction. This asymmetry is structurally motivated: while trigger identification is a relatively self-contained objective that is harmed by the competing demands of simultaneous argument generation, arguments are by definition properties of their trigger, with each argument span and role conditioned on a specific trigger identity. Joint generation therefore reflects the natural structure of EAE, and the adapter learns to model trigger-argument relationships directly within a single pass. This coupling partially compensates for the absence of dedicated EAE parameter capacity. On RichERE, this AC|TC strength is sufficient to sustain competitive AC performance despite weak TC. On ACE2005, where argument extraction is substantially harder, the AC|TC performance cannot compensate for the poor TC, and AC collapses accordingly.

Mixed Training is the weakest strategy for AC and AC|TC. S5 ranks sixth on ACE2005 and last on RichERE for AC, and second-last on AC|TC on ACE2005 and last on RichERE, a striking reversal of its competitive TC performance. The contrast with S4 is instructive: despite underperforming S5 on TC, S4 consistently outperforms S5 on AC|TC. Both strategies share a single adapter set without dedicated EAE capacity, but S4's unified output space forces the adapter to jointly model trigger-argument relationships within a single generation pass, which benefits argument extraction given correct triggers. S5, trained on interleaved but separate ED and EAE instances, never jointly models these relationships and therefore lacks this benefit while equally lacking dedicated EAE parameter capacity. S5 thus benefits from neither the trigger-argument coupling of joint generation nor dedicated parameter capacity, accounting for its consistently poor argument extraction performance despite competitive TC.

Partial Sharing performs inconsistently on AC and AC|TC. The hybrid strategies exhibit pronounced benchmark dependence on both metrics. On AC, S6.1 (25% sharing) ranks second on ACE2005 but sixth on RichERE, while S6.2 (50% sharing) ranks seventh on ACE2005 but fourth on RichERE. S6.3 (75% sharing) is comparatively sta-

ble, ranking fifth on AC across both benchmarks. On ACE2005, where argument extraction is substantially harder, S6.1 performs best among hybrid configurations: preserving the greatest task-specific capacity in the upper layers outweighs the benefit of additional cross-task signal when the EAE objective is demanding. On RichERE, S6.2 performs best, suggesting that a more even balance between shared and task-specific capacity is optimal when argument extraction is less demanding. On AC|TC, the hybrid strategies are uniformly weak, with no configuration ranking above third on either benchmark, and their relative ordering shifts inconsistently across benchmarks. Taken together, the hybrid strategies offer no reliable advantage for argument extraction, with performance varying considerably across benchmarks and metrics.

Overall AC and AC|TC paradigm ranking. At the strategy level, S3 ranks first on AC across both benchmarks while S2 ranks first on AC|TC across both benchmarks. At the paradigm level, disjoint strategies perform best on average across both metrics, followed by hybrid and then fully shared training. Unlike TC, where cross-task parameter sharing drives performance improvements, argument extraction benefits most consistently from dedicated task-specific adapter capacity, with paradigm ranking inversely related to the degree of parameter sharing.

4.3 TC vs AC|TC: A Divergent Picture

Comparing TC and AC|TC rankings reveals a systematic shift in strategy performance across the two metrics. S3 ranks first on both TC and AC, but the relative ordering of all other strategies changes considerably. S2, which ranks in the lower half on TC, rises to first on AC|TC across both benchmarks, elevating the disjoint paradigm as the strongest overall for argument extraction. Conversely, the hybrid strategies, competitive on TC, drop substantially on AC|TC. S5, despite competitive performance on TC, falls to second-last on AC|TC on ACE2005 and last on RichERE. S4, the weakest strategy on TC, recovers to third on AC|TC for RichERE and fifth on ACE2005.

This divergence reflects a fundamental disparity between what each subtask demands. Event detection benefits from controlled cross-task interaction, whether through initialisation-based transfer as in S3, or partial parameter sharing as in the hybrid strategies. Argument extraction, by contrast, is more sensitive to dedicated task-specific capacity: the disjoint strategies, which maintain fully independent EAE adapters, consistently outperform shared and hybrid alternatives on AC|TC. S3 satisfies both requirements simultaneously—the ED adapter benefits from EAE initialisation while the independently trained EAE adapter retains full parameter capacity—which accounts for its consistent dominance across both TC and AC. S2 foregoes the benefit of cross-task initialisation for ED but retains full EAE capacity as well as ED-informed initialisation, producing the strongest AC|TC performance overall.

5 Conclusion

We present a systematic study of eight LoRA-based training strategies for generative event extraction, spanning three paradigms: disjoint, fully shared and hybrid parameter allocation. Experiments across three instruction-tuned LLMs and two benchmarks demonstrate that training strategy has a consistent and meaningful effect on extraction performance, with effects more pronounced at smaller model scales. Backward Transfer is the strongest strategy overall on TC and AC, while Forward Transfer is the strongest on AC|TC. More broadly, event detection benefits from cross-task inductive biases, whether through initialisation-based transfer or parameter sharing, while argument extraction is most reliably served by dedicated task-specific adapter capacity. We believe these findings may generalise beyond event extraction to other information extraction tasks that decompose into interdependent subtasks.

Limitations

Our evaluation covers three LLMs ranging from 3B to 12B parameters. While this provides diversity across model family and scale, extending the analysis to larger models would strengthen the generality of our findings. Similarly, while ACE2005 and RichERE differ in scale and schema complexity, evaluation on additional benchmarks spanning broader domains would provide a more comprehensive assessment. Likewise, evaluating on larger datasets would be insightful. We follow recommended practice and use a fixed 2-epoch training schedule across all settings; however, exploring longer training schedules and selecting checkpoints based on validation loss could further refine performance comparisons across strategies. Finally, our

experiments are conducted under LoRA-based fine-tuning. Whether the observed trends hold under full fine-tuning or alternative parameter-efficient methods remains an open question.

References

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: efficient finetuning of quantized llms. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ACE) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal. European Language Resources Association (ELRA).

Xinya Du and Claire Cardie. 2020. Event extraction by answering (almost) natural questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 671–683, Online. Association for Computational Linguistics.

Jun Gao, Huan Zhao, Changlong Yu, and Ruifeng Xu. 2023. Exploring the feasibility of chatgpt for event extraction. *Preprint*, arXiv:2303.03836.

I-Hung Hsu, Kuan-Hao Huang, Elizabeth Boschee, Scott Miller, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2022. DEGREE: A data-efficient generation-based event extraction model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1890–1908, Seattle, United States. Association for Computational Linguistics.

I-Hung Hsu, Kuan-Hao Huang, Shuning Zhang, Wenxin Cheng, Prem Natarajan, Kai-Wei Chang, and Nanyun Peng. 2023. TAGPRIME: A unified framework for relational structure extraction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12917–12932, Toronto, Canada. Association for Computational Linguistics.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Kuan-Hao Huang, I-Hung Hsu, Tanmay Parekh, Zhiyu Xie, Zixuan Zhang, Prem Natarajan, Kai-Wei Chang, Nanyun Peng, and Heng Ji. 2024. TextEE: Benchmark, reevaluation, reflections, and future challenges in event extraction. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12804–12825, Bangkok, Thailand. Association for Computational Linguistics.

J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

LDC. 2005. Ace (automatic content extraction) english annotation guidelines for events. Accessed: 2025-02-12.

Ying Lin, Heng Ji, Fei Huang, and Lingfei Wu. 2020. A joint neural model for information extraction with global features. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7999–8009, Online. Association for Computational Linguistics.

Yaojie Lu, Hongyu Lin, Jin Xu, Xianpei Han, Jialong Tang, Annan Li, Le Sun, Meng Liao, and Shaoyi Chen. 2021. Text2Event: Controllable sequence-to-structure generation for end-to-end event extraction. 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)*, pages 2795–2806, Online. Association for Computational Linguistics.

Meta and 1 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Mistral AI. 2024. Mistral nemo. Accessed: 2026-03-28.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. *Preprint*, arXiv:2101.05779.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R. Bowman. 2020. Intermediate-task transfer learning with pretrained language models: When and why does it work? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5231–5247, Online. Association for Computational Linguistics.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Rishi Ravikumar, Nuhu Ibrahim, and Riza Batista-Navarro. 2026. Lost in formatting: How output formats skew LLM performance on information extraction. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5498–5513, Rabat, Morocco. Association for Computational Linguistics.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2021. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *Preprint*, arXiv:1706.05098.

Zhiyi Song, Ann Bies, Stephanie Strassel, Tom Riese, Justin Mott, Joe Ellis, Jonathan Wright, Seth Kulick, Neville Ryant, and Xiaoyi Ma. 2015. From light to rich ERE: Annotation of entities, relations, and events. In *Proceedings of the 3rd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 89–98, Denver, Colorado. Association for Computational Linguistics.

Saurabh Srivastava, Sweta Pati, and Ziyu Yao. 2025. Instruction-tuning LLMs for event extraction with annotation guidelines. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 13055–13071, Vienna, Austria. Association for Computational Linguistics.

The Comprehensive R Archive Network. *interpret_kendalls_w: Interpret Kendall's Coefficient of Concordance W*. CRAN. Accessed: 2026-05-18.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Wei Xiang and Bang Wang. 2019. A survey of event extraction from text. *IEEE Access*, 7:173111–173137.