TRACE: Training and Inference-Time Interpretability Analysis for Language Models

Nura Aljaafari[1†], **Danilo S. Carvalho**[3], **André Freitas**[1,2,3]
[1] Department of Computer Science, University of Manchester, United Kingdom
[2] Idiap Research Institute, Switzerland
[3] National Biomarker Centre, CRUK-MI, Univ. of Manchester, United Kingdom
{firstname.lastname}@[postgrad.][†]manchester.ac.uk

github.com/neuro-symbolic-ai/trace_package Short Video

Abstract

Understanding when and how linguistic knowledge emerges during language model training remains a central challenge for interpretability. Most existing tools are post hoc, rely on scalar metrics, or require nontrivial integration effort, making comprehensive interpretability analysis difficult to deploy and maintain. We introduce **TRACE**, a modular toolkit for training and inference-time interpretability analysis of transformer models. It enables lightweight, in-training analysis of linguistic and representational signals, including features probing, intrinsic dimensionality, Hessian curvature, and output diagnostics. It integrates with **ABSynth**, a controllable synthetic corpus generator that provides structured annotations for precise evaluation of linguistic feature acquisition. Experiments with autoregressive transformers demonstrate that TRACE reveals developmental phenomena such as early syntactic emergence, delayed semantic acquisition, and representational compression, signals overlooked by traditional scalar metrics such as loss or accuracy. With minimal integration effort, the tool enables layer-wise diagnostics, convergence-based early stopping, and detection of structural errors, making transformer analysis interpretable, actionable, and reproducible.

1 Introduction

Interpreting the behaviour of transformer-based language models (LMs) has been gaining interest and importance (Nostalgebraist, 2020; Wang et al., 2023; Hanna et al., 2023; Belrose et al., 2023; Meng et al., 2022), especially with the widespread use of them across various domains. This interpretation is fundamental for verifying their correctness, analysing reasoning processes, and improving robustness (Bereska and Gavves, 2024).

Although various approaches to interpretability have been proposed, most are post hoc: they examine fully trained models using input-output correlations (Kokhlikyan et al., 2020; Lundberg and Lee, 2017; Zhao and Shan, 2024), structured tasks such as mathematics or algorithmic reasoning (Hanna et al., 2023; Nanda et al., 2023), or mechanistic analysis (Belrose et al., 2023; Meng et al., 2022). Tools that operate in training time, such as TensorBoard or Weights & Biases, focus narrowly on scalar metrics like loss or gradient norms. They reveal *whether* a model is learning, but not *what* is being learned or *when* key capabilities emerge. Lightweight, modular tools for tracking representational development remain scarce.

This creates a critical gap: researchers lack accessible tools for comprehensive analysis of *when* and *how* semantic structure emerges, whether during training or in deployed models. As a result, the relationship between optimisation objectives and representational development remains poorly understood, despite its importance for training decisions, generalisation, and debugging failures. This blind spot hinders both scientific understanding of language acquisition in LMs and the practical optimisation of model training workflows.

We introduce **TRACE** (Tracking Representation Abstraction and Compositional Emergence), a modular toolkit for training and inference-time interpretability analysis of transformer models.[1] TRACE exposes internal learning dynamics through lightweight instrumentation requiring minimal code changes, enabling structured tracking of semantic emergence, representational compression, and loss landscape evolution—signals that are invisible to conventional training logs.

To support systematic experimentation, TRACE pairs with **ABSynth**, a controllable synthetic corpus generator that provides structured linguistic data with aligned annotations[1]. While TRACE can analyse any transformer training process, ABSynth enables precise, repeatable experiments grounded in known linguistic structures.

[1]TRACE & ABSynth are released under GPLv3 License.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 806–820
November 4-9, 2025

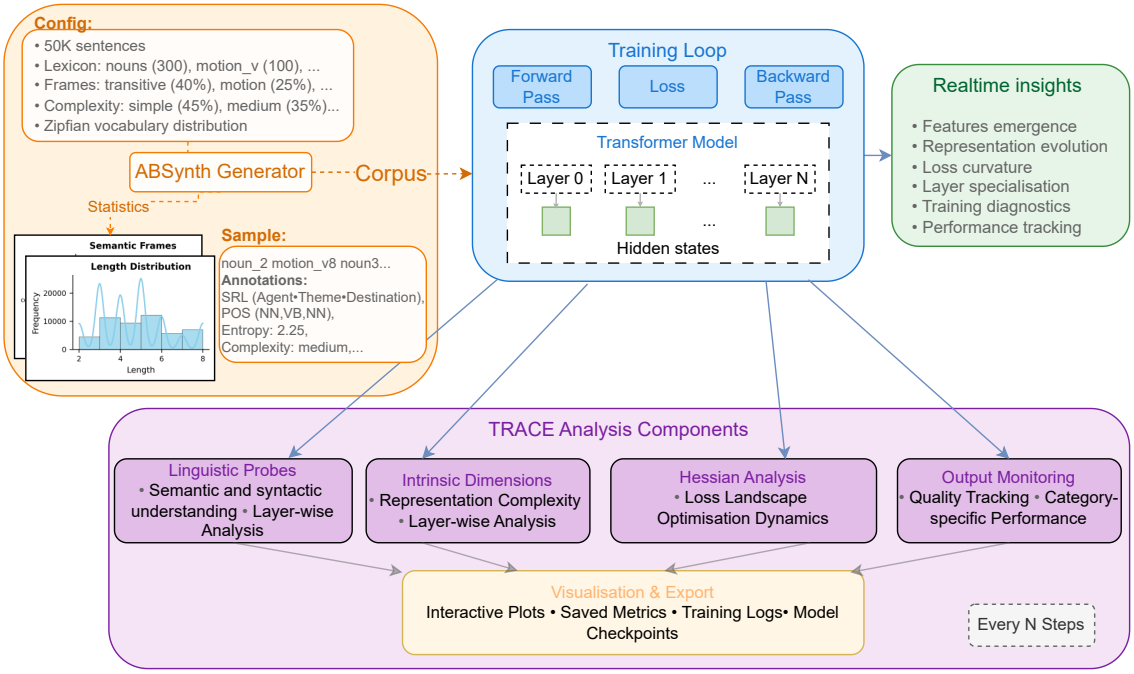


Figure 1: TRACE system overview. ABSynth generates controlled linguistic datasets with explicit annotations, which feed into transformer training loops instrumented with lightweight analysis hooks. Four modular components track linguistic emergence: semantic/syntactic probes, intrinsic dimensionality analysis, Hessian landscape exploration, and output monitoring. All modules generate automated visualisations and can operate independently during training or post-hoc analysis.

TRACE is designed for researchers and developers seeking a modular, easy-to-integrate interpretability tool to better understand, debug, or intervene on transformer models. Demonstrations on decoder-only models trained with ABSynth reveal distinct developmental phases in syntax and semantics, compression of internal representations, and curvature shifts in optimisation dynamics, insights not available through traditional monitoring alone. Thus, it transforms training from an opaque optimisation process into an interpretable, research-ready workflow for studying representational learning and guiding practical model development, with additional support for inference-time analysis.

2 Overview

TRACE (Figure 1) is a modular toolkit for interpretability analysis of transformer training. It supports dynamic monitoring of linguistic, geometric, and optimisation signals, integrating seamlessly into training loops and model evaluation pipelines. It is also model-agnostic, extensible, and requires only minimal code changes. By default, it supports the construction of standard transformer architectures as described in (Vaswani et al., 2017), and adapting to custom models typically requires only minor adjustments. A companion dependency, ABSynth, supplies linguistically annotated training data with controllable complexity.

2.1 Design Overview

The toolkit consists of two core modules and one external dependency:

- **Monitoring Hooks**: Injected into the training loop or inference pipelines, these capture hidden states, gradients, and loss signals without interfering with model execution.
- **Analysis Modules**: Pluggable components for probing, intrinsic dimensionality estimation, loss analysis, and output-level diagnostics.
- **ABSynth (external dependency)**: A synthetic corpus generator that provides structured, annotated training data with controllable linguistic complexity.

2.2 Key Capabilities

TRACE combines four capabilities that are difficult to achieve simultaneously within existing interpretability frameworks:

- **Minimal Integration Effort:** Requires only a few lines of code to enable comprehensive analysis, with modular components that can be added, removed, or extended without code refactoring.
- **Flexible Analysis Scope:** Supports both live analysis during training and post-hoc inspection of trained models, continuously monitoring linguistic probes, representational geometry, and loss curvature.
- **Automated Diagnostics:** Automatically plots learning curves, phase changes, and divergence indicators, with outputs saved in structured formats (e.g. JSON, CSV) for further analysis.
- **Actionable Insights:** Reveals how design and training choices shape model dynamics. Model size influences stabilisation patterns, often correlating with intrinsic dimensionality; removing components can increase curvature volatility without preventing generalisation. Hyperparameter changes may accelerate probe gains but often lead to sharper curvature. Such contrasts provide concrete guidance for early stopping, architecture tuning, and training schedule adjustments.

3 Controlled Corpus Generation (ABSynth)

ABSynth is a synthetic corpus generator that produces linguistically annotated datasets with explicitly controlled syntactic and semantic structures. Inspired by frame semantics (Baker et al., 1998; Fillmore, 1982), ABSynth models language as a composition of semantic frames, participant roles, and lexical units. Unlike natural corpora, which entangle linguistic properties or lack annotations, ABSynth provides fine-grained control over linguistic dimensions while generating English-like data. This enables systematic experiments on representational learning, generalisation, and abstraction.

Minimal API and Basic Usage

ABSynth supports rapid corpus generation with a simple interface:

```
from absynth.corpus import
    SyntheticCorpusGenerator
generator = SyntheticCorpusGenerator()
corpus = generator(10000)
corpus.save("corpus_full.json", indent=2)
```

A typical generated sentence appears as:

```
noun139    transitive_verb8s    noun40
preposition4 location2
```

This sentence instantiates the `transitive_action` semantic frame with structured annotations automatically generated during corpus creation.

Sentence Structure and Generation

Each sentence is constructed from a semantic frame containing a syntactic template. Frames specify roles (e.g., Agent, Patient, Location), which are mapped to lexical items from Zipfian-distributed pools (Zipf, 1949; Piantadosi, 2014). Templates determine constituent order (e.g., `[arg0, verb, arg1, prep, arg2]`) and syntactic structure. ABSynth automatically generates token-level annotations, including part-of-speech tags, semantic roles with positions, and metadata as follows:

- **Semantic Roles:**
  - `noun139` — Agent (position 0)
  - `noun40` — Patient (position 2)
  - `location2` — Location (position 4)
- **POS Tags:** `[NN, VB, NN, IN, NN]`
- **Metadata:** Complexity = `medium`; Entropy = 2.25; Length = 5

Corpus properties can be customised via frame and complexity distributions:

```
corpus = generator.generate_corpus(
    num_sentences=10000,
    complexity_distribution={
        "simple": 0.55, "medium": 0.35,
        "complex": 0.1},
    semantic_frame_distribution={
        "transitive_action": 0.4,
        "motion": 0.3,
        "intransitive_action": 0.3})
```

Key controllable dimensions include: (i) **Lexical selection:** Tokens sampled from role-specific pools with Zipfian (Zipf, 1949; Piantadosi, 2014) distributions; (ii) **Frame selection and creation:** Controls argument structure and semantic relations; (iii) **Complexity distribution:** Specifies proportions of simple, medium, and complex constructions; and (iv) **Statistical properties:** Entropy profiles, collocational strengths, and predictability patterns.

4 Dynamic Linguistic Tracking (TRACE)

As a sample use case, we present results from training a 2-layer decoder-only transformer with 3 attention heads, 384-dimensional MLP, and 96-dimensional hidden states on an ABSynth-generated corpus of 50K examples (batch size 128, learning rate $1e-3$, 70K steps). All analysis modules were enabled: probing, intrinsic dimensionality, Hessian curvature, and output diagnostics. While we show a combined diagnostic plot for conciseness, each module also outputs standalone visualisations and structured logs. A complete walkthrough example of the tool usage, including detailed inputs, outputs, and interpretations, is provided in Appendix C.

API Integration

TRACE wraps a standard training loop with minimal configuration:

```
from trace.training import Trainer,
    TrainingConfig
config = TrainingConfig(
    epochs=10,
    track_interval=500,
    save_visualization=True
)
trainer = Trainer(config, tokenizer, model)
trainer.train(train_loader, val_loader)
```

All modules run independently, log structured outputs (e.g., JSON, CSV), and support training and post-hoc visualisation.

Linguistic Structure Probing. Semantic and syntactic probes can be applied to any layer, supporting both static and dynamically updated models. Probes return confidence per linguistic role, revealing where and when linguistic features emerge and consolidate.

```
config = TrainingConfig(
    track_semantic_probes=True,
    probe_load_paths={
        (0, 'decoder'):
            './probes/pos_layer0.pt',
    }
)
```

Figure 2a shows semantic probe confidence for the decoder layer 1. Core roles (e.g., AGENT, ACTION) stabilise early, while adjunct roles (e.g., LOCATION, DESTINATION) fluctuate, indicating delayed consolidation. Several dips in confidence align with changes in curvature and intrinsic dimensionality (Figure 2b), suggesting linked representational shifts. The key capabilities of this module include:

- **Multi-label probing:** Simultaneous detection of multiple linguistic features
- **Category-specific analysis:** Performance breakdown by syntactical and semantic categories
- **Emergence tracking:** Identification of critical learning phases for different linguistic structures

Intrinsic Dimensionality Analysis. Representational complexity is estimated through intrinsic dimensionality (ID) metrics, including TwoNN (Facco et al., 2017) and PCA-based estimators (Cangelosi and Goriely, 2007):

```
config = TrainingConfig(
    track_intrinsic_dimensions=True,
    id_method="TwoNN")
```

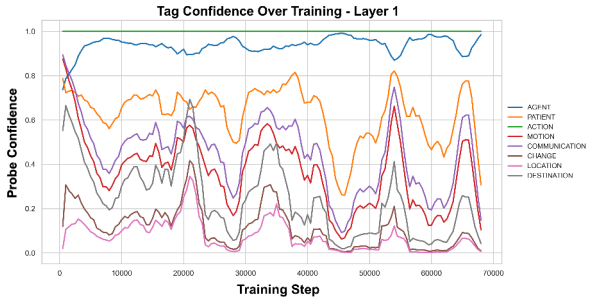
ID can be reported per layer or model average, capturing when and where representation compression or expansion occurs across model layers. Figure 2b shows an early drop in average dimensionality followed by a sharp rebound and stabilisation. These trends suggest a shift from early overcompression to stabilised abstraction. Such transitions coincide with early probe confidence dips, supporting the view that dimensionality evolution reflects changes in internal representational. These measures can be used to detect abstraction shifts, potential representational bottleneck or underutilised representational capacity.

Optimisation Landscape Analysis. Loss curvature is tracked using Lanczos-based Hessian approximations (Lanczos, 1950). The framework captures gradient–Hessian alignment, dominant eigenvalue shifts, trace, and other spectral metrics:

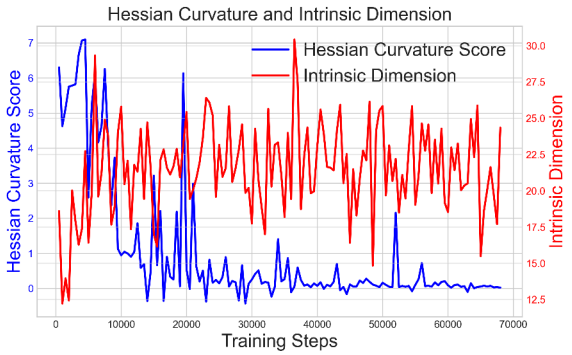
```
config = TrainingConfig(
    hessian_n_components=10,
    track_component_hessian=True,
    track_gradient_alignment=True)
```

In Figure 2b, early spikes in curvature precede shifts in ID, revealing a transition from sharp loss regions to flatter ones, memorisation and generalisation. Additional moderate spikes later in training suggest ongoing adjustments during fine-tuning phases. Beyond this visualisation, the tool simultaneously monitors other optimisation metrics, including:

- **Curvature evolution:** Changes in loss landscape sharpness and flatness, trace, and dominant eigenvalues.



(a) Probe confidence over semantic roles



(b) Intrinsic dimension and curvature evolution

Figure 2: **Sample TRACE visual diagnostics.** (a) Semantic probe confidence per role at decoder layer 1. TRACE tracks dynamic shifts, revealing phase transitions in linguistic encoding (e.g., mid-training dips and late recovery). (b) Joint evolution of Hessian curvature and average intrinsic dimensionality, exposing compression, abstraction, and optimisation phase shifts. Together, these metrics offer temporal insights missed by scalar loss alone.

- **Component analysis:** Separate tracking of components (e.g. attention, feed-forward) contributions
- **Memorisation detection:** Train/validation landscape divergence analysis
- **Gradient alignment:** Relationship between optimisation direction and principal curvature

Output-Level Diagnostics. Model predictions are analysed by token class and semantic role, enabling stratified accuracy reporting across linguistic categories. This diagnostic helps detect linguistic structural misalignment, where the model predicts the correct type (e.g., a noun or a patient role) but fails to recover the intended lexical item (e.g., `noun3` vs. `noun542`). Such errors indicate representational drift: the model learns the general grammatical form but lacks alignment to specific semantic slots or referents.

```
config = TrainingConfig(
    track_semantic_roles=True,
    semantic_roles_granularity='detailed' )
```

System Evaluation. This demonstration serves as a functional evaluation of TRACE. The example training run reveals distinct, interpretable dynamics across modules, semantic emergence in probe confidence, representational compression through intrinsic dimensionality drops, and curvature shifts in the optimisation landscape. The observed alignment between independent metrics further validates the utility of TRACE for diagnosing learning phase transitions. Full training configuration is provided in Appendix A, and further comparisons with existing interpretability tools appear in Table 1.

Research Questions and Applications. TRACE addresses fundamental questions in transformer interpretability: (i) **Emergence timing:** When and where do syntactic and semantic features first appear?; (ii) **Representational evolution:** Which layers compress or expand representational space during training?; (iii) **Generalisation patterns:** Is the model developing structural understanding or overfitting to surface patterns?; and (iv) **Training efficiency:** Can representation convergence inform early stopping decisions?

Automatic Visualisation and Reporting. All modules generate comprehensive visualisations automatically, including: (i) **Linguistic confidence evolution:** Per-category confidence scores across training steps; (ii) **Dimensionality trajectories:** Layer-wise ID evolution with compression detection; (iii) **Hessian landscape analysis:** Eigenvalue evolution and curvature dynamics; and (iv) **Performance breakdown:** Category-specific accuracy trends and structural alignment metrics. Results are also saved in standard formats (e.g., CSV, JSON) for further analysis and integration with external tools. By exposing model internals at different stages, TRACE makes transformer training interpretable, inspectable, and diagnostic by design.

5 Related Work

Post-hoc Representational Analysis. A common approach to interpreting language models involves probing classifiers or sparse autoencoders

Feature	TRACE (Ours)	Post-hoc Probing	Manual PCA / ID	TransformerLens
Timing of Analysis	During and after training	After training	After training	During and after training
Layer-wise Tracking	Yes	Yes	Yes	Yes
Temporal Resolution	High (per N steps)	Sparse (checkpoints)	Sparse	Sparse (checkpoints)
Model Support	Any custom PyTorch model	Any checkpoint	Any checkpoint	HuggingFace models only (limited)
User Effort	Low (1 config file)	High (custom scripts)	High	High (custom pipelines)
Causal Intervention Support	Partial (custom scripting required)	No	No	Partial (custom scripting required)
Emergence Detection	Yes	No	No	No
Training Support	Yes (native)	No	No	Partial (custom scripting required)
Supports Early Stopping	Yes (live signals)	No	No	No
Gradient/Loss Monitoring	Yes	Indirect	N/A	Yes

Table 1: Comparison of TRACE with post-hoc interpretability methods and TransformerLens. TRACE offers low-effort, modular interpretability with native training integration, unlike existing tools that require custom scripting and lack temporal tracking.

(SAEs) trained on frozen representations. Probes evaluate the presence of linguistic features using lightweight supervised models (Hewitt and Manning, 2019; Belinkov et al., 2018), while SAEs attempt to recover features in latent spaces in an unsupervised setting (Bricken et al., 2023; Kantamneni et al., 2025). While both techniques can be adapted for use during training, they are typically applied post hoc and require manual effort for their models construction, retraining, and integration. As standalone tools, they provide limited visibility into when features emerge or how they evolve.

Attribution and Causal Methods. Attribution tools (e.g., integrated gradients (Sundararajan et al., 2017) and attention flow analysis (Voita et al., 2019)) and causal interventions (e.g., activation patching (Meng et al., 2022) and path patching (Wang et al., 2023)) identify critical inputs or subcircuits. However, they operate on static checkpoints and provide no insight into the dynamics of representational development.

Model Inspection Frameworks. Frameworks like BertViz (Vig, 2019), InterpretDL (Li et al., 2022), and TransformerLens (Nanda and Bloom, 2022) enable weight and attention analysis. TransformerLens partially supports training-time instrumentation, but requires nontrivial modification. Platforms like TensorBoard or Weights & Biases (Biewald, 2020) track scalar metrics, offering no view into linguistic or geometric structure.

Controlled Datasets and Linguistic Benchmarks. Several datasets evaluate LMs' capabilities. Static benchmarks such as GLUE (Wang et al., 2018), BLiMP (Warstadt et al., 2020), and COGS (Kim and Linzen, 2020) evaluate linguistic competence but lack structural flexibility. CounterFact (Meng et al., 2022) and IOI (Wang et al., 2023) target specific phenomena, while mathematical datasets (Power et al., 2022; Saxton et al., 2019) offer training control but limited linguistic coverage. ABSynth provides extensible, richly annotated data aligned with TRACE's dynamic instrumentation.

Comparison with Existing Tools. Table 1 highlights how TRACE differs from typical interpretability pipelines and frameworks. It supports native training-time analysis, live monitoring, and plug-and-play extensibility with structured logging.

6 Extensibility and Integration

TRACE is implemented around a modular design where models are explicitly constructed within the framework rather than wrapped directly from external libraries. While models from Hugging Face and similar repositories expose high-level APIs, their internal states (e.g., layer outputs, normalization points, or intermediate activations) vary across implementations, making uniform integration difficult. To ensure consistent access to hidden representations and loss signals, TRACE provides a set of composable building blocks (e.g. encoder, decoder, attention, feed-forward modules) in `components.py`. New architectures can therefore be added by instantiating or extending these modules, while still benefiting from a standardised interface for monitoring and analysis. This approach requires users to

define their model structure within TRACE, but it guarantees compatibility with all analysis modules (probes, intrinsic dimensionality, Hessian curvature) without modification. In practice, this design balances flexibility for custom research architectures with stability in the interpretability pipeline. We leave direct integration with Hugging Face models as future work.

```
from trace.components import Encoder, Decoder
from trace.models import Transformer
from trace.trainer import Trainer,
    TrainingConfig

# Define custom encoder-decoder model
encoder = Encoder(num_layers=6, d_model=512,
    num_heads=8, d_ff=2048)
decoder = Decoder(num_layers=6, d_model=512,
    num_heads=8, d_ff=2048)
model = Transformer(encoder=encoder,
    decoder=decoder, vocab_size=32000)

# Configure TRACE analyses
config = TrainingConfig(
    track_semantic_probes=True,
    track_intrinsic_dimensions=True,
    track_hessian=True,
    track_interval=500,
)

trainer = Trainer(config, tokenizer, model)
trainer.train(train_loader, val_loader)
```

This example illustrates how a user can construct a model from TRACE components and automatically enable monitoring tools during training.

7 Conclusion

We presented **TRACE**, a modular toolkit for interpretability of language models, and **ABSynth**, a companion corpus generator for controlled linguistic experimentation. TRACE enables in-training analysis of linguistic, geometric, and optimisation signals, shifting interpretability from a post-hoc task to a dynamic diagnostic process. Applied to a decoder transformer, TRACE revealed phase-structured training dynamics, including semantic emergence, representational compression, and curvature shifts, not visible through traditional training metrics. These signals also enable targeted interventions such as early stopping, architecture tuning, and training schedule adjustment. The tool's modular design allows each component to operate independently or in combination, enabling a broad range of interpretability research. While current demonstrations use synthetic corpora, TRACE can generalise to natural language models. Planned extensions include Hugging Face integration and support for automatic annotation via tools such as NLTK, further broadening TRACE's scope for real-world applications. By making internal learning dynamics observable and actionable, TRACE advances both theoretical insight and practical control over LM analysis.

Limitations

While TRACE advances training-time interpretability, several limitations remain. Our demonstrations primarily rely on ABSynth, which provides controlled conditions but does not capture the full complexity of natural corpora. Although TRACE can be trained on arbitrary text data, it does not currently provide built-in annotation pipelines for linguistic probes; users must preprocess data or connect external tools to supply supervision. Hessian-based curvature estimates are efficient for medium-scale models but remain costly for very large architectures. Finally, TRACE highlights correlations between representational, linguistic, and optimisation signals, but does not establish causality. These limitations point to natural directions for future work, including broader integration with natural-language annotations, more scalable curvature analysis, and causal interventions built on TRACE's monitoring capabilities.

Acknowledgements

This work was partially funded by the SNSF project RATIONAL (200021E_229196), the CRUK National Biomarker Centre, and supported by the Manchester Experimental Cancer Medicine Centre and the NIHR Manchester Biomedical Research Centre.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *COLING 1998 Volume 1: The 17th International Conference on Computational Linguistics*.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2018. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. *arXiv preprint arXiv:1801.07772*.

Nora Belrose, Zach Furman, Logan Smith, Danny Halawi, Igor Ostrovsky, Lev McKinney, Stella Biderman, and Jacob Steinhardt. 2023. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*.

Leonard Bereska and Stratis Gavves. 2024. Mechanistic interpretability for AI safety - a review. *Transactions on Machine Learning Research*. Survey Certification, Expert Certification.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, and 6 others. 2023. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Richard Cangelosi and Alain Goriely. 2007. Component retention in principal component analysis with application to cdna microarray data. *Biology direct*, 2:1–21.

Elena Facco, Maria d'Errico, Alex Rodriguez, and Alessandro Laio. 2017. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific reports*, 7(1):12140.

Charles J. Fillmore. 1982. Frame semantics. In *Linguistics in the Morning Calm*, pages 111–137. Hanshin Publishing Co., Seoul.

Michael Hanna, Ollie Liu, and Alexandre Variengien. 2023. How does GPT-2 compute greater-than?: Interpreting mathematical abilities in a pre-trained language model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

John Hewitt and Christopher D Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. 2025. Are sparse autoencoders useful? a case study in sparse probing. *arXiv preprint arXiv:2502.16681*.

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and 1 others. 2020. Captum: A unified and generic model interpretability library for pytorch. *arXiv preprint arXiv:2009.07896*.

Cornelius Lanczos. 1950. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of research of the National Bureau of Standards*, 45(4):255–282.

Xuhong Li, Haoyi Xiong, Xingjian Li, Xuanyu Wu, Zeyu Chen, and Dejing Dou. 2022. Interpretdl: Explaining deep models in paddlepaddle. *Journal of Machine Learning Research*, 23(197):1–6.

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 36. ArXiv:2202.05262.

Neel Nanda and Joseph Bloom. 2022. Transformerlens. `https://github.com/TransformerLensOrg/TransformerLens`.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. 2023. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*.

Nostalgebraist. 2020. interpreting GPT: the logit lens.

Steven T Piantadosi. 2014. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic bulletin & review*, 21:1112–1130.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. 2022. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *International Conference on Learning Representations*.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. Blimp: The benchmark of linguistic minimal pairs for english. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Zhixue Zhao and Boxua Shan. 2024. "reagent: Towards a model-agnostic feature attribution method for generative language models". In *Proceedings of AAAI Workshop on Responsible Language Models*.

George Kingsley Zipf. 1949. *Human Behavior and the Principle of Least Effort*. Addison-Wesley.

A Experimental setup

A.1 Software Environment and Dependencies

All experiments were run on a single NVIDIA RTX A6000 GPU. The environment was built using Python 3.11.13 and configured via Conda. Core libraries include PyTorch (v2.7.1) and ABSynth (v0.1.1) for synthetic dataset generation (with the same requirements listed here). Other tools include scikit-learn (v1.7.0), SciPy (v1.15.3), NumPy, and scikit-dimension (v0.3.4). Visualisation components rely on Matplotlib (v3.10.3), Matplotlib-inline (v0.1.7), and Seaborn (v0.13.2). Additional packages include tqdm (v4.67.1). The full setup is specified in the environment.yml file included with the code repository and is fully reproducible.

A.2 Model Architecture

We use a lightweight Transformer architecture to enable rapid training and frequent checkpointing. This allows TRACE to track representational evolution with minimal overhead.

- **Architecture:** Decoder-only Transformer
- **Layers:** 2 decoder layers
- **Hidden size:** 96
- **Feed-forward dimension:** 384
- **Attention heads:** 3
- **Maximum sequence length:** 16

A.3 Training Configuration

- **Corpus:** 25,000 synthetic sentences
- **Batch size:** 128
- **Epochs:** 500
- **Learning rate:** 1×10^{-4}
- **Optimizer:** Adam
- **Tracking frequency:** Every 500 steps

B Licensing

TRACE is released under the GNU General Public License v3.0 (GPLv3). This license ensures that the software remains free and open-source. Users are free to use, modify, and distribute the code, provided that derivative works also adopt the GPLv3 license.

C Example Use Case

This appendix illustrates TRACE's functionality and interpretability workflow. It includes an end-to-end case study using a transformer model trained on an ABSynth-generated corpus. This walkthrough demonstrates how TRACE can be used to monitor semantic emergence during training and interpret the resulting behaviour via multiple analytical lenses.

C.1 ABSynth Data Generation

We provide an example of data generation, where we set all the parameters, but **emphasise** that the user has the liberty to use the default configurations by specifying only the number of examples needed, as shown in Section 3.

C.1.1 ABSynth Input Configuration

```
from absynth.lexicon import Vocabulary, LexiconGenerator

# Define vocabulary sizes
vocab = Vocabulary({
    "noun":300, "transitive_verb":40, "intransitive_verb":25,
    "communication_verb":20, "motion_verb":20, "change_verb":15, "adjective":40,
    "adverb":25, "location":150, "temporal":35, "instrument":25, "preposition":15,
    "conjunction":10, "determiner":8
})

lexicon = LexiconGenerator(
    vocab_sizes=vocab,           # Custom vocabulary sizes
    num_clusters=5,              # Number of semantic clusters to create
    zipfian_alpha=1.05,             # Alpha parameter for Zipfian distribution
    error_bias=0.00001,              # Error bias for word generation
    random_seed=42               # For reproducible generation
)

from absynth.sentence import SentenceGenerator, FrameManager
templates = FrameManager()
sentence_generator = SentenceGenerator(lexicon, templates)

from absynth.corpus import SyntheticCorpusGenerator
generator = SyntheticCorpusGenerator(lexicon=lexicon, sentence_generator=sentence_generator)
corpus = generator.generate_corpus(
    num_sentences=25000,
    complexity_distribution={"simple": 0.55, "medium": 0.35, "complex": 0.10},
    semantic_frame_distribution={
        "transitive_action": 0.1,
        "transitive_with_location": 0.15,
        "motion_with_source": 0.15,
        "temporal_action": 0.15,
        "instrumental_action": 0.15,
        "multi_action": 0.15,
        "temporal_complex": 0.15,
    }
)
from absynth.visualization import Visualizer
visualizer = Visualizer(log_dir='./plots')
visualizer.visualize(corpus)
```

This setting creates a corpus of 25K sentences, with a mix of syntactic templates and semantic frames. Sentences vary in their surface structure but preserve underlying argument structures aligned with FrameNet-like role schemas.

C.1.2 Sample Generated Data

A representative sentence from the generated corpus is:

```
Input: "noun139 transitive_verb8s noun40 preposition4 location2"
```

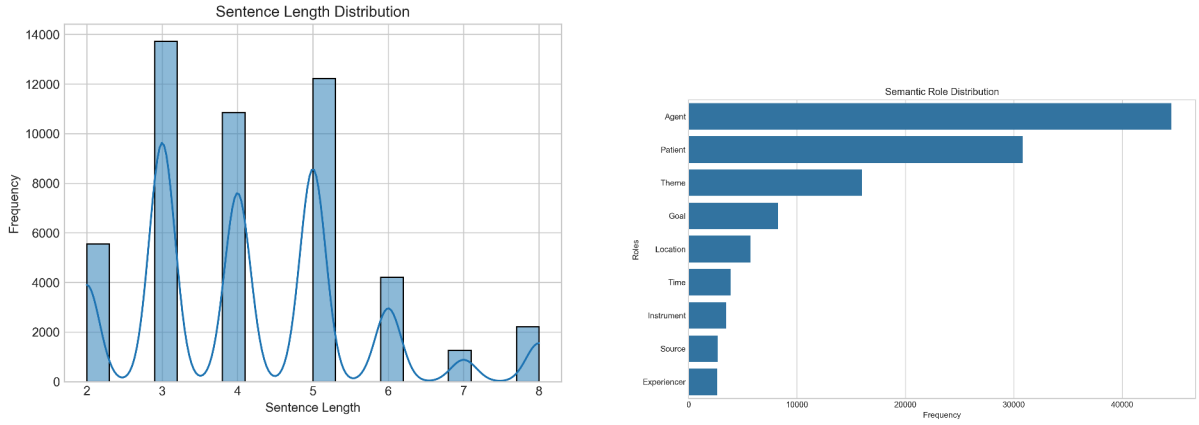
The corresponding annotations include both structural and semantic metadata:

```
{
  "sentence": "noun139 transitive_verb8s noun40 preposition4 location2",
  "semantic_roles": {
    "noun139": {"role": "Agent", "position": 0},
    "noun40": {"role": "Patient", "position": 2},
    "location2": {"role": "Location", "position": 4}
  },
  "pos_tags": ["NN", "VB", "NN", "IN", "NN"],
  "metadata": {
    "complexity": "medium",
    "frame": "transitive_action",
    "length": 5,
    "entropy": 2.25
  }
}
```


This format enables fine-grained probing of semantic representations during training and evaluation. Each token is associated with a role and position, and every sentence is traceable to its underlying generation rule.

C.1.3 Corpus Statistics and Visualisation

Figure 3, and Table 2 represents a sample of the visualisation and statistical summarisation that can be generated for each corpus.



(a) Hessian trace over training steps.

(b) Gradient and Hessian norm dynamics.

Figure 3: (Sample statistical visualisation of ABSynth: (Left) Distribution of sentence lengths in the generated ABSynth corpus. (Right) Frequency distribution of semantic roles across the dataset.

Metric	Value
Total sentences	25,000
Vocabulary size	7910 tokens
Average sentence length	4.17012 tokens
Semantic frame coverage	6 distinct frames
Zipfian compliance	$\alpha = 1.05$

Table 2: Sample ABSynth corpus summary statistics.

C.2 Model Training Setup

C.2.1 Model Configuration

To evaluate TRACE's interpretability modules in a controlled training setting, we trained a lightweight decoder-only Transformer on the ABSynth corpus introduced in Section 4. This setup enables fine-grained

temporal analysis of representational and optimisation dynamics. While this demonstration focuses on decoder-only architectures, we note that TRACE also supports encoder-only and encoder-decoder models.

```
model_config = TransformerConfig(
    model_type="decoder_only",
    vocab_size=7000,
    d_model=96,              # Hidden dimension
    num_heads=3,             # Attention heads
    num_decoder_layers=2,    # Number of layers
    d_ff=384,               # Feed-forward dimension
    max_seq_length=16,      # Maximum sequence length
    dropout=0.1
)
```

C.2.2 Training Configuration

```
training_config = TrainingConfig(
    epochs=30,
    learning_rate=1e-3,
    batch_size=128,

    # Enable all analysis modules
    track_hessian=True, # Loss landscape analysis
    track_linguistic_probes=True, # POS understanding
    track_semantic_probes=True, # Semantic role understanding
    track_intrinsic_dimensions=True, # Representation dimensionality
    track_pos_performance=True,  # Output POS accuracy
    track_semantic_roles_performance=True,  # Output semantic accuracy
    probe_load_paths=probe_paths,
    semantic_probe_load_path=semantic_probe_paths,

    # Analysis frequency
    track_interval=500,      # Every 500 steps
    save_visualization=True
)
```

C.3 TRACE Analysis Results

This section presents representative outputs from TRACE's core modules, applied to the synthetic training run described in Sections C.1–C.2. These examples demonstrate how TRACE captures distinct learning dynamics across layers, linguistic abstractions, and training stages. All inputs and outputs are automatically logged by TRACE when the corresponding tracking module is enabled by the user.

C.3.1 Linguistic Probe Evolution

We trained semantic role probes on model checkpoints throughout training to extract interpretable structure from hidden representations. The probes are trained to predict semantic roles and POS based on intermediate activations. This allows us to track when and where in training various linguistic categories emerge.

Input: Hidden states h from decoder layers 0 and 1, sampled at regular training intervals (500 in this experiment).

Output Example: Figure 4 shows probe confidence scores for Layer 1 across training steps. Confidence values indicate the strength of alignment between hidden states and semantic roles.

Interpretation:

- **Core Role Emergence:** AGENT, ACTION, and PATIENT show rapid gains in probe confidence, stabilising early in training. These roles are central to the predicate-argument structure, and their early emergence suggests the model internalises core semantics first.
- **Adjunct Role Delay:** LOCATION and other adjunct roles show lower and more delayed gains, indicating that peripheral semantic categories are acquired later and less robustly.

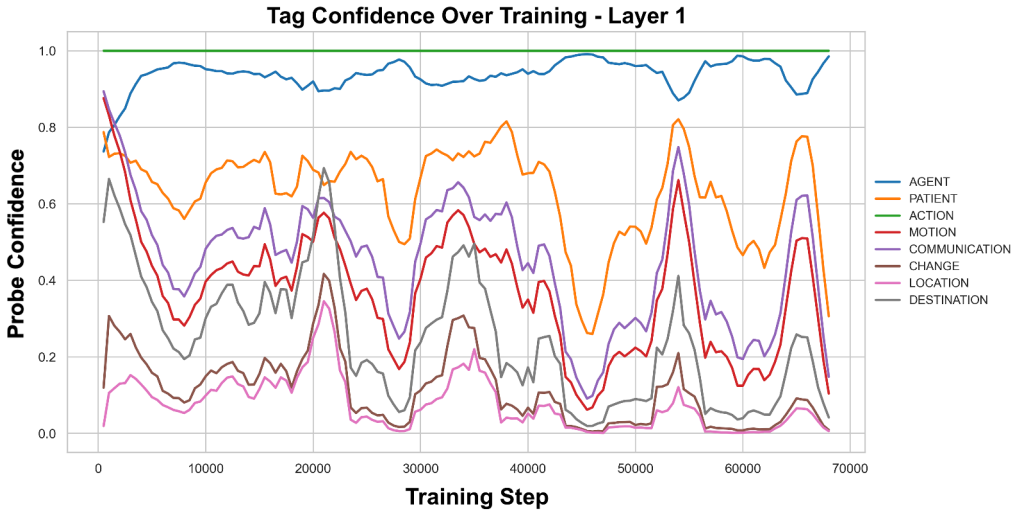


Figure 4: Semantic role probe scores across training steps. Layer 1 shows alignment with core roles over time with steps where there was higher alignment and other for lower.

- **Fluctuation Signals Reorganisation:** Adjunct roles exhibit greater fluctuations in confidence across training steps. These fluctuations may reflect ongoing representational reorganisation as the model adjusts the alignment of semantic roles across layers and training stages.
- **Semantic Acquisition Hierarchy:** The emergence pattern suggests a learning hierarchy, with core argument structure learned before modifiers, consistent with theoretical linguistics and prior neural acquisition work.
- **Alignment with Representational Dynamics:** Confidence dips at intermediate steps (e.g., 10k–25k) align with known transitions in intrinsic dimensionality and Hessian curvature, reflecting internal restructuring as the model shifts from memorisation to abstraction.

C.3.2 Intrinsic Dimensionality Analysis

Input: Hidden states h from decoder layers 0 and 1, sampled at regular training intervals (500 in this experiment).

Output Example: Figure 5 and Table 3 show a sample of the ID analysis output, the figure shows the overall ID over the whole course of training steps, while the table shows a sample of the results. They illustrate the evolution of representational complexity.

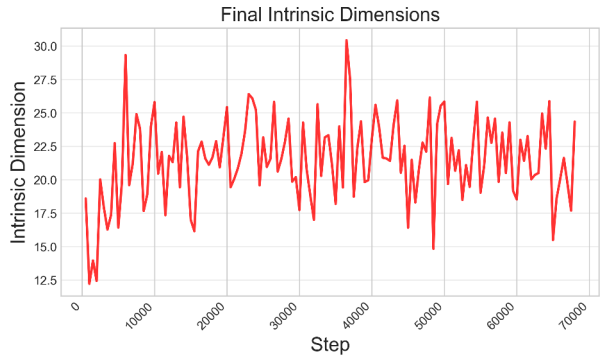


Figure 5: Intrinsic dimensionality across training steps.

Training Step	Avg. ID
500	18
2,000	12
5,000	16
10,000	25
25,000	23

Table 3: Sample averaged Intrinsic dimensionality over layers across training steps.

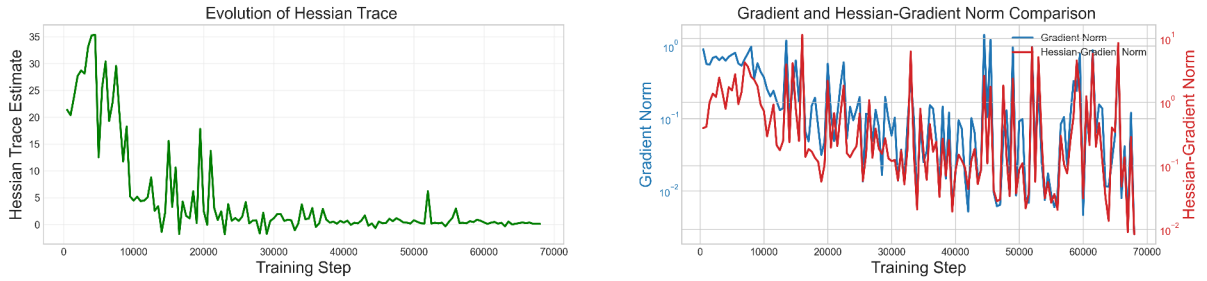
Interpretation:

- **Initial Compression and Expansion Pattern:** Early and sharp drop, 18 to 12 (steps 500-2,000) followed by and expansion to 25, patterns indicating transitioning from simple pattern matching to more complex linguistic understanding.
- **Dynamic Training Regime:** Persistent oscillations between ID 17-25 throughout training suggest active representational restructuring rather than static convergence, reflecting adaptive complexity based on linguistic content.
- **Cross-Metric Transitions:** Notable jumps in intrinsic dimensionality coincide with decreases in Hessian trace and fluctuations in probe confidence. These temporal alignments suggest coordinated signals between them.

C.3.3 Hessian Landscape Analysis

Input: Loss gradients and second-order information calculated by our tool.

Sample output: Figure 6 demonstrates the evolution of Hessian-based metrics throughout training, revealing characteristic patterns of curvature dynamics and spectral properties.



(a) Hessian trace over training steps.

(b) Gradient and Hessian norm dynamics.

Figure 6: Evolution of curvature and norm characteristics during training.

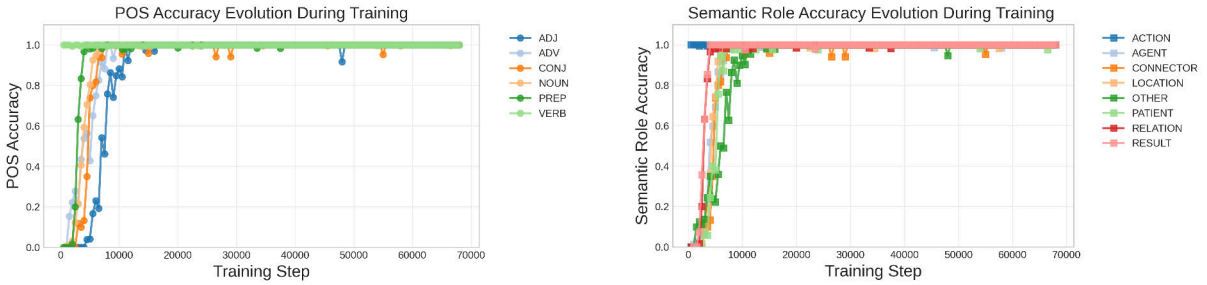
Interpretation:

- **Landscape Evolution Dynamics:** The Hessian trace exhibits a pronounced early peak followed by steady decrease, reflecting the model's transition from highly curved regions associated with memorisation to flatter regions indicating abstraction and generalisation. This is common in randomly initialised networks.
- **Curvature–Dimensionality Duality:** An interesting pattern is observed between curvature and intrinsic dimensionality: periods of rising dimensionality coincide with drops in the Hessian trace, suggesting that representational expansion is facilitated by local flattening in the optimisation landscape.
- **Structural Transition Markers:** Isolated spikes in Hessian curvature, e.g. around 55k steps, serve as indicators of local representational restructuring. These events temporally co-occur with inflection points in probing metrics and effective rank changes, underscoring the diagnostic value of second-order geometry for tracking internal phase shifts.

C.3.4 Output Quality Monitoring

Input: Model predictions at regular training intervals, evaluated against gold-standard semantic role and POS annotations generated by ABSynth.

Output Example: Progressive analysis of model predictions across POS and semantic role categories reveals distinct learning trajectories for different linguistic abstractions, different than earlier illustrated differences. Figure 7 demonstrates category-specific performance evolution, with concrete grammatical categories achieving rapid convergence while abstract semantic roles exhibit more gradual acquisition patterns.



(a) POS tagging accuracy across training.

(b) Semantic role prediction accuracy across training.

Figure 7: (Left) POS accuracy across training steps. (Right) Semantic role prediction accuracy.

Interpretation:

- **Semantic Role Acquisition Spectrum:** Roles such as `ACTION`, `RELATION`, and `RESULTS` are acquired early and maintain stable, high accuracy throughout training, indicating reliable identification of core predicate-argument relations. In contrast, roles like `LOCATION` and `PATIENT` converge more slowly and exhibit very few lingering fluctuations, reflecting their higher contextual variability and integration complexity. Notably, once core roles are learned (10K steps), the model rarely misclassifies them, even during intermediate representational reorganisations, suggesting firm grammatical competence, even if confidence (as seen in probing) continues to be refined.

- **POS Category Learning Hierarchy:** Among POS tags, `PREPOSITION` reaches high accuracy early, likely due to its limited variability and syntactic rigidity. `VERB` predictions stabilise quickly, note that the plotted category captures base forms rather than diverse verb types but that is also available in the tool. `NOUN` accuracy improves more gradually, reflecting greater lexical variability, but still outpaces more abstract modifiers like `ADJECTIVE` and `ADVERB`, which remain the most challenging throughout training.

- **Stable Predictions vs. Ongoing Representation Refinement:** Once acquired, role predictions remain stable throughout training and exhibit few regressions. This contrasts with the confidence patterns observed in probing layers, where internal representation alignment continues to shift, suggesting external outputs stabilise before internal semantics fully consolidate, which highlights a gap between external performance and internal semantic stability.

C.4 Summary

This walkthrough illustrates TRACE's ability to surface rich interpretability signals across diverse analytical lenses. By combining probing, dimensionality tracking, loss landscape analysis, and output performance monitoring, TRACE enables multi-faceted insight into model development over time. In this controlled ABSynth setting, TRACE reveals coherent trends in linguistic acquisition, such as early emergence of core roles, delayed abstraction in modifiers, and coordinated dynamics across representation, curvature, and output quality. Notably, we observe a divergence between surface-level grammatical accuracy and internal representational confidence, highlighting that models may produce correct outputs even as their internal abstractions continue to evolve. Such findings underscore the need for tools like TRACE to go beyond external metrics and expose latent dynamics during learning. These capabilities generalise across architectures and datasets, making TRACE a practical framework for probing both model behaviour and training dynamics in applied or research contexts.