machine learning & knowledge extraction

MDPI

Article

Primitive-Augmented Transformers with Event-Role Side State: Architecture Evidence, Warm-Started Modulation, and Decoupled Tool Interfaces

Nurgali Kadyrbek * and Madina Mansurova

Department of AI & Big Data, Faculty of Information Technologies, Al-Farabi Kazakh National University, Almaty 050040, Kazakhstan; madina.mansurova@kaznu.edu.kz
* Correspondence: kadyrbek.nurgali@kaznu.kz

Abstract

Large language models can emit fluent text while leaving intermediate semantic structure implicit. We study whether explicit event-role and logical-primitive side-state can improve a pretrained decoder without damaging its language behavior. We introduce PAT-ER, a decoder architecture with a normal token stream, an event-role register stream, and a primitive register stream. The primitive stream is motivated by the view that logical primitives answer characteristic semantic questions, such as what licenses a conclusion, what conflicts with it, or why evidence is insufficient. Across eight seeds on the same Qwen3-0.6B backbone, replacing token-pooled auxiliary heads with typed PAT-ER registers improves primitive macro-F1 by 0.209 (95% CI [0.182, 0.237]) and role-to-primitive macro-F1 by 0.091 (95% CI [0.074, 0.110]) with no language-model loss cost. A generic-register control shows that this is not merely the effect of adding latent registers: typed PAT-ER improves over generic registers by 0.116 primitive macro-F1 and 0.110 role-to-primitive macro-F1, with both confidence intervals excluding zero. A warm-started model then recovers pretrained language quality (LM loss 1.344 versus 2.555 for the frozen-backbone register model) while retaining most side-state behavior. Finally, a decoupled interface mode produces robust schema-grounded function calls on 242 held-out prompts (Hermes parse 0.952, exact arguments 0.981, JSON validity 1.000, IDK F1 1.000) while base-mode side-state metrics remain byte-identical to the warm-start baseline. The model is not a theorem prover and does not achieve perfect unseen tool-name copying; the contribution is a measured architecture signal and a usable, guarded interface.

Keywords: large language models; neural-symbolic learning; semantic roles; logical reasoning; tool use; model architecture; function calling

Academic Editor: Nikolaos Pitropakis

Received: 4 June 2026
Revised: 7 July 2026
Accepted: 8 July 2026
Published: 9 July 2026

1. Introduction

Transformer decoders [1,2] provide a powerful token-level substrate for language modeling, but their intermediate semantic organization is normally implicit. This is a practical problem when a model is expected to distinguish proof from refutation, abduction from deduction, evidence from unsupported assertion, and a safe answer from a necessary abstention. These distinctions are not merely output labels. They are semantic operations that structure the answer.

This paper asks whether logical primitives can be treated as *semantic heads*: fixed, named latent channels that answer characteristic questions. For example, implication and

modus ponens ask "from what does this follow?"; abduction asks "what hypothesis would explain the observation?"; contradiction asks "what conflicts with this?"; unknown or IDK asks "what is missing before answering?". The core hypothesis is architectural: if these primitives are represented by a side-state rather than only by pooled token labels, a decoder should learn more stable mappings between event roles, primitive class, and output behavior.

PAT-ER (Primitive-Augmented Transformer with Event-Role Stream) implements this hypothesis with three interacting streams:

1. a standard causal token stream;
2. an event-role stream that stores event and argument registers; and
3. a primitive stream that stores semantic primitive registers.

The event-role stream is grounded in linguistic work on semantic roles and argument structure [3–5]. The primitive stream is grounded in the use of formal reasoning labels and proof/refutation metadata in logical NLP datasets such as ProofWriter [6] and FOLIO [7]. The point is not to turn a language model into a proof engine. The point is to expose a fixed semantic flow that can be ablated, measured, and used. This places PAT-ER inside the broader hybrid-reasoning agenda surveyed in recent MDPI work on LLM–knowledge-graph integration and neural–symbolic reasoning [8,9], but with a narrower architectural claim: the structured state is inside the decoder rather than supplied only as an external graph, retrieved context, or post hoc explanation.

The experimental record supports three bounded claims. First, the typed register architecture matters: on the same Qwen3-0.6B backbone and same data, PAT-ER registers outperform both token-pooled auxiliary heads and a generic learned-register control across eight seeds. The generic-register condition is important because it separates typed event-role $\rightarrow$ primitive flow from the broader effect of adding latent state. Second, the side-state can be attached to a pretrained decoder without destroying language-model behavior, provided the injection path is register-only, detached, and trained at a calibrated injection learning rate. Third, product format generation can be separated from side-state measurement by a mode-gated interface path: base mode preserves the side-state exactly, while `interface_mode` learns schema-grounded tool calls and JSON output.

The paper is also intentionally negative where the evidence is negative. Several plausible paths failed: synthetic external-style augmentation hurt the real external slice; explicit conflict NLI hurt ProofWriter contradiction; token and evidence-item fact heads failed at the available data scale; aux-refresh after warm-start caused class forgetting; and shared-layer interface SFT improved tool calls but damaged the role-to-primitive bridge. These failures are not hidden. They are part of the method: audit, run a bounded gate, document pass and fail, and freeze only after a pass.

Contributions

This article contributes:

- A semantic-head framing of logical primitives as fixed answer-structuring questions, implemented as a primitive side-state coupled to event-role registers.
- A decoder architecture with explicit event-role and primitive streams, including formulas for token/register updates, warm-start-safe injection, and set-valued span supervision.
- An eight-seed architecture matrix showing that PAT-ER registers improve primitive macro-F1 and role-to-primitive macro-F1 over both token-pooled auxiliary heads and generic learned registers on the same pretrained backbone.
- A warm-start procedure that preserves pretrained language behavior while allowing measured side-state modulation.

https://doi.org/10.3390/make8070201

- A decoupled interface mode that produces schema-grounded tool calls and JSON at zero measured cost to the base-mode side-state.

Figure 1 summarizes the three streams and the separation between base-mode semantic flow and the decoupled interface path.

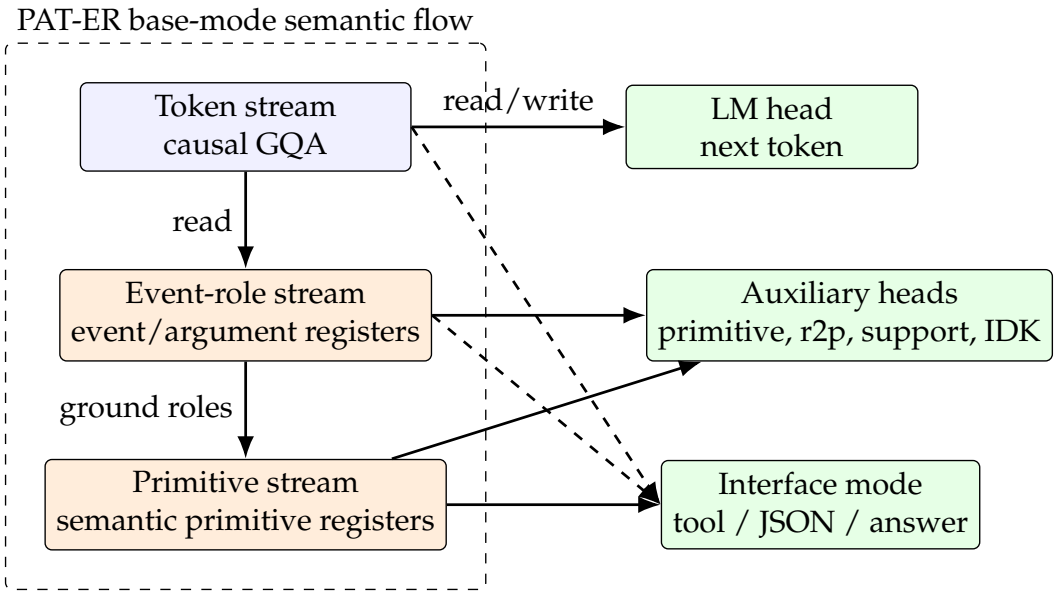

Figure 1. PAT-ER contains a normal token stream plus event-role and primitive side-state streams. Blue boxes denote the token stream, orange boxes denote event-role/primitive registers, and green boxes denote output heads or interface behavior. Solid arrows indicate base-mode information flow; dashed arrows indicate the decoupled interface path. The architecture claim is evaluated in base mode. Product-format generation is handled separately by the decoupled interface path.

2. Related Work

2.1. Transformers and Pretrained Language Models

The Transformer architecture [1] replaced recurrent sequence processing with attention over token states and remains the dominant substrate for pretrained language models. BERT [10] showed the effectiveness of large-scale bidirectional pretraining for language understanding, while GPT-style autoregressive language models demonstrated few-shot behavior at scale [2]. Parameter-efficient adaptation methods such as LoRA [11] show that pretrained models can often be modified without updating all backbone parameters.

PAT-ER builds on this pretrained-decoder substrate but does not treat the decoder state as the only semantic carrier. Its central comparison is therefore not "pretrained versus not pretrained", but "token-pooled labels versus generic learned registers versus typed semantic side-state on the same pretrained backbone".

2.2. Semantic Roles and Event Structure

Semantic-role theory studies how predicates structure their arguments. Dowty's proto-role analysis [3] argues that roles such as agent and patient are better understood through clusters of entailments than through a small rigid inventory. PropBank [4] (https://propbank.github.io/, accessed on 7 July 2026) and FrameNet [5] (https://framenet.icsi.berkeley.edu/, accessed on 7 July 2026) operationalized predicate-argument annotation for computational NLP. These resources motivate the event-role side of PAT-ER: the model should not only classify a primitive but also bind it to the event and argument roles that license it.

Table 1 positions PAT-ER against the structured state mechanisms most relevant to the architecture comparison.

Table 1. Structured-state comparison used to position PAT-ER. The present experiment directly controls token pooling, generic learned registers, and typed PAT-ER registers on the same backbone and data.

Mechanism	State Representation	What It Tests	Status in This Paper
Token-pooled auxiliary heads	Last token/pooled decoder state	Whether labels alone are sufficient without side-state	Condition B
Generic learned registers	Homogeneous latent register bank with comparable update machinery	Whether gains come from any learned latent memory	Condition G
Typed PAT-ER registers	Event-role registers feeding primitive registers	Whether typed semantic flow helps beyond generic state	Condition C
Adapters/LoRA-style adaptation	Parameter-efficient weight changes in decoder layers	Efficient tuning, not explicit semantic state	Related baseline family; not the main control
External graph/RAG methods	Retrieved or symbolic structure outside the decoder	External knowledge support rather than internal side-state	Related work and future integration path
Tool traces/action prompting	Textual or API-level action format	Output behavior and tool use	Decoupled interface mode

2.3. Logical Reasoning Benchmarks

Natural-language reasoning datasets provide structured labels for entailment, contradiction, and uncertainty. SNLI [12] introduced a large NLI benchmark with entailment, contradiction, and neutral labels. FEVER [13] framed fact verification as claim evaluation against evidence. ProofWriter [6] is closer to the present work because it includes proofs, implications, and abductive statements over templated natural language. FOLIO [7] provides natural-language premises with first-order logic annotations and verified reasoning labels.

PAT-ER uses these sources not to copy chain-of-thought into the input but to derive supervised side-state targets: primitive class, support state, rule depth, entailment/refutation status, evidence pointers, and role-to-primitive bridges. The no-chain-of-thought constraint is important: proof metadata is used as supervision, not as rendered reasoning text.

2.4. Hybrid Knowledge Extraction and Neural–Symbolic Reasoning

Recent MDPI literature frames LLM reasoning as part of a wider effort to combine statistical language models with structured knowledge. Dehal et al. [8] survey the reciprocal relationship between LLMs and knowledge graphs, emphasizing that structured representations can improve transparency, factual consistency, and reliability while LLMs automate entity, relation, and schema extraction. Ivanisenko et al. [14] demonstrate a domain-specific hybrid pipeline in which ontology models, graph neural networks, and fine-tuned LLMs are combined for scientific knowledge extraction. Liang et al. [9] review neural–symbolic reasoning architectures and identify symbolic–continuous alignment, dynamic rule learning, and unified architectures as open problems.

PAT-ER is aligned with this direction but takes a different design point. Rather than attach a separate knowledge graph to a frozen text generator, it introduces small

event-role and primitive registers as first-class latent state inside the decoder. The predicate-decomposition result of Tian and Meng [15] is especially relevant: it shows that predicate granularity and predicate polysemy matter for interpretable knowledge-graph reasoning. PAT-ER applies a related principle to language-model internals: logical primitives are treated as a fixed predicate-like inventory whose semantic distinctions can be learned, ablated, and measured.

2.5. Structured Neural State and Tool Use

Structured neural computation has a long history, from graph neural networks [16] to neural modules that impose explicit intermediate structure. Tool-use work such as Toolformer [17] and ReAct [18] shows the value of coupling language models to external actions. PAT-ER differs in two ways. First, the side-state is designed around semantic primitive flow rather than around textual reasoning traces. Second, the tool interface is decoupled from the architecture evaluation path: base mode measures side-state, while `interface_mode` handles schema-grounded generation.

3. Architecture

Let $X_\ell \in \mathbb{R}^{T\times d}$ be the token stream at layer ℓ, $E_\ell \in \mathbb{R}^{M_e\times d}$ the event-role register bank, and $P_\ell \in \mathbb{R}^{M_p\times d}$ the primitive register bank. The normal decoder stream is updated by grouped-query causal self-attention:

$$H_\ell = X_\ell + \mathrm{Attn}_{\mathrm{causal}}(\mathrm{LN}(X_\ell)). \tag{1}$$

The event-role stream reads from token states:

$$\widetilde{E}_\ell = E_\ell + \mathrm{Attn}_{E\leftarrow X}(Q = \mathrm{LN}(E_\ell),\ K = \mathrm{LN}(H_\ell),\ V = \mathrm{LN}(H_\ell)). \tag{2}$$

The token stream then receives an event-role signal Z_ℓ^E:

$$Z_\ell^E = \mathrm{Attn}_{X\leftarrow E}\Big(Q = \mathrm{LN}(H_\ell),\ K = \mathrm{LN}(\widetilde{E}_\ell),\ V = \mathrm{LN}(\widetilde{E}_\ell)\Big). \tag{3}$$

In the from-scratch architecture, the event injection can use token state, register state, and their interaction:

$$H'_\ell = H_\ell + W_E\Big[H_\ell \,\|\, Z_\ell^E \,\|\, (H_\ell \odot Z_\ell^E)\Big]. \tag{4}$$

For warm-started pretrained decoders, this residual fusion is too aggressive: the pretrained hidden state H_ℓ dominates the injection path. The validated warm-start rule is register-only and detached:

$$H'_\ell = H_\ell + \alpha_E W_E\,\mathrm{sg}(Z_\ell^E), \quad \alpha_E \ll 1. \tag{5}$$

The primitive stream reads from the event-enriched token and event states:

$$\widetilde{P}_\ell = P_\ell + \mathrm{Attn}_{P\leftarrow X,E}\Big(Q = \mathrm{LN}(P_\ell),\ K = \mathrm{LN}([H'_\ell;\widetilde{E}_\ell]),\ V = \mathrm{LN}([H'_\ell;\widetilde{E}_\ell])\Big), \tag{6}$$

and writes primitive information back to tokens:

$$Z_\ell^P = \mathrm{Attn}_{X\leftarrow P}\Big(Q = \mathrm{LN}(H'_\ell),\ K = \mathrm{LN}(\widetilde{P}_\ell),\ V = \mathrm{LN}(\widetilde{P}_\ell)\Big). \tag{7}$$

The primitive-enriched token state is

$$H''_\ell = H'_\ell + W_P\Big[H'_\ell \,\|\, Z_\ell^P \,\|\, (H'_\ell \odot Z_\ell^P)\Big], \tag{8}$$

or, in warm-start-safe mode,

$$H''_\ell = H'_\ell + \alpha_P W_P \, \mathrm{sg}(Z_\ell^P). \tag{9}$$

Finally, the layer output combines the base feed-forward network with primitive-conditioned adapters:

$$X_{\ell+1} = H''_\ell + \mathrm{FFN}_{\mathrm{base}}(H''_\ell) + \sum_{k=1}^{K} g_k(\widetilde{P}_\ell) \; \mathrm{Adapter}_k(H''_\ell). \tag{10}$$

3.1. Parameterization and Practical Cost

The warm-started configuration uses hidden size $d = 1024$, 28 decoder layers, context length 1024, 4 event registers, 6 argument registers, and 14 primitive registers. Cross-stream attention is applied every four layers. The model uses 17 proto-role properties, 6 primitive-conditioned FFN adapters of rank 32, and rank-32 vocabulary-pressure projections. The from-scratch 450 M comparison configuration uses the same hidden size with 16 layers and 457.9 M parameters. The Qwen3-compatible warm-start model loads 595.8 M pretrained backbone parameters; the remaining PAT-ER parameters include the extended vocabulary, side-state streams, auxiliary heads, pressure heads, and interface components.

The register counts are small relative to the token context. At a cross-stream layer, token-register attention adds a term proportional to $O(T(M_e + M_a + M_p)d)$, while causal self-attention remains proportional to $O(T^2 d)$ for sequence length T. For $T = 1024$ and 24 total registers, the register-attention term is therefore not the dominant asymptotic cost. The reported experiments measure model metrics and memory-feasible training gates; they do not claim a hardware-independent wall-clock benchmark.

3.2. Primitive Semantic Heads

We use *semantic head* operationally: a fixed side-state channel and readout trained against a characteristic semantic target. This is not a claim that every individual register has been mechanistically interpreted. Each primitive head is associated with a characteristic question:

modus ponens	:	from which verified antecedent does this follow?
syllogism	:	which chained rule licenses this conclusion?
contradiction	:	what evidence or derivation conflicts with it?
abduction	:	what hypothesis would explain the observation?
idk/unknown	:	what evidence is missing before answering?

(11)

This question view is not just interpretive. It determines the supervised labels and losses attached to each head: primitive class, support status, entailment state, role-to-primitive mapping, verifier likelihood, evidence pointer, IDK action, tool intent, and schema mode. Causal interventions and register-level probing are left as future interpretability work.

3.3. Losses

The supervised objective is

$$\mathcal{L} = \lambda_{\mathrm{LM}} \mathcal{L}_{\mathrm{LM}} + \sum_{h \in \mathcal{H}} \lambda_h \, \mathrm{CE}(y_h, f_h(X, E, P)) + \lambda_{\mathrm{span}} \mathcal{L}_{\mathrm{span}} + \lambda_{\mathrm{iface}} \mathcal{L}_{\mathrm{iface}} + \beta \mathcal{L}_{\mathrm{KL}}. \tag{12}$$

https://doi.org/10.3390/make8070201

For ambiguous repeated logical atoms, strict first-occurrence span labels are not semantically well defined. Let S^+ be the set of all token positions that are valid mentions of the same gold argument. The set-valued cross-entropy is

$$\mathcal{L}_{\text{multi}+}(\ell, S^+) = \log \sum_j \exp(\ell_j) - \log \sum_{i \in S^+} \exp(\ell_i). \tag{13}$$

For joint span decoding, valid start/end pairs are restricted to a width band $\mathcal{B}$:

$$(\hat{s}, \hat{e}) = \arg \max_{(s,e) \in \mathcal{B}, e \geq s} \left(\ell_s^{\text{start}} + \ell_e^{\text{end}} \right). \tag{14}$$

3.4. Decoupled Interface Mode

The usable checkpoint has two execution modes. Base mode uses the original decoder blocks and measures the architecture:

$$y_{\text{base}} = F_{\text{D}}(x; \theta_{\text{base}}). \tag{15}$$

Interface mode swaps only the copied top-K interface blocks:

$$y_{\text{iface}} = F_{\text{D}}\left(x; \theta_{\text{base}}, \theta_{\text{iface}}^{(L-K+1:L)} \right). \tag{16}$$

The training invariant is

$$F_{\text{D}}(x; \theta_{\text{base}}) = F_{\text{D+iface}}(x; \theta_{\text{base}}) \quad \text{in base mode,} \tag{17}$$

which is checked empirically as byte-identical primitive, role-to-primitive, and LM metrics.

4. Materials and Methods

4.1. Data Sources and Conversion

Experiments used Python 3.11 (Python Software Foundation, Wilmington, DE, USA), PyTorch 2.5.1 with CUDA 12.4 (Meta Platforms, Inc., Menlo Park, CA, USA; NVIDIA Corporation, Santa Clara, CA, USA), Hugging Face Transformers/tokenizers (Hugging Face, Inc., New York, NY, USA; https://huggingface.co/docs/transformers, accessed on 7 July 2026), and the Qwen3-0.6B backbone from Qwen/Qwen3-0.6B (https://huggingface.co/Qwen/Qwen3-0.6B, accessed on 7 July 2026).

The dataset stack combines synthetic PAT-ER records with converted external reasoning records. Synthetic records provide balanced coverage over 14 primitive families, diversified argument surfaces, support status coverage, hard negatives, and tool/IDK/schema labels. External records come from ProofWriter [6] and FOLIO [7], converted into PAT-ER records with provenance metadata and without rendering proof traces into the model input. NLI and explicit conflict sources such as SNLI [12] and FEVER [13] were evaluated as candidate additions, but the main mix rejects SNLI-style explicit conflict data because it interferes with ProofWriter-style proof refutation.

The current recommended reasoning mix is the repaired real-external mix: diversified synthetic support coverage plus real ProofWriter meta-abduction and shallow OWA reasoning records. Deeper OWA/CWA ProofWriter slices are retained as infrastructure but are not merged into the baseline because they trade away the shallow contradiction competence of the 450M-from-scratch track.

4.2. Supervision Reliability Audit

Because the architecture is trained against converted and synthetic targets, we audited the supervision deterministically. The audit checks label vocabulary membership, event-local primitive/support consistency where the record-level primitive and event-local primitive are intended to coincide, absence of proof metadata or chain-of-thought leakage in the rendered input, provenance completeness, and recoverability of argument spans and evidence targets under the tokenizer. Interface/meta records whose record-level primitive is `tool`, `schema`, or `provenance` are reported as event-local exemptions because their event graph can legitimately describe an observation or conflict while the record-level target supervises interface behavior. The audit does not use an LLM as a label judge.

Predicate relocation was also measured as a diagnostic but not used as a pass/fail criterion because predicates are often lemma-level or abstract event labels rather than surface spans. In the audit sample, predicate relocation was 0/60 for external records and 27/60 for synthetic records while argument spans and evidence targets relocated reliably.

Table 2 summarizes the deterministic reliability audit used to check the converted and synthetic supervision.

Table 2. Deterministic label reliability audit over sampled converted and synthetic records. The audit uses source and schema invariants, not LLM judging. Predicate relocation is diagnostic only because many predicates are lemma-level or abstract event labels rather than surface spans.

Source	n	Primitive Vocab	Support Vocab	Event Consistency	Exemptions	No CoT Leak	Provenance	Arg Spans	Evidence
External	60	1.000	1.000	1.000	0	1.000	1.000	0.920 (104/113)	1.000 (60/60)
Synthetic	60	1.000	1.000	1.000	14	1.000	1.000	1.000 (87/87)	1.000 (44/44)

4.3. Conditions

Conditions B, G, and C form the primary architecture comparison: same pretrained Qwen3-0.6B backbone, same data, same auxiliary labels, and the same training schedule. B reads token-pooled states, G uses a homogeneous learned-register bank with comparable register-update machinery, and C uses typed PAT-ER event-role and primitive registers. Condition D is the usable warm-start model. Condition E is documented as a rejected aux-refresh variant.

Table 3 defines the experimental conditions used in the architecture matrix and warm-start comparisons.

Table 3. Experimental conditions used in the paper.

Label	Description	Purpose	Status
A	From-scratch 450 M PAT-ER	Establish from-scratch reasoning ceiling	Secondary baseline
B	Qwen3-0.6B plus token-pooled auxiliary heads	Same-backbone label baseline without registers	Architecture control
G	Qwen3-0.6B plus generic learned registers	Same register count/update machinery without typed event-role → primitive flow	Structured-state control
C	Qwen3-0.6B plus PAT-ER registers, no injection	Test event-role and primitive side-state	Primary architecture condition
D	Warm-start PAT-ER through Stage 5B	Usable model with recovered LM quality	Model of record
E	Stage 5B plus aux-refresh	Test whether aux heads should be refreshed after adaptation	Rejected: class forgetting

4.4. Training Gates

The experimental method is gate-based:

1. audit labels and input leakage before training;
2. run a bounded experiment with explicit pass/fail criteria;
3. document both positive and negative outcomes;
4. freeze only after a pass;
5. scale only after the failure cause is understood.

This method is why failed branches are visible in the results rather than absorbed into an undocumented search process.

4.5. Statistical Reporting

For the main B/G/C/D matrix, paired seed deltas are bootstrapped with 2000 resamples. For a metric m, seed i, source condition X, and target condition Y, the paired delta is

$$\Delta_i^{X \to Y} = m_i^{(Y)} - m_i^{(X)}. \tag{18}$$

The reported effect is

$$\bar{\Delta} = \frac{1}{n} \sum_{i=1}^{n} \Delta_i, \tag{19}$$

and the 95% confidence interval is the 2.5/97.5 percentile interval of the bootstrap distribution of $\bar{\Delta}$.

4.6. Evaluation Metrics

Primary architecture metrics are primitive macro-F1, role-to-primitive macro-F1, contradiction precision/recall/F1, entailment-refuted precision/recall/F1, and LM loss. Product metrics are Hermes parse rate, exact argument correctness, argument value accuracy, JSON validity, required-key correctness, IDK F1, and base-mode drift against Condition D.

Strict first-occurrence span exactness is diagnostic only. When logical atoms repeat, any co-referent mention is a valid grounding target; therefore, the primary span metric is any-occurrence joint span accuracy.

4.7. Reproducibility Artifacts

The public reproducibility package contains the model implementation, configuration files, dataset converters, validation scripts, small converter fixtures, manuscript source, and compact result tables used to construct the figures and tables in this article. It is available at https://github.com/Pronto-Sage/primitive-augmented-transformer (accessed on 7 July 2026). The usable warm-start/interface model artifacts are available at https://huggingface.co/nur-dev/primitive-augmented-transformer (accessed on 7 July 2026). Converted external records are not redistributed; instead, the repository provides conversion and validation scripts for reconstructing PAT-ER records from the upstream datasets under their respective licenses.

5. Results

5.1. Architecture and Structured-State Controls

The most important result is the B/G/C architecture matrix. Adding typed PAT-ER registers to the same pretrained backbone improves primitive macro-F1 by 0.209 and role-to-primitive macro-F1 by 0.091 over the token-state baseline across eight seeds. A generic register control also improves some metrics over token pooling, so it is not a strawman. However, typed PAT-ER adds a further significant increment over generic registers: +0.116 primitive macro-F1, +0.110 role-to-primitive macro-F1, +0.273 primitive-contradiction recall,

and +0.227 primitive-contradiction F1 on shallow OWA validation data. The same ordering holds on deep/CWA validation.

Figure 2 visualizes the B/G/C contribution pattern, and Tables 4 and 5 report the corresponding eight-seed metrics and paired deltas.

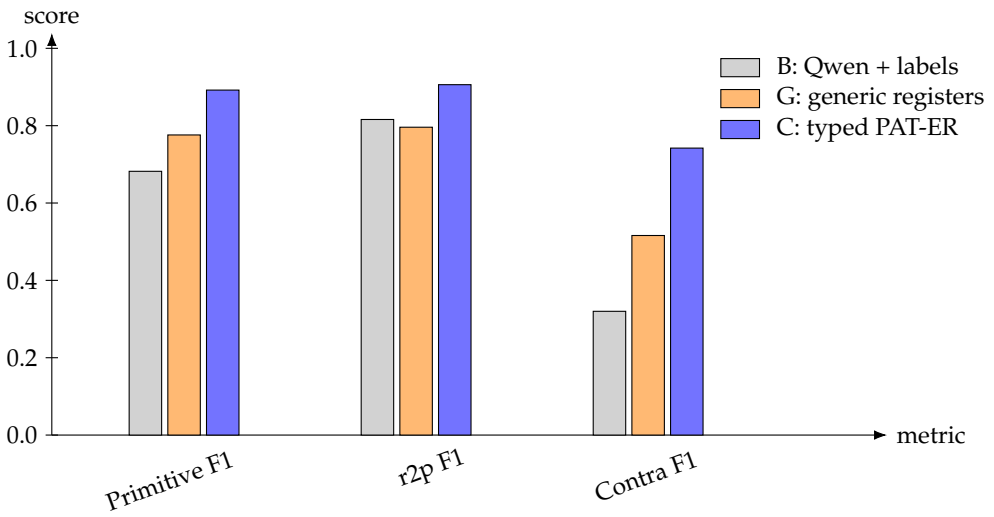

Figure 2. Contribution matrix graph. Generic learned registers improve some metrics over token pooling, but typed PAT-ER registers add a further gain, especially on the role-to-primitive and contradiction axes.

Table 4. Architecture and structured-state controls on shallow OWA validation data, eight seeds. B uses token-pooled auxiliary heads, G uses generic learned registers, C uses typed PAT-ER event-role and primitive registers, and D is the usable warm-start model. Bold indicates the best value in each row among the compared conditions.

Metric	B: Token State	G: Generic Reg.	C: PAT-ER Typed	D: Usable Model
Primitive macro-F1	0.682 ± 0.028	0.775 ± 0.053	**0.891 ± 0.022**	0.841 ± 0.061
Role-to-primitive macro-F1	0.815 ± 0.005	0.795 ± 0.063	**0.905 ± 0.029**	0.873 ± 0.053
Contradiction precision (primitive)	0.337 ± 0.147	0.702 ± 0.224	**0.780 ± 0.176**	0.745 ± 0.191
Contradiction recall (primitive)	0.307 ± 0.153	0.494 ± 0.215	**0.767 ± 0.183**	0.761 ± 0.233
Contradiction F1 (primitive)	0.320 ± 0.150	0.515 ± 0.181	**0.742 ± 0.090**	0.707 ± 0.109
Contradiction F1 (role-to-primitive)	0.586 ± 0.052	0.341 ± 0.300	**0.755 ± 0.095**	0.735 ± 0.101
Entailment-refuted F1	0.036 ± 0.101	0.059 ± 0.168	**0.335 ± 0.302**	0.261 ± 0.333
LM loss	2.563 ± 0.059	2.563 ± 0.059	2.555 ± 0.057	**1.344 ± 0.099**

Table 5. Paired seed deltas for the structured-state control. Bold entries have bootstrap 95% confidence intervals excluding zero. The G → C rows test whether typed PAT-ER flow improves over generic learned registers.

Delta	Primitive Macro-F1	R2P Macro-F1	Contradiction Recall	Contradiction F1	Entailment-Refuted Recall
B → G, shallow	**+0.094 [0.060, 0.130]**	−0.020 [−0.062, 0.020]	**+0.188 [0.023, 0.364]**	**+0.195 [0.072, 0.316]**	+0.073 [−0.062, 0.281]
G → C, shallow	**+0.116 [0.080, 0.151]**	**+0.110 [0.064, 0.160]**	**+0.273 [0.148, 0.398]**	**+0.227 [0.099, 0.380]**	+0.333 [−0.031, 0.688]
B → C, shallow	**+0.209 [0.182, 0.237]**	**+0.091 [0.074, 0.110]**	**+0.460 [0.278, 0.636]**	**+0.422 [0.277, 0.561]**	**+0.406 [0.167, 0.677]**
G → C, deep/CWA	**+0.097 [0.070, 0.125]**	**+0.091 [0.048, 0.136]**	**+0.307 [0.182, 0.420]**	**+0.245 [0.146, 0.354]**	**+0.398 [0.057, 0.761]**

The role-to-primitive contradiction axis is the clearest discriminator between generic and typed structure. Generic registers are worse than token pooling on role-to-primitive contradiction F1, while typed PAT-ER recovers it. In paired seed deltas, B → G is negative on this axis (−0.273, 95% CI [−0.443, −0.097]), whereas G → C is strongly positive (+0.483, 95%

CI [0.239, 0.733]). This indicates that undifferentiated register capacity cannot substitute for typed event-role → primitive coupling.

5.2. Warm-Started Usable Model

Condition C proves the architecture. Condition D is the usable model: it retains much of the side-state behavior while reducing LM loss from 2.555 to 1.344. The warm-start sequence was not a single lucky tuning run. It required four validated stages: frozen-backbone register learning, safe register-only injection, upper-layer adapter opening, and KL-guarded top-layer adaptation.

Figure 3 and Table 6 summarize the warm-start validation sequence.

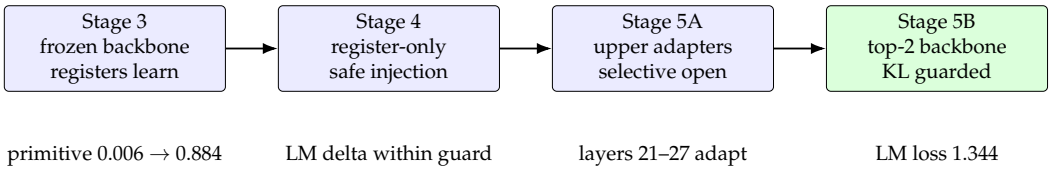

Figure 3. Warm-start validation sequence. Blue boxes denote intermediate warm-start stages, the green box denotes the accepted KL-guarded stage, and arrows indicate the training progression. The usable model is obtained only after safe injection, selective adapter opening, and KL-guarded top-layer adaptation.

Table 6. Warm-start validation stages.

Stage	What Was Tested	Key Result
Stage 3	Train PAT-ER registers on frozen Qwen3 backbone	Primitive macro-F1 0.006 to 0.884; LM unchanged
Stage 4	Open safe register-only injection	LM delta within guard; KL below threshold; generation modulated
Stage 5A	Open upper-layer PAT-ER adapters	Layers 21–27 adapt; lower layers remain unchanged
Stage 5B	Unfreeze top two Qwen layers under KL guard	Backbone adapts safely; KL guard never triggers

5.3. Decoupled Product Interface

The first shared-layer interface SFT improved tool calls but damaged the role-to-primitive bridge by 0.094. The accepted solution decouples product generation from side-state measurement: train copied top decoder blocks only in `interface_mode`; keep base mode byte-identical to Condition D.

Figure 4 and Table 7 report the held-out product-interface evaluation.

Table 7. Decoupled interface evaluation on 242 held-out prompts, three seeds. Bold indicates metrics that meet or exceed the stated target/pass criterion.

Metric	Result	Target	Verdict
Hermes parse	**0.952**	≥0.90	Pass
Tool arguments exact	**0.981**	≥0.80	Pass
Argument value accuracy	0.923	–	Reported
JSON valid/required keys	**1.000/1.000**	≥0.95	Pass
IDK precision/recall/F1	**1.000**	1.000	Pass
Base-mode primitive/r2p/LM vs. D	byte-identical	exact	Pass
Tool-name exact accuracy	0.886	≥0.90	Boundary

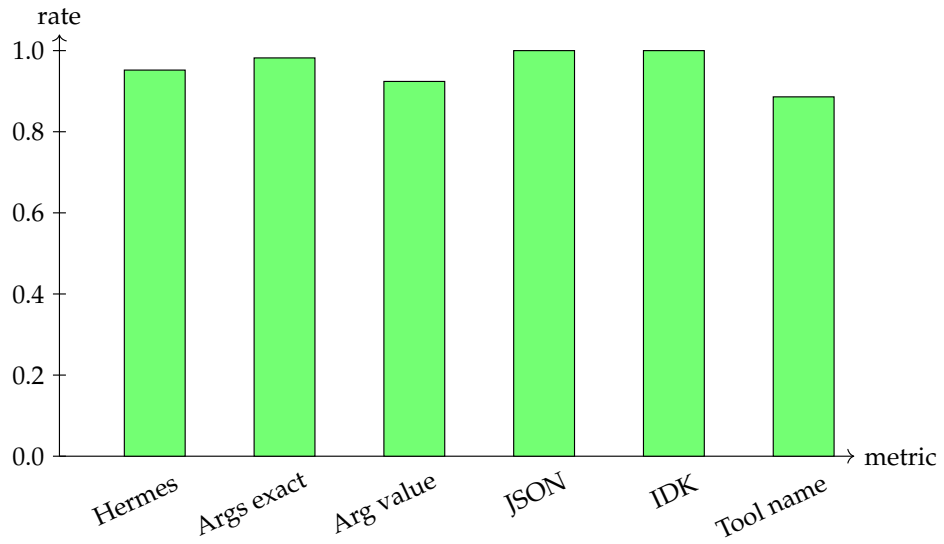

Figure 4. Decoupled interface metrics on 242 held-out prompts. Tool-name exact copying is the remaining boundary; schema-grounded calls, exact arguments, JSON, and IDK pass.

On 242 held-out prompts across three seeds, the decoupled interface reaches Hermes parse 0.952, exact argument correctness 0.981, JSON validity 1.000, and IDK F1 1.000. Exact unseen tool-name copying remains imperfect at 0.886. This is reported as an interface limit, not as a side-state regression, because base mode is identical to Condition D.

5.4. Negative Results

The negative result record is central to the claim because it shows that the reported path is not merely the last surviving configuration. Several plausible alternatives failed for diagnosed reasons.

The pattern is consistent. Synthetic substitutes for real external data hurt the real external slice. Explicit conflict NLI helps its own surface contradiction but hurts ProofWriter contradiction. Fact-level pointer heads do not help at the available refutation density. Aux-refresh on the adapted backbone causes class forgetting. Shared-layer SFT improves output formatting but damages the bridge. The accepted recipe avoids these failure modes by keeping architecture measurement and product generation decoupled.

Figure 5 and Table 8 summarize the documented negative-result gates and retained decisions.

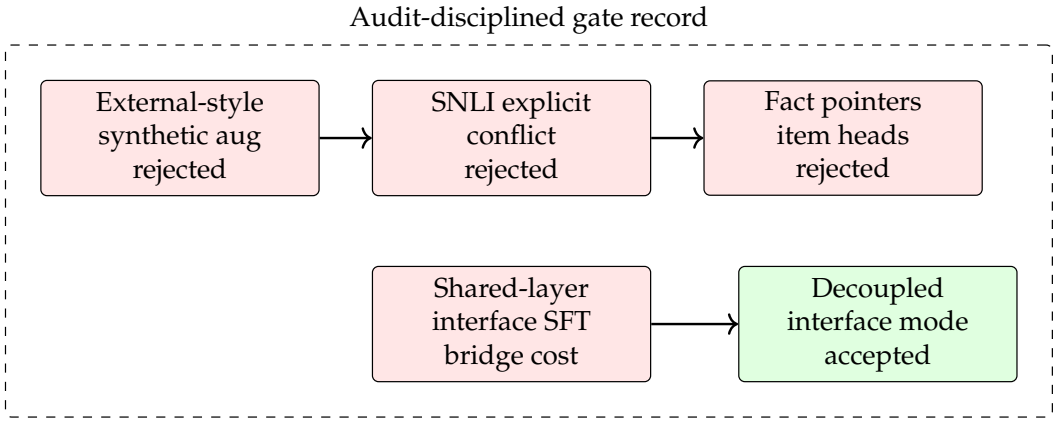

Figure 5. Negative-result map. Plausible additions were tested and rejected when they harmed the targeted slice or damaged the bridge.

Table 8. Documented negative results and retained decision.

Gate	Observed Failure	Decision
Synthetic external-style augmentation	Hurt the real external slice even when synthetic-ext spans were learnable	Do not merge; use only for eval/curriculum
SNLI/explicit conflict NLI	Improved its own conflict form but damaged ProofWriter contradiction	Reject from main mix
Token fact pointers	Failed to learn distributed fact sets and regressed MP	Gate off
Evidence-item heads	Supporting item learned weakly; contradicting item did not learn	Gate off
Deep/CWA from-scratch data	Improved in-distribution deep/CWA but destroyed shallow contradiction	Do not merge for 450M-from-scratch
Stage 6B aux-refresh	Catastrophic forgetting of contingency and axiom boundaries	Reject E
Shared-layer interface SFT	Tool calls improved but role-to-primitive regressed by 0.094	Replace with decoupled interface mode

6. Discussion

6.1. Logical Primitives as Semantic Heads

The evidence supports the view that logical primitives can be operationalized as semantic heads. The primitive stream is not a symbolic theorem prover. It is a fixed latent interface that forces the model to answer different semantic questions with different state channels. This matters because many reasoning failures are not simply failures to produce the right text; they are failures to separate the kind of answer being produced.

The strongest support is the B/G/C ladder. All three conditions share the same Qwen3-0.6B backbone, data, and supervised labels. B has access to labels through token-pooled auxiliary heads. G adds learned registers but removes typed event-role-to-primitive organization. C restores the typed PAT-ER flow. The consistent G-to-C gain means that the result is not explained merely by label availability or by adding generic latent state.

6.2. Fixed Flow Versus Flexible Generation

The answer to whether the approach can be used as fixed flow is yes, but only where the flow is treated as semantic state, not as a complete reasoning engine. The fixed flow is:

$$\text{tokens} \rightarrow \text{event roles} \rightarrow \text{primitive registers} \rightarrow \text{support/IDK/tool/interface decisions.} \quad (20)$$

This flow is useful because it is ablatable and measurable. Removing registers or replacing them with token-pooled heads changes the result. However, deep closed-world refutation remains difficult, and the model should not be described as solving theorem proving.

6.3. Why Warm-Start Changed the Strategy

The 450M-from-scratch track was valuable because it exposed the right data and label structure. It also reached a reasoning ceiling. Warm-starting from Qwen3-0.6B changed the regime: the pretrained decoder supplies language competence, while PAT-ER supplies explicit semantic side-state. The safe modulation rule—register-only fusion, detached fuse input, injection-only training, and very small injection learning rate—is therefore not an implementation detail; it is the condition under which side-state can attach to a pretrained decoder without damaging it.

6.4. Usability Requires Decoupling

The product interface result shows that architecture evidence and interface behavior should not share all adaptation capacity. Shared-layer SFT created a direct conflict: it made tool calls better but damaged the role-to-primitive bridge. Mode-gated interface layers solve this by making output behavior a separate execution path. This is a practical design rule for usable systems: measure side-state in base mode; generate product-format outputs in `interface_mode`; never treat the two as interchangeable.

7. Limitations

PAT-ER proposes and predicts; it does not prove, certify, or guarantee.

First, the reasoning datasets are narrow. ProofWriter and FOLIO are appropriate for controlled logical supervision, but they do not cover broad real-world reasoning. The B/G/C matrix strengthens the architectural interpretation, but the metrics remain supervised probes over the PAT-ER annotation space. Second, deep/CWA refutation remains a known limit. Warm-started registers exceed the from-scratch 450 M ceiling, but contradiction recall is not perfect. Third, the usable interface is not safety-certified. Tool calls are proposals and must be validated against a real schema before execution. Fourth, unseen tool-name exact accuracy is 0.886, below the exact-name target of 0.90. The failures are semantic substitution and multi-token truncation on a minority of unseen names. Fifth, 760M-from-scratch is not validated; the validated usable path is warm-started Qwen3-0.6B plus PAT-ER side-state and decoupled interface mode.

The paper also does not claim that every logical primitive can be exhaustively captured by a fixed neural head. The claim is narrower: selected primitive families can be treated as semantic heads well enough to produce measurable architecture signal and usable guarded output.

8. Conclusions

PAT-ER tests a concrete hypothesis: logical primitives can function as semantic heads in a decoder architecture when they are coupled to event-role registers rather than inferred only from pooled token states or stored in generic learned registers. The eight-seed B/G/C matrix supports this hypothesis. Relative to token pooling, typed PAT-ER improves primitive macro-F1 by 0.209 and role-to-primitive macro-F1 by 0.091 with no LM-loss cost on the same pretrained backbone. Relative to generic registers, it adds +0.116 primitive macro-F1 and +0.110 role-to-primitive macro-F1, showing that typed flow matters beyond undifferentiated latent capacity.

The usable-model path is also established but bounded. Warm-started PAT-ER recovers pretrained LM quality, safe register-only modulation preserves the decoder, and decoupled interface mode yields robust schema-grounded function calling and JSON output at zero measured side-state cost. The model is not a theorem prover, not production safety-certified, and not a perfect tool-name copying system. Its contribution is a measured, ablatable semantic side-state and a practical way to use it without damaging the base language model.

Author Contributions: Conceptualization, N.K. and M.M.; methodology, N.K.; software, N.K.; validation, N.K. and M.M.; formal analysis, N.K.; investigation, N.K.; data curation, N.K.; writing—original draft preparation, N.K.; writing—review and editing, N.K. and M.M.; visualization, N.K.; supervision, M.M.; project administration, M.M.; funding acquisition, M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Higher Education of the Republic of Kazakhstan, grant number BR24993001, "Creation of a large language model (LLM) to maintain the implementation of Kazakh language and increase the technological progress".

Institutional Review Board Statement: Not applicable. This work uses synthetic data and publicly released research datasets under their stated licenses. No new human-subjects data was collected.

Informed Consent Statement: Not applicable.

Data Availability Statement: The manuscript source, model code, configuration files, conversion scripts, evaluation scripts, converter fixtures, and compact result-summary records are publicly available at https://github.com/Pronto-Sage/primitive-augmented-transformer (accessed on 7 July 2026). The usable warm-start/interface model artifacts are available at https://huggingface.co/nur-dev/primitive-augmented-transformer (accessed on 7 July 2026). Upstream datasets are available from their original providers and are cited in the manuscript; converted external records are not redistributed because reuse is governed by upstream licenses. The repository provides scripts to reconstruct and validate converted PAT-ER records from the upstream sources.

Acknowledgments: The authors gratefully acknowledge support from the Ministry of Science and Higher Education of the Republic of Kazakhstan, and computational resources provided by Al-Farabi Kazakh National University.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems 30, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
2. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In Proceedings of the Advances in Neural Information Processing Systems 33, Online, 6–12 December 2020; pp. 1877–1901.
3. Dowty, D.R. Thematic Proto-Roles and Argument Selection. *Language* **1991**, *67*, 547–619. [CrossRef]
4. Palmer, M.; Gildea, D.; Kingsbury, P. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Comput. Linguist.* **2005**, *31*, 71–106. [CrossRef]
5. Baker, C.F.; Fillmore, C.J.; Lowe, J.B. The Berkeley FrameNet Project. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Montreal, QC, Canada, 10–14 August 1998; pp. 86–90. [CrossRef]
6. Tafjord, O.; Dalvi, B.; Clark, P. ProofWriter: Generating Implications, Proofs, and Abductive Statements over Natural Language. In *Proceedings of the Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 3621–3634. [CrossRef]
7. Han, S.; Schoelkopf, H.; Zhao, Y.; Qi, Z.; Riddell, M.; Zhou, W.; Coady, J.; Peng, D.; Qiao, Y.; Benson, L.; et al. FOLIO: Natural Language Reasoning with First-Order Logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 22017–22031. [CrossRef]
8. Dehal, R.S.; Sharma, M.; Rajabi, E. Knowledge Graphs and Their Reciprocal Relationship with Large Language Models. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 38. [CrossRef]
9. Liang, B.; Wang, Y.; Tong, C. AI Reasoning in Deep Learning Era: From Symbolic AI to Neural–Symbolic AI. *Mathematics* **2025**, *13*, 1707. [CrossRef]
10. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. 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*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2019; pp. 4171–4186. [CrossRef]
11. Hu, E.J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models. In Proceedings of the International Conference on Learning Representations, Online, 25–29 April 2022.
12. Bowman, S.R.; Angeli, G.; Potts, C.; Manning, C.D. A Large Annotated Corpus for Learning Natural Language Inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2015; pp. 632–642. [CrossRef]

13. Thorne, J.; Vlachos, A.; Christodoulopoulos, C.; Mittal, A. FEVER: A Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2018; pp. 809–819. [CrossRef]
14. Ivanisenko, T.V.; Demenkov, P.S.; Ivanisenko, V.A. An Accurate and Efficient Approach to Knowledge Extraction from Scientific Publications Using Structured Ontology Models, Graph Neural Networks, and Large Language Models. *Int. J. Mol. Sci.* **2024**, *25*, 11811. [CrossRef] [PubMed]
15. Tian, X.; Meng, Y. PDEC: A Framework for Improving Knowledge Graph Reasoning Performance through Predicate Decomposition. *Algorithms* **2024**, *17*, 129. [CrossRef]
16. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The Graph Neural Network Model. *IEEE Trans. Neural Netw.* **2009**, *20*, 61–80. [CrossRef] [PubMed]
17. Schick, T.; Dwivedi-Yu, J.; Dessi, R.; Raileanu, R.; Lomeli, M.; Hambro, E.; Zettlemoyer, L.; Cancedda, N.; Scialom, T. Toolformer: Language Models Can Teach Themselves to Use Tools. In Proceedings of the Advances in Neural Information Processing Systems 36, Orleans, LA, USA, 10–16 December 2023.
18. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. In Proceedings of the International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.