Post-Deployment Self-Improving Language Agents: Learning, Memory, and Adaptation

Estevam Hruschka
Megagon Labs
Mountain View, CA, USA
estevam@megagon.ai

Abstract

Large language models are increasingly deployed as long-running language agents that retrieve information, use tools, maintain memory, and operate in environments that change after deployment. Yet post-deployment learning is still often handled as a loose collection of prompt edits, retrieval refreshes, memory writes, corrections, or occasional fine-tuning. This tutorial develops a unified view of **post-deployment self-improving language agents**: systems that use deployment experience to produce persistent changes that improve subsequent performance while remaining empirically grounded, auditable, and controllable.. We connect modern agent research with never-ending, lifelong, and continual learning while shifting the unit of analysis from model parameters only to the complete deployed system. We explore the literature, from classic papers to very recent approaches. As running example, we use a information-extraction agent that continuously extracts typed entities and relations from an evolving document stream into a persistent knowledge base, encounters new terminology and relations, receives corrections and conflicting evidence, and uses retained knowledge to influence later extraction. This example grounds important questions: what should be learned, what evidence should justify an update, where the change should reside, and how it should be validated, governed, and evaluated over time. Particular emphasis is placed on persistent state, learning substrates, provenance, feedback loops, rollback, regression detection, forgetting, temporal generalization, and trajectory-based evaluation. The material for this tool is available at: github.com/megagonlabs/pdsi-tutorial .

1 Motivation and Scope

NLP is moving from single-turn prediction toward persistent language agents that retrieve documents, call tools, maintain memory, interact with users, and act over extended time horizons. Their environments are inherently non-stationary: knowledge sources and terminology change, APIs evolve, and new failure modes emerge after release. Work on retrieval, tool use, memory, reflection, post-training, and agent evaluation has consequently made adaptation a central problem for NLP (Lewis et al., 2020; Schick et al., 2023; Yao et al., 2023; Shinn et al., 2024; Salama et al., 2025).

The tutorial goes beyond models that can adapt within their current context: it explores whether a deployed system can use experience to improve its subsequent performance, and retain what it has learned across interactions. A system may retrieve an updated document without retaining anything from the episode; conversely, it may keep its weights frozen while learning a reusable extraction pattern, concept, procedure, or routing rule. The scientific object is therefore the evolving compound agent rather than only a checkpoint.

To keep this argument concrete, we use a fictional information-extraction agent, A_1, throughout. $A1$ reads scientific and technical documents and extracts typed entities and relations into an evolving knowledge base (KB_1). It is inspired by the closed-loop pattern exemplified by DySECT, where extraction populates a KB and accumulated knowledge feeds back to improve later extraction (Amin-Naseri et al., 2026). Our example extends that pattern to corrections, provenance, ontology revision, conflicting evidence, update gating, regression testing, and rollback, and by conneting these aspects to recent approaches described in the literature.

This perspective connects modern agents with lifelong learning's emphasis on transfer (Thrun, 1998), continual learning's concern with change and preservation (De Lange et al., 2022), and never-ending learning's sustained acquisition in systems whose current knowledge shapes what they learn next (Carlson et al., 2010; Mitchell et al., 2018).

Proceedings of the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 15th International Joint Conference on Natural Language Processing: Tutorial Abstracts, pages 27–31
November 6-10, 2026 ©2026 Association for Computational Linguistics

2 What Counts as Learning?

Self-improving systems are often described through actions, such as generating a critique, retaining a memory, updating a KB, revising a prompt, training or fine-tuning a model. None alone establishes improvement. We distinguish four elements. **Experience** includes documents, extracted candidates, feedback, failures, corrections, and observations. **Persistent state** is information retained across the declared episode or deployment boundary. **Learning substrates** are components that may change, such as memory, retrieval, structured knowledge, instructions, tools, policies, evaluators, or parameters. **Evaluation** determines whether the retained change improves a declared capability without unacceptable regressions.

For A_1, correcting *(Model-X, evaluated-on, Benchmark-Y)* could mean fixing one triple, learning an alias, adding a negative example, revising an ontology relation, changing the extraction prompt, altering retrieval, or updating parameters. These are different learning decisions with different scope and risk.

Following Mitchell (1997), we adopt a performance-based view of learning: a system learns from experience E, with respect to a class of tasks T and performance measure P, when its performance on T, as measured by P, improves with E. For post-deployment language agents, we additionally make persistence explicit: the effect of experience must be retained across the relevant interaction or deployment boundary so that it can influence subsequent behavior. A persistent update alone is therefore not evidence of learning. A KB can accumulate falsehoods, memory can amplify stale patterns, and parameter updates can degrade previously reliable behavior; the learning claim requires evidence of improved subsequent performance.

Never-ending learning further motivates a second question: does accumulated experience improve not only current performance, but also the system's ability to learn subsequent tasks? We therefore consider adaptation speed, evidence efficiency, transfer, preservation, and recovery in addition to endpoint performance.

3 Persistent State and Learning Substrates

Language agents provide more learning substrates than parameter-centric formulations suggest. A_1 makes these alternatives visible.

Memory and structured knowledge. The agent can retain individual corrections and failure cases, consolidate repeated evidence into concepts or rules, and update document indices, entity catalogs, ontologies, and KB facts. Long-term memory work illustrates the general problem of what should be remembered, consolidated, forgotten, or scoped (Maharana et al., 2024; Salama et al., 2025); RAG illustrates the separation between model parameters and external knowledge access (Lewis et al., 2020).

Prompts, examples, policies, and tools. A validated correction can become a few-shot example; an ontology constraint can become a validation rule; repeated ambiguity can trigger abstention or clarification; and the system can acquire procedures for normalization, entity resolution, contradiction checking, or source verification. Tool-use and reflection methods provide mechanisms that can participate in such loops (Schick et al., 2023; Yao et al., 2023; Madaan et al., 2023; Shinn et al., 2024).

Model parameters. Model training (FST, RL, continuous pre-training, etc.) remains important when repeated evidence suggests a broad pattern that should generalize beyond explicit memory or prompting. But parameter updates are one substrate among many and are often harder to attribute, reverse, and audit than localized changes. The recurring question is: *where should this experience be learned?* We compare substrates by persistence, scope, reversibility, generalization, cost, provenance, and the risk that an error propagates into future decisions.

The tutorial also distinguishes **what** is being learned. For A_1, *knowledge* includes facts, entity identities, taxonomic relations, and temporal scope; *skills* include extraction, normalization, entity resolution, and verification procedures; *policies* determine retrieval, routing, abstention, promotion, and recovery; and *self-monitoring* includes uncertainty, contradiction detection, and failure prediction. A single incident may bear on several targets, but the evidence needed to justify each can differ. A corrected triple may justify a local fact update without justifying a new global extraction rule. This separation helps prevent a common failure mode in self-improving systems: generalizing an update beyond the scope actually supported by the evidence. Special attention should be given to the forgetting ability, that is key for the parametric knowledge, but not necessarily the same in a post-deployment system. (where forgetting can prevent evidence accumulation)

4 Running Example: A_1

A_1 begins with a fixed extractor, relation schema, document retriever, and sparse KB_1. It processes documents and produces records such as *(method,*

evaluated-on, benchmark), *(system, uses, model)*, or *(organization, develops, technology)*. Across the tutorial, A_1 encounters a sequence of incidents. Several documents first use a new term for an existing concept: should the agent learn an alias, create a concept, or leave the ambiguity unresolved? Repeated evidence then suggests a relation absent from the ontology: when is schema expansion justified, and should older extractions be revisited? A curator corrects a mistaken relation: is this an instance repair or evidence for a reusable rule? Later, documents support incompatible triples, raising questions about provenance, source authority, temporal validity, and uncertainty.

The key incident comes when KB_1 feeds accumulated knowledge back into extraction. Validated concepts and relations can improve later recognition, as in closed-loop extraction systems such as DySECT (Amin-Naseri et al., 2026). But the same mechanism can create a self-reinforcing error: an incorrect concept biases future extraction, which then produces apparently supporting evidence for that concept. Persistent state has changed not only predictions but also the evidence the system is likely to encounter next.

Finally, authoritative evidence invalidates an earlier belief or ontology decision. A_1 must revise rather than merely append/forget. Successful learning may therefore require making an old belief inactive while retaining provenance needed for historical questions or diagnosis. But, the old belief should not be forgotten, as it might happen that new future evidence can make it valid again. The same incidents are revisited through the lenses of substrates, evidence, governance, and evaluation so that the tutorial develops one cumulative argument.

5 Evidence and Governed Improvement

Persistent adaptation alone is insufficient. An extraction system that promotes every candidate can learn from copied misinformation, correlated documents, noisy feedback, adversarial content, or self-generated evidence that merely echoes its current beliefs. Post-deployment learning therefore requires an explicit transition from experience to evidence to confidence update to an authorized state change.

For A_1, a candidate triple is not automatically knowledge. Relevant evidence includes the source passage, provenance, temporal scope, relation constraints, corroborating or contradicting sources, and review status/confidence. Several pages quoting one original source do not constitute independent corroboration. A curator may be authoritative about a schema decision but not every factual claim. A model-generated explanation may help locate an error without being sufficient evidence for a persistent update.

This framing separates **autonomous update generation** from **autonomous update deployment**. A_1 may propose a relation type, rule, memory, or training example without being authorized to promote it directly. Validation gates, staged release, provenance, rollback, and runtime constraints support bounded autonomy (Wang et al., 2025; Zhu et al., 2025). Closed-loop feedback makes independence especially important: if a mistaken ontology category influences retrieval, prompting, and judging, agreement among those components may reflect a shared error. The correct learning decision can therefore be to quarantine, defer, or reject an update.

Contradiction is likewise not always an error to eliminate. Two sources may describe different time periods, domains, or definitions. A_1 should first align scope before choosing among them, and it may preserve both records with explicit validity conditions. This makes revision different from indiscriminate forgetting: an obsolete active belief can become inactive for current extraction while remaining available for historical reasoning and audit. Evidence can therefore justify retention, revision, rejection, or deliberate non-update.

6 Evaluation Through Time

Post-deployment learning evaluation focuses on whether an evolving system becomes more capable while preserving capabilities that still matter. Snapshot precision, recall, or F1 are necessary but insufficient.

Update effects: whether one retained change improves behavior relative to a matched starting configuration: does learning an alias improve entity linking on unseen documents beyond the corrected instance? **Update-policy effects**: whether rules for proposing and committing changes produce useful trajectories across many incidents: do high-confidence KB_1 additions improve later extraction without amplifying false relations? **Deployment effects**: whether the evolving system remains beneficial under the changing document stream and task definition.

Metrics therefore span extraction quality (precision, recall, calibration, error type), knowledge-state quality (contradictions, provenance, stale facts, ontology coherence), and learning quality (gain, durability, transfer, forgetting, evidence efficiency, regression, and recovery). A larger KB is not itself evidence of better competence, and success on previously cor-

rected cases is not evidence of generalization.

The preservation target must also be explicit. If a relation type has genuinely become obsolete, changing current behavior is not forgetting; if the task still requires both old and new usages, collapsing them is a regression. Likewise, learning speed and cumulative benefit matter: an update policy that eventually reaches the same quality as a frozen baseline may still be valuable if it needs less supervision on later concepts, while one that repeatedly relearns the same distinction has not demonstrated effective cumulative learning.

Temporal evaluation matters when world knowledge or task definitions change (Tang et al., 2025; Jiang et al., 2026). Evaluation data must also be protected from repeated optimization: if documents used to choose ontology updates are recycled as evidence of success, adaptive overfitting can masquerade as continual improvement. Thus, we emphasize held-out or delayed evaluation channels, regression suites, exposure tracking, and explicit accounting for distributional change. And also, which capabilities improved, which remained reliable, at what cost, over what horizon, and under what comparison.

7 Tutorial Structure and Learning Outcomes

Part I – From train-once/deploy-many to learning after deployment (20 min). System boundaries and the difference between transient adaptation and persistent learning, introduced through A_1's static extraction baseline.

Part II – Foundations for learning under change (25 min). Lifelong, continual, and never-ending learning; transfer, stability, plasticity, forgetting, drift, consolidation, and cumulative acquisition.

Part III – Persistent state and learning substrates (30 min). Memory, retrieval, structured knowledge, ontologies, prompts, tools, policies, evaluators, and parameter updates; A_1 illustrates where a correction or new concept should reside.

Part IV – Experience and evidence (25 min). Candidate extractions, corrections, retrieved and synthetic evidence, contradiction, provenance, dependence, and temporal scope.

Part V – Governed self-improvement (30 min). Update proposals, schema changes, validation, authorization, staged deployment, rollback, selective forgetting, and autonomy.

Part VI – Longitudinal evaluation (30 min). Extraction and knowledge-state quality, regression detection, transfer, forgetting, temporal generalization, evaluation exposure, and trajectory-based comparison.

Part VII – A_1 exercise and open problems (10 min). Participants diagnose an extraction failure, choose among an instance correction, memory update, ontology change, policy revision, or model update, specify admissible evidence and validation, and decide whether the proposal should be promoted. A 10-minute break is included separately.

Participants should leave with: (1) a framework for distinguishing learning from transient adaptation and improvement from mere state growth; (2) a vocabulary for deciding what an agent should learn, where it should reside, and how it should be governed; and (3) an evaluation framework for testing whether an evolving agent genuinely improves over time.

8 Breadth, Ethics, and Materials

The tutorial is not about a single IE architecture. A_1 is a pedagogical thread connecting language agents, retrieval and memory, post-training, never-ending and continual learning, knowledge representation, and evaluation under non-stationarity. The principles apply equally when persistent state is stored in memories, documents, code, tools, policies, or parameters.

Post-deployment learning also creates risks: persistent stores can expose private or proprietary information; feedback loops can amplify bias or misinformation; schema changes can erase useful distinctions; and temporary attacks or extraction errors can become durable state. We emphasize data minimization, provenance, access control, temporal validity, validation, rollback, appropriate human or institutional authority, and monitoring for uneven regressions. The objective is inspectable, empirically evaluated adaptation rather than unconstrained self-modification.

Slides, a reading list, conceptual and evaluation maps, and worked exercises will be made available (github.com/megagonlabs/pdsi-tutorial). The tutorial targets NLP/LLM researchers, practitioners working on adaptive or agentic NLP systems, and graduate students familiar with modern neural NLP and LLMs; prior background in continual or never-ending learning is not required.

9 Conclusion

As language models become persistent agents, learning cannot be identified only with parameter updates. A corrected triple or newly discovered concept matters scientifically only when we can explain what persisted, why the evidence justified the change, how it affected later behavior, and whether benefits survived without unacceptable regressions. The tutorial develops this system-level perspective as a bridge between lifelong, continual, and never-ending learning and the challenge of building language agents that can improve responsibly after deployment.

References

Moin Amin-Naseri, Hannah Kim, and Estevam Hruschka. 2026. A Dynamic Self-Evolving Extraction System. arXiv:2603.06915.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2022. A continual learning survey: Defying forgetting in classification tasks. *IEEE TPAMI*, 44(7):3366–3385.

Xunyi Jiang, Dingyi Chang, Julian McAuley, and Xin Xu. 2026. When benchmarks age: Temporal misalignment through large language model factuality evaluation. In *EACL 2026, Volume 2*, pages 500–512.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 33:9459–9474.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Swabha Swayamdipta, Yiming Yang, and Shrimai Prabhumoye. 2023. Self-Refine: Iterative refinement with self-feedback. In *NeurIPS*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *ACL 2024*, pages 13851–13870.

Tom M. Mitchell. 1997. *Machine Learning*. McGraw-Hill.

Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. 2018. Never-ending learning. *Communications of the ACM*, 61(5):103–115.

Rana Salama, Jason Cai, Michelle Yuan, Anna Currey, Monica Sunkara, Yi Zhang, and Yassine Benajiba. 2025. MemInsight: Autonomous memory augmentation for LLM agents. In *EMNLP 2025*, pages 33124–33140.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, 36:68539–68551.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*.

Wei Tang, Yixin Cao, Yang Deng, Jiahao Ying, Bo Wang, Yizhe Yang, Yuyue Zhao, Qi Zhang, Xuan-Jing Huang, Yu-Gang Jiang, et al. 2025. EvoWiki: Evaluating LLMs on evolving knowledge. In *ACL 2025*, pages 948–964.

Sebastian Thrun. 1998. Lifelong learning algorithms. In *Learning to Learn*, pages 181–209. Springer.

Haoyu Wang, Christopher M. Poskitt, and Jun Sun. 2025. AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. arXiv:2503.18666.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *ICLR*.

He Zhu, Tianrui Qin, King Zhu, Heyuan Huang, Yeyi Guan, Jinxiang Xia, Hanhao Li, Yi Yao, Ningning Wang, Pai Liu, et al. 2025. OAgents: An empirical study of building effective agents. In *Findings of EMNLP 2025*, pages 13354–13369.