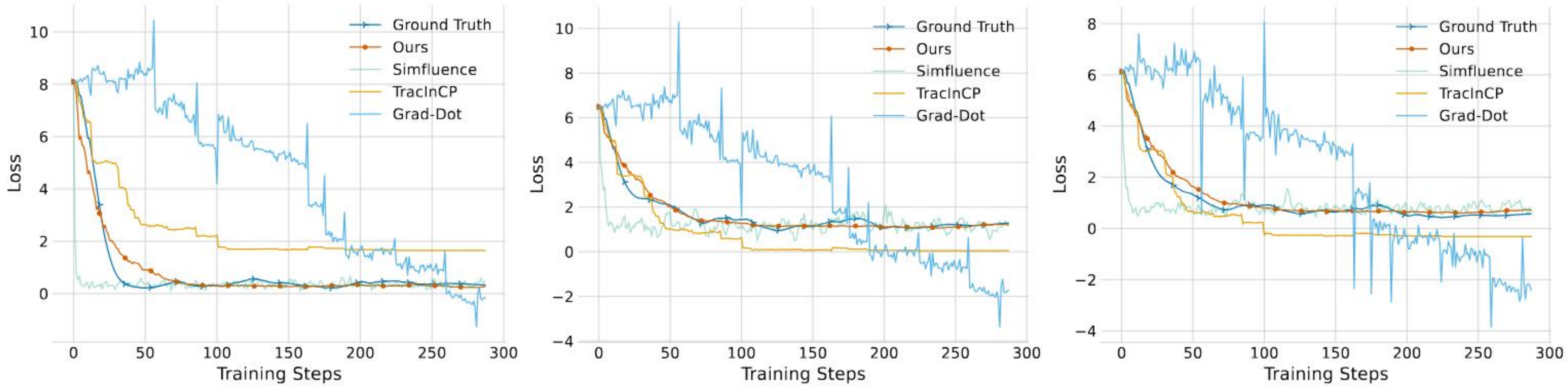

# On Training Data Influence of GPT Models

Qingyi Liu* Yekun Chai* Shuohuan Wang Yu Sun Qiwei Peng Hua Wu

Sun Yat-sen University Baidu Inc. University of Copenhagen

***Code:*** https://github.com/ernie-research/gptfluence
***Dataset:*** https://huggingface.co/datasets/baidu/GPTDynamics

# GPTfluence



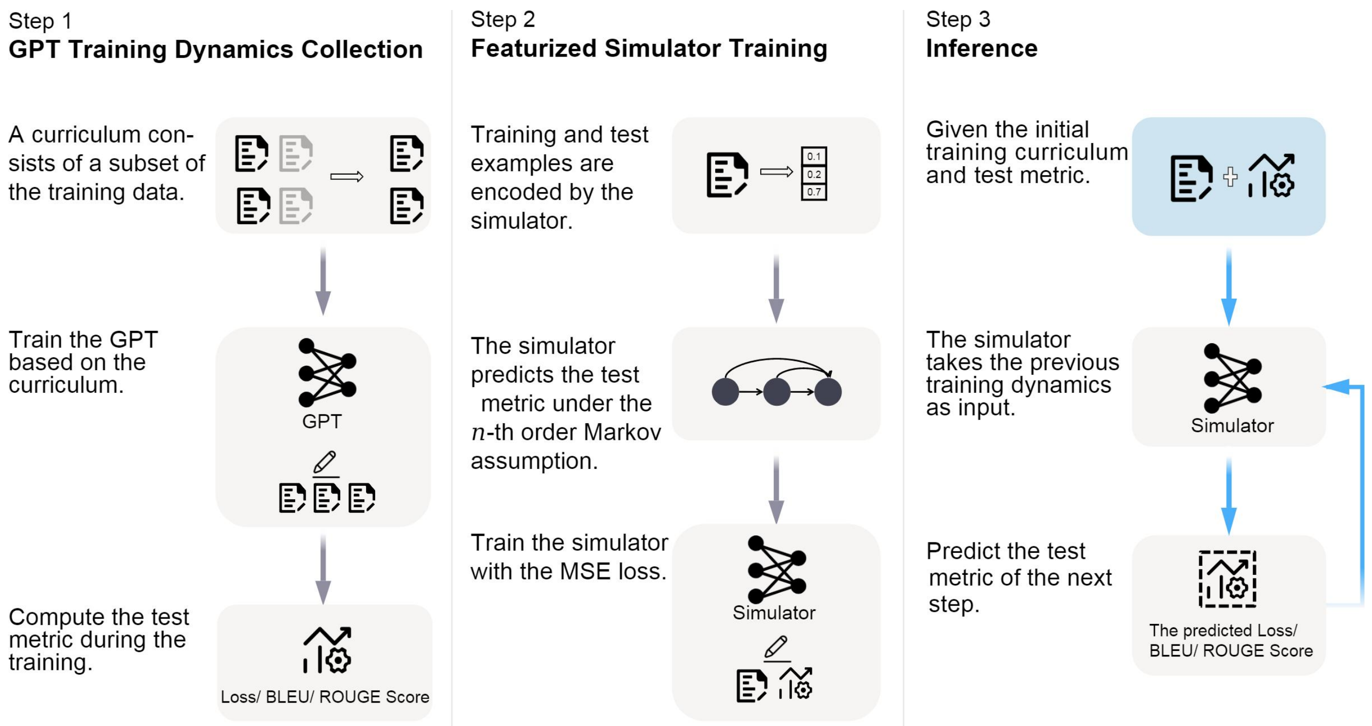

Despite GPT models have redefined performance standards across an extensive range of tasks, the training dynamics of GPT models remains a significantly underexplored area.

Current *training data attribution (TDA)* methods:

- Has *yet* to focus comprehensively on the influence of training data on **autoregressive language models**.
- Mainly focused on test loss, neglecting **other vital performance indicators**.
- Additionally, the challenge of **generalizability** persists as a significant barrier.

Therefore, we introduce, ***GPTfluence***, a novel approach that leverages a featurized simulation to assess the impact of training examples on the training dynamics of GPT models.

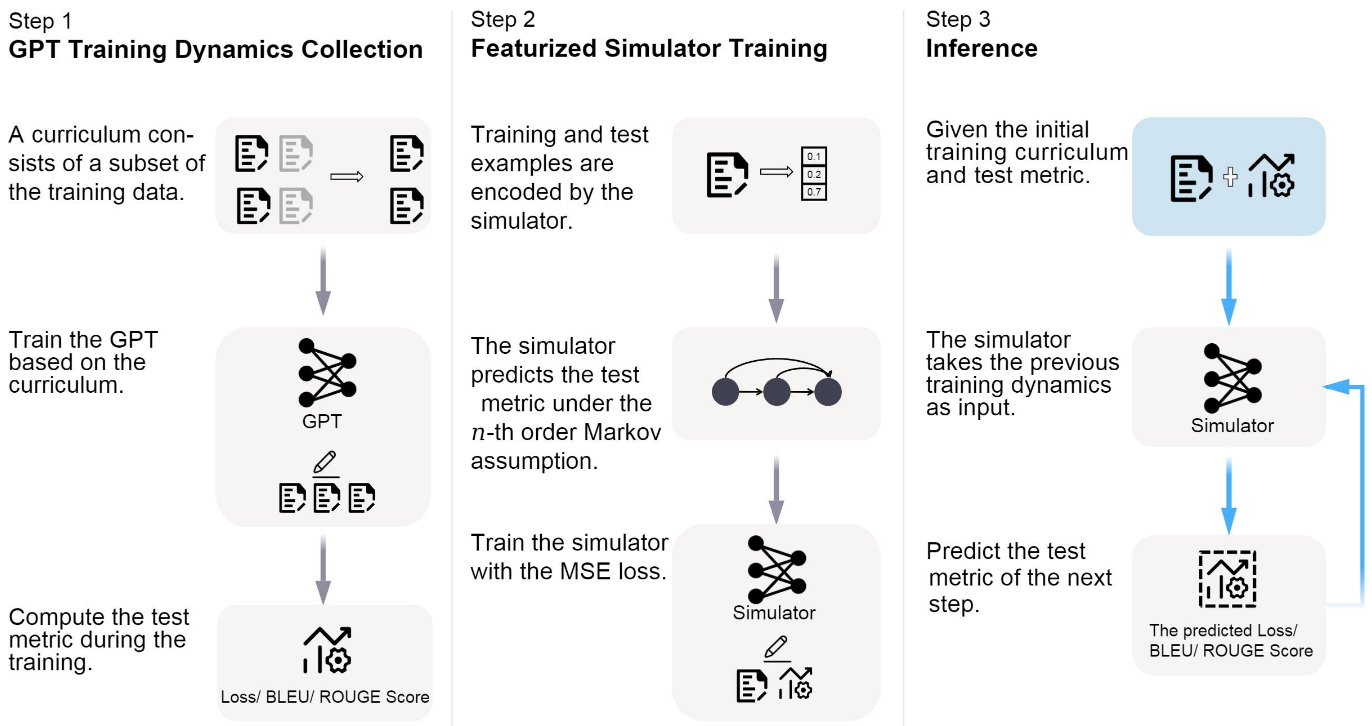
# GPTfluence




**Preliminaries**: A $T$ time steps training run is characterized by a sequence of training batches c, each contributing to the GPT's evolving parameters, $\theta_t$, through gradient descent.

***GPTfluence*** tracking the impact of training examples on the training dynamics of GPT models using a featurized simulator. The process of ***GPTfluence***, encompassing: *1.* the collection of training dynamics; *2.* the training of the simulator; *3*: the execution of the final simulation.

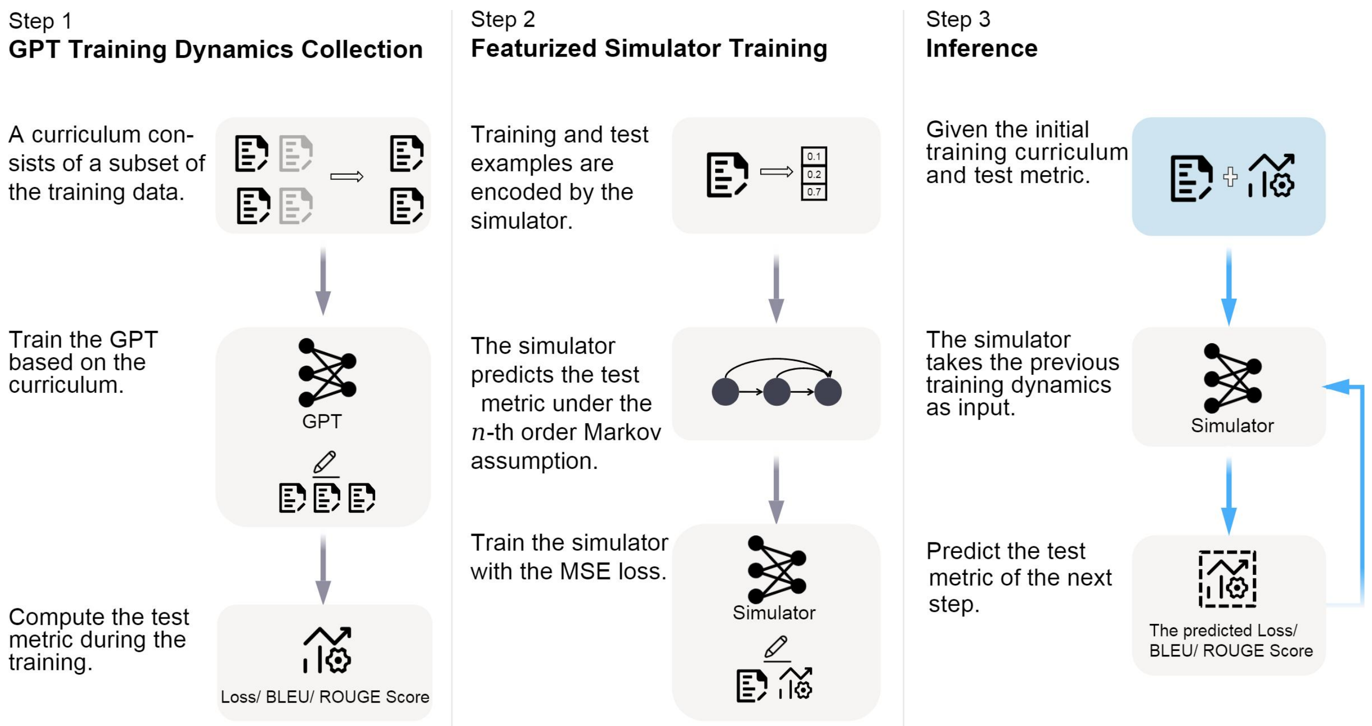
# GPTfluence




***Step1*: the collection of training dynamics.**

- From a broader dataset $D$, we sample $K$ subsets $D' \subset D$ for GPT model training, resulting in $K$ distinct training runs.
- Each runs includes both the training curriculum and the sequential target metric scores $\phi$ for each test point $z'$.

# GPTfluence

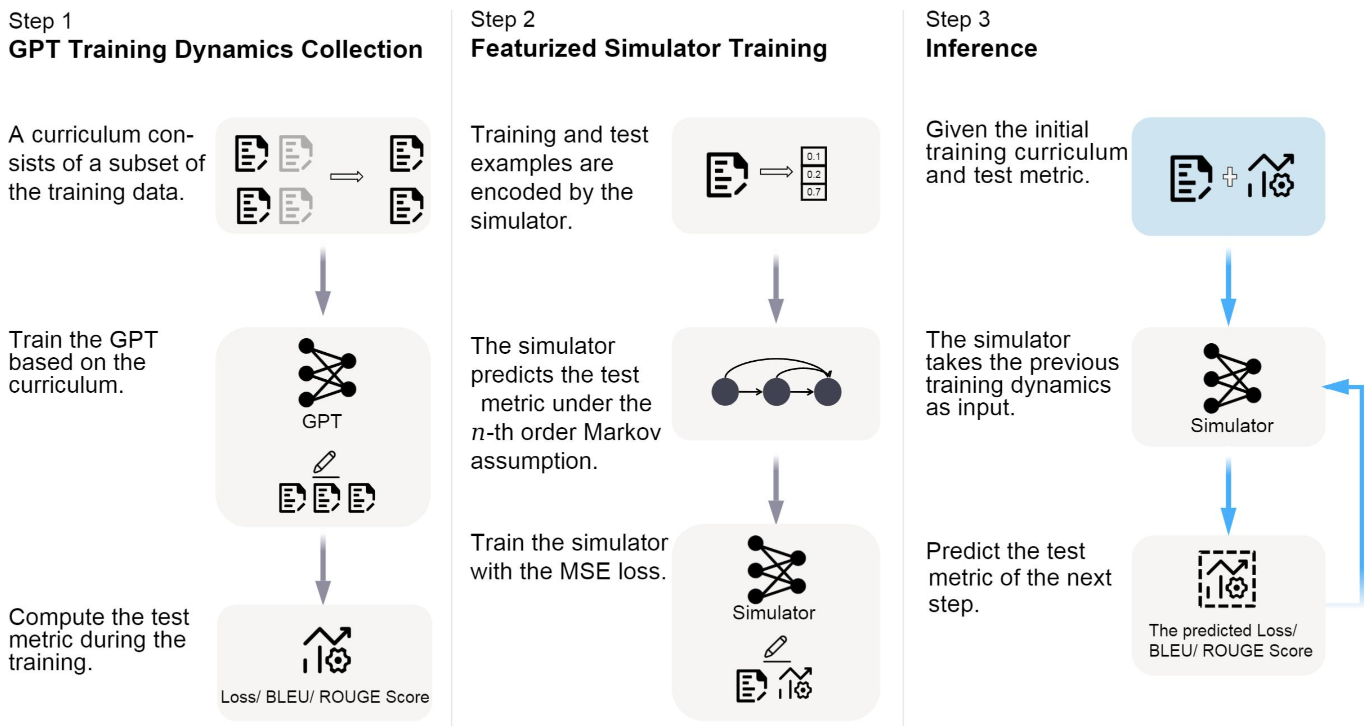



## ***Step2*: the training of the simulator**

- Our simulator integrates both multiplicative and additive components within the simulation, and the performance trajectory of a test sample $z'$ is thus delineated by a combination of these factors:

$$\phi_t(z') = \sum_{j=1}^{n} \alpha_j(c_t)\phi_{t-j}(z') + \beta(c_t)$$

# GPTfluence




## ***Step2*: the training of the simulator**

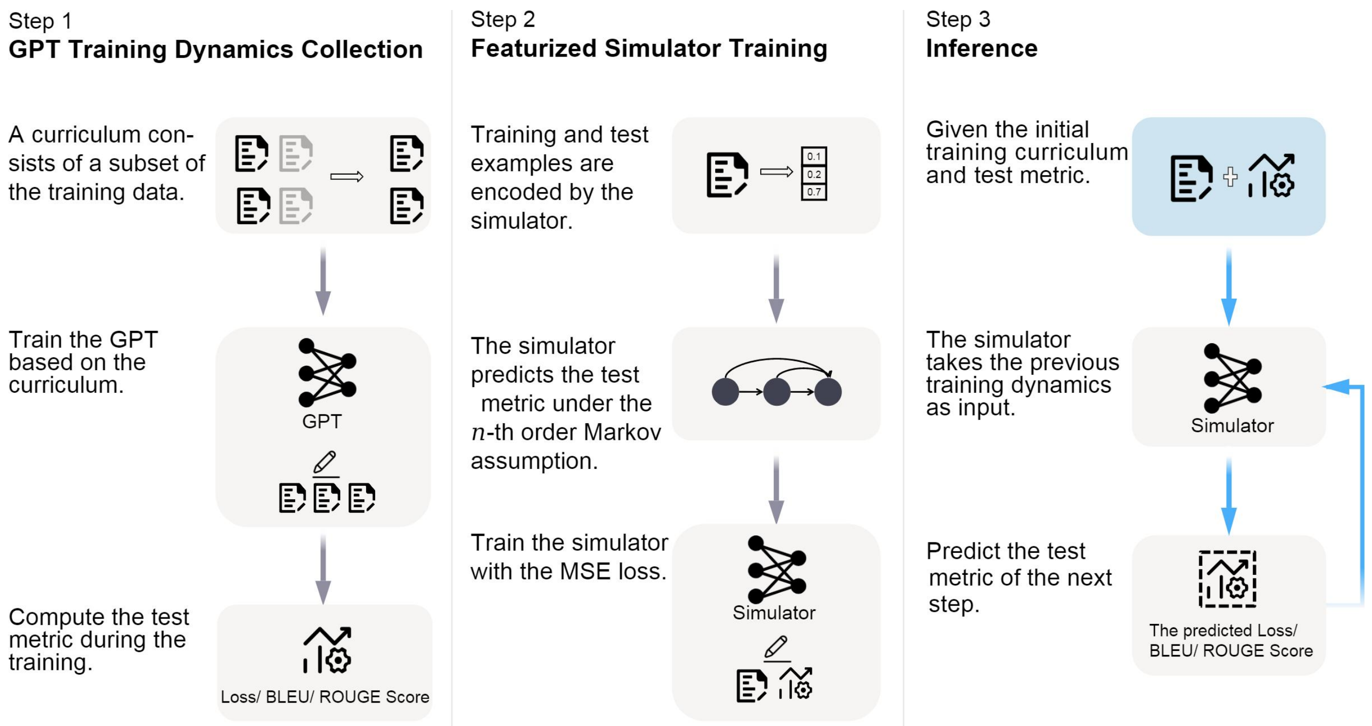
- Then, we introduce a **parameterized, featurized simulator** that employs a pre-trained encoder $\Psi(\cdot)$. This is adept at processing each training example $z_i$ and test example $z'$, generating predictive influence factors through the encoded representations $h^{z_i}$ and $h^{z'}$ .

$$h^{z_i} = \Psi(z_i), \quad h^{z'} = \Psi(z')$$

- To learn our featurized simulator $\Theta$, we optimize the following L2-regularized regression objective:

$$\Theta^\star = \operatorname*{argmin}_{\Theta} \sum_{t \in T} (y_t - \hat{\phi}_t(z'))^2 + \lambda(||\Theta||_2^2)$$

# GPTfluence




***Step3*: the execution of the final simulation.**

- The execution of this algorithm yields a ***GPTfluence*** simulator, which is adept at simulating the target performance trajectory and assessing the impact of training examples on a given test point.

# Experiments

| Method | #Param | RTE | | | SST-2 | | | BoolQ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | All-Steps MSE (↓) | All-Steps MAE (↓) | Final-Step Spearman's $\rho$ (↑) | All-Steps MSE (↓) | All-Steps MAE (↓) | Final-Step Spearman's $\rho$ (↑) | All-Steps MSE (↓) | All-Steps MAE (↓) | Final-Step Spearman's $\rho$ (↑) |
| TracIn-CP (10-steps) | 410M | 1.156(0.838) | 0.787(0.339) | 0.460 | 0.551(0.560) | 0.584(0.307) | -0.089 | 0.957(0.728) | 0.735(0.332) | -0.066 |
| TracIn-CP (all-steps) | | 0.757(0.591) | 0.629(0.299) | 0.460 | 0.446(0.555) | 0.525(0.321) | -0.089 | 0.782(0.690) | 0.680(0.339) | -0.066 |
| Grad-Dot | | 12.061(3.688) | 2.906(0.410) | 0.459 | 7.715(1.543) | 1.918(0.205) | -0.084 | 12.527(3.617) | 2.900(0.344) | -0.071 |
| Simfluence | | 1.477(0.274) | 0.634(0.111) | 0.426(0.340) | 1.133(0.287) | 0.455(0.082) | 0.696(0.156) | 1.189(0.362) | 0.485(0.082) | 0.793(0.201) |
| Ours | | **0.220(0.184)** | **0.334(0.140)** | **0.644(0.174)** | **0.111(0.045)** | **0.224(0.047)** | **0.834(0.129)** | **0.132(0.073)** | **0.251(0.075)** | **0.828(0.154)** |
| TracIn-CP (10-steps) | 1B | 1.225(0.744) | 0.979(0.344) | -0.203 | 4.412(1.301) | 1.697(0.170) | -0.058 | 0.999(1.034) | 0.793(0.400) | 0.649 |
| TracIn-CP (all-steps) | | 1.137(0.740) | 0.939(0.343) | -0.203 | 2.158(0.782) | 1.218(0.187) | -0.058 | 0.858(1.043) | 0.731(0.416) | 0.649 |
| Grad-Dot | | 21.928(7.871) | 4.332 (0.874) | -0.198 | 6.601(1.927) | 2.077(0.193) | -0.057 | 18.270(5.630) | 3.563(0.711) | 0.650 |
| Simfluence | | 0.889(0.551) | 0.523(0.197) | 0.360(0.207) | 0.582(0.253) | 0.410(0.084) | 0.712(0.148) | 0.876(0.470) | 0.469(0.198) | 0.862(0.050) |
| Ours | | **0.099(0.078)** | **0.227(0.097)** | **0.757(0.123)** | **0.096(0.075)** | **0.221(0.084)** | **0.807(0.175)** | **0.068(0.058)** | **0.187(0.070)** | **0.953(0.034)** |

| Method | #Param | WebNLG | | | WMT-16 DE/EN | | | Average | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | All-Steps MSE (↓) | All-Steps MAE (↓) | Final-Step Spearman's $\rho$ (↑) | All-Steps MSE (↓) | All-Steps MAE (↓) | Final-Step Spearman's $\rho$ (↑) | All-Steps MSE (↓) | All-Steps MAE (↓) | Final-Step Spearman's $\rho$ (↑) |
| TracIn-CP (10-steps) | 410M | 0.048(0.072) | 0.168(0.115) | 0.836 | 0.030(0.071) | 0.122(0.107) | 0.963 | 0.548 | 0.479 | 0.421 |
| TracIn-CP (all-steps) | | 0.050(0.073) | 0.173(0.113) | 0.836 | 0.030(0.071) | 0.123(0.107) | 0.963 | 0.413 | 0.426 | 0.421 |
| Grad-Dot | | 0.062(0.080) | 0.187(0.113) | 0.837 | 0.033(0.073) | 0.127(0.109) | 0.963 | 6.479 | 1.608 | 0.421 |
| Simfluence | | 0.036(0.029) | 0.130(0.049) | 0.986(0.002) | 0.016(0.013) | 0.101(0.034) | 0.997(0.001) | 0.770 | 0.361 | 0.779 |
| Ours | | **0.002(0.002)** | **0.033(0.017)** | **0.994(0.001)** | **0.002(0.004)** | **0.033(0.023)** | **0.998(0.000)** | **0.093** | **0.175** | **0.860** |
| TracIn-CP (10-steps) | 1B | 0.032(0.053) | 0.132(0.095) | 0.885 | 0.012(0.032) | 0.075(0.069) | 0.981 | 1.336 | 0.735 | 0.451 |
| TracIn-CP (all-steps) | | 0.033(0.053) | 0.135(0.094) | 0.885 | 0.012(0.032) | 0.076(0.069) | 0.981 | 0.840 | 0.620 | 0.451 |
| Grad-Dot | | 0.044(0.061) | 0.154(0.097) | 0.881 | 0.013(0.033) | 0.075(0.071) | 0.981 | 9.371 | 2.040 | 0.451 |
| Simfluence | | 0.167(0.127) | 0.323(0.112) | 0.823(0.030) | 0.171(0.269) | 0.309(0.168) | 0.925(0.007) | 0.537 | 0.407 | 0.737 |
| Ours | | **0.007(0.005)** | **0.068(0.022)** | **0.984(0.005)** | **0.004(0.004)** | **0.049(0.020)** | **0.997(0.001)** | **0.087** | **0.212** | **0.839** |

Table 1: Results of test loss estimation for *instruction tuning*. **Bold** are the optimal values.

| Dataset | Method | All-Steps MSE (↓) | All-Steps MAE (↓) | Final-Step Spearman's $\rho$ (↑) |
|---|---|---|---|---|
| RTE | Simfluence | **0.035(0.022)** | **0.151(0.054)** | 0.743(0.094) |
| | Ours | 0.036(0.029) | **0.151(0.060)** | **0.746(0.095)** |
| SST-2 | Simfluence | 0.037(0.017) | 0.128(0.030) | 0.938(0.074) |
| | Ours | **0.014(0.006)** | **0.081(0.018)** | **0.943(0.073)** |
| BoolQ | Simfluence | 0.032(0.019) | 0.140(0.038) | 0.992(0.002) |
| | Ours | **0.011(0.011)** | **0.082(0.049)** | **0.994(0.002)** |
| WebNLG | Simfluence | 0.016(0.012) | 0.094(0.036) | 0.984(0.002) |
| | Ours | **0.011(0.014)** | **0.078(0.043)** | **0.985(0.002)** |
| WMT-16 DE/EN | Simfluence | 0.010(0.008) | 0.067(0.029) | 0.998(0.003) |
| | Ours | **0.002(0.002)** | **0.031(0.018)** | **0.999(0.000)** |
| Average | Simfluence | 0.026 | 0.116 | 0.931 |
| | Ours | **0.015** | **0.084** | **0.933** |

Table 2: Results of test loss estimation for *fine-tuning*.

- **Test loss estimation for *instruction-tuning* and *fine-tuning*. *GPTfluence*** surpass Simfluence and other gradient-based TDA techniques across a set of five NLU and NLG tasks, as evidenced by the MSE and MAE metrics for the entire trajectory, alongside the Spearman correlation coefficients at the final time step across various test samples.

# Experiments

| Method | #Param | BLEU All-steps MSE (↓) | BLEU All-steps MAE (↓) | BLEU Final-step Spearman's ρ (↑) | Rouge-L All-steps MSE (↓) | Rouge-L All-steps MAE (↓) | Rouge-L Final-step Spearman's ρ (↑) |
|---|---|---|---|---|---|---|---|
| **WebNLG** | | | | | | | |
| Simfluence | 410M | 23.47(63.52) | 2.34(3.26) | 0.81(0.02) | 0.007(0.008) | 0.055(0.038) | 0.708(0.067) |
| Ours | 410M | **9.11(18.41)** | **1.73(1.82)** | **0.90(0.03)** | **0.005(0.006)** | **0.045(0.034)** | **0.796(0.047)** |
| Simfluence | 1B | 20.58(60.80) | 2.01(3.03) | **0.87(0.03)** | 0.006(0.006) | 0.052(0.031) | 0.878(0.035) |
| Ours | 1B | **9.72(23.70)** | **1.63(2.02)** | 0.86(0.03) | **0.004(0.005)** | **0.043(0.029)** | **0.903(0.020)** |
| **WMT-16 DE/EN** | | | | | | | |
| Method | #Param | BLEU All-steps MSE (↓) | BLEU All-steps MAE (↓) | BLEU Final-Step Spearman's ρ (↑) | Rouge-L All-steps MSE (↓) | Rouge-L All-steps MAE (↓) | Rouge-L Final-Step Spearman's ρ (↑) |
| Simfluence | 410M | 32.15(116.17) | 2.25(4.08) | 0.83(0.03) | 0.007(0.017) | 0.039(0.055) | 0.931(0.014) |
| Ours | 410M | **7.71(28.05)** | **1.14(1.92)** | **0.92(0.02)** | **0.004(0.009)** | **0.030(0.041)** | **0.964(0.012)** |
| Simfluence | 1B | 162.94(466.30) | 5.71(9.03) | 0.76(0.03) | 0.025(0.038) | 0.094(0.098) | 0.833(0.031) |
| Ours | 1B | **46.33(122.50)** | **3.34(4.68)** | **0.93(0.01)** | **0.013(0.020)** | **0.066(0.069)** | **0.910(0.011)** |
| **Average** | | | | | | | |
| Method | #Param | BLEU All-steps MSE (↓) | BLEU All-steps MAE (↓) | BLEU Final-step Spearman's ρ (↑) | Rouge-L All-steps MSE (↓) | Rouge-L All-steps MAE (↓) | Rouge-L Final-step Spearman's ρ (↑) |
| Simfluence | 410M | 27.81 | 2.29 | 0.82 | 0.007 | 0.047 | 0.820 |
| Ours | 410M | **8.41** | **1.43** | **0.91** | **0.004** | **0.037** | **0.880** |
| Simfluence | 1B | 91.76 | 3.86 | 0.81 | 0.015 | 0.073 | 0.855 |
| Ours | 1B | **28.02** | **2.51** | **0.90** | **0.008** | **0.055** | **0.907** |

Table 3: Results of test metric estimation on NLG datasets for *instruction-tuning*.

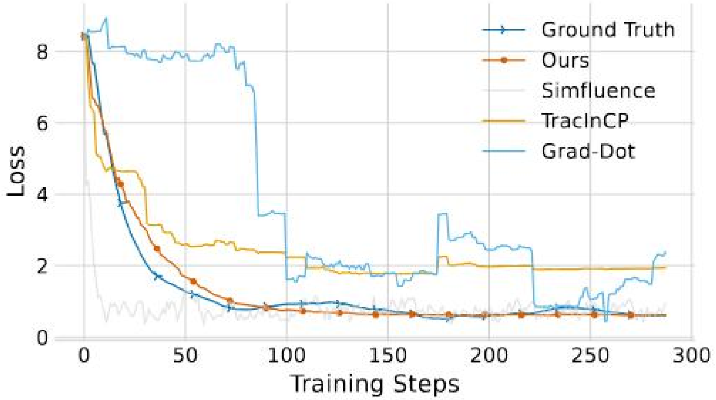
| Dataset | Metric | Method | All-steps MSE (↓) | All-steps MAE (↓) | Final-Step Spearman's ρ (↑) |
|---|---|---|---|---|---|
| WebNLG | BLEU | Simfluence | **43.33 (77.34)** | **4.23 (3.52)** | 0.78 (0.02) |
| | | Ours | 43.98 (81.40) | 4.28 (3.57) | **0.80 (0.01)** |
| | Rouge-L | Simfluence | 0.008 (0.007) | 0.066 (0.031) | 0.706 (0.038) |
| | | Ours | **0.007 (0.006)** | **0.060 (0.029)** | **0.765 (0.040)** |
| WMT-16 DE/EN | BLEU | Simfluence | 32.11 (89.13) | **2.76 (3.75)** | 0.82 (0.02) |
| | | Ours | **30.26 (77.23)** | 2.91 (3.69) | **0.81 (0.02)** |
| | Rouge-L | Simfluence | 0.018 (0.025) | 0.091 (0.075) | 0.796 (0.032) |
| | | Ours | **0.012 (0.016)** | **0.075 (0.057)** | **0.843 (0.010)** |
| **Average** | BLEU | Simfluence | 37.72 | **3.49** | 0.80 |
| | | Ours | **37.12** | 3.59 | **0.81** |
| | Rouge-L | Simfluence | 0.013 | 0.079 | 0.751 |
| | | Ours | **0.009** | **0.068** | **0.805** |

Table 4: Results of test metric estimation on NLG datasets for *fine-tuning*.

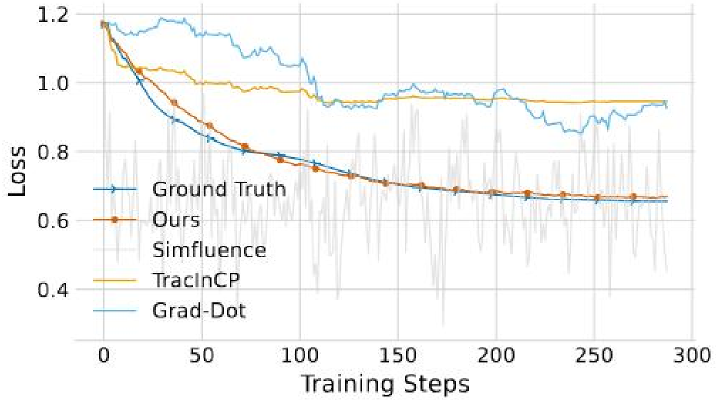
- **Generalizing to test metric estimation for *instruction-tuning* and *fine-tuning*. *GPTfluence*** expands the test loss evaluation limitation of gradient-based TDA methods to vital measures and has a superior performance over Simfluence.

# Experiments

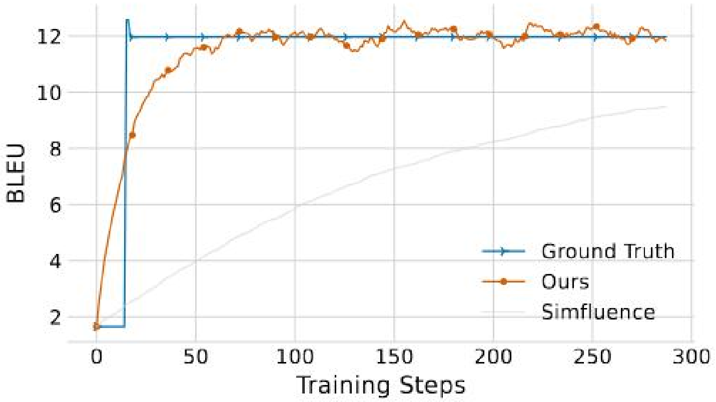
- Examples of Test Loss & Metric Estimation of *GPTfluence*



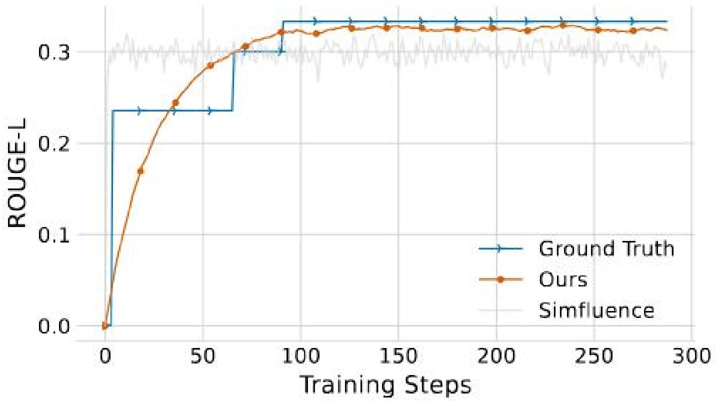

(a) Loss Simulation on NLU Task (*RTE*)




(b) Loss Simulation on NLG Task (*WMT16 DE/EN*)




(c) BLEU Simulation on *WebNLG*




(d) ROUGE-L Simulation on *WMT16 DE/EN*

Figure 2: Illustration of *loss* and *metric* simulation on natural language understanding (**NLU**) and natural language generation (**NLG**) tasks with different TDA methods for *instruction tuning*. See the Appendix for more examples.

# Experiments

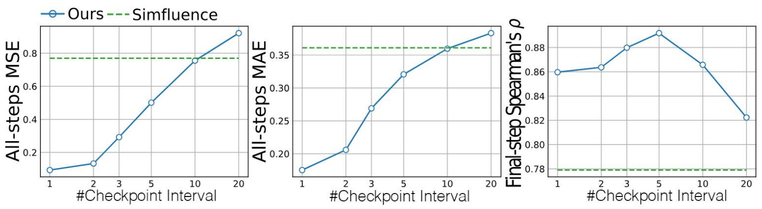



Figure 3: Variation curves of the average performance of GPTfluence for loss simulation in five datasets when different checkpoint intervals are selected.

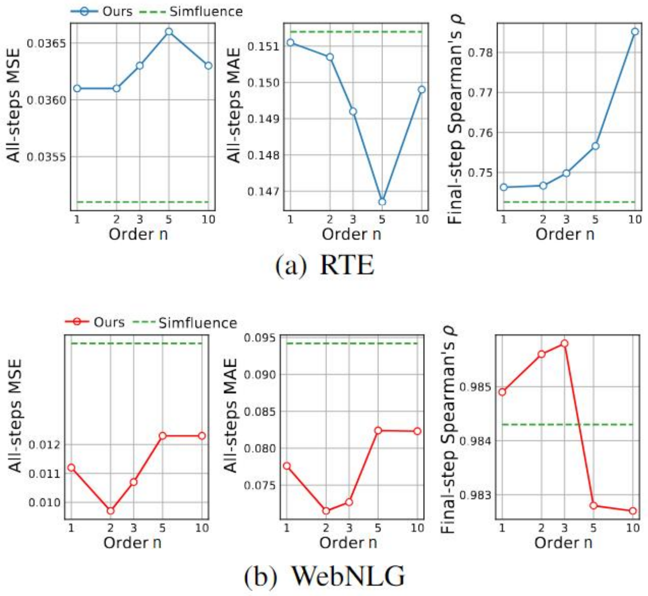



Figure 4: Analysis on the impact of *n-th order Markov process* on language understanding (RTE) and generation (WebNLG) tasks, varying $n$ from 1 to 10.

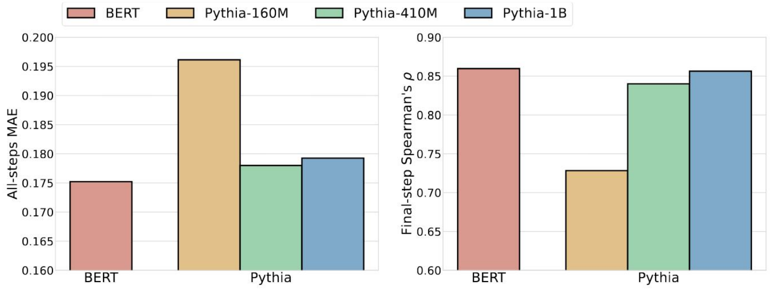



Figure 5: Impact of feature representation of different pre-trained encoders on loss simulation.

- **Ablation of Practical influence via checkpoints.** The performance deteriorates as the number of checkpoint intervals increases but still is comparable when even intervals= 10, saving almost 90% data collection cost.
- **Ablation of Markov Order Dependency.** The simulation error initially increases and decreases, with more preceding training information, for both datasets.
- **Ablation of Different Feature Representations.** BERT's feature representations generally produce better simulation results than the Pythia encoder.

# Experiments

## Analysis

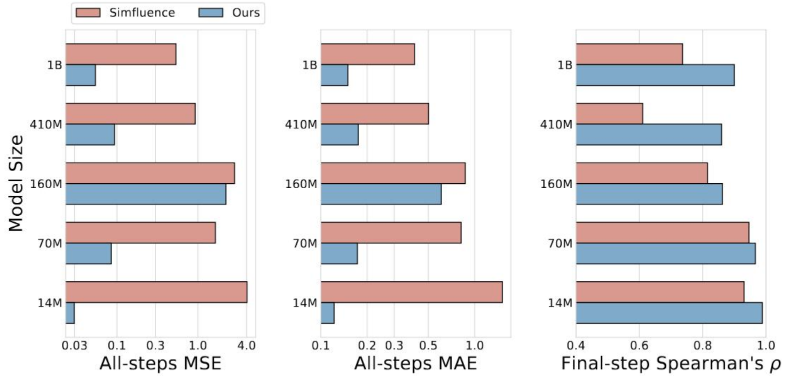
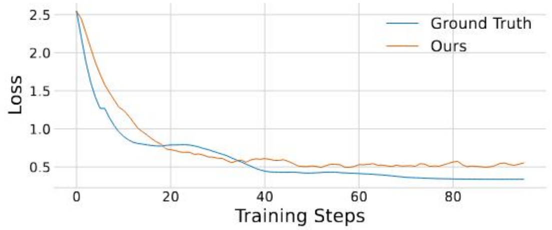
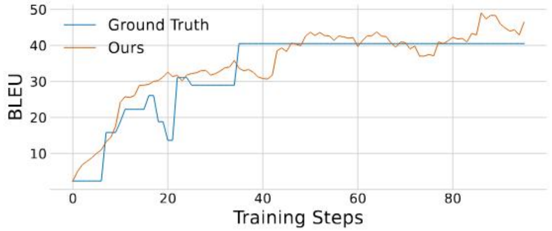



Figure 6: Comparison of the loss simulation performance between GPTfluence and Simfluence when instruction tuning Pythia models of various sizes.




(a) RTE




(b) WebNLG

Figure 7: Illustration of simulation results on unseen *training* data. The *top* shows the loss simulation for the RTE dataset, while the *bottom* shows the BLEU metric simulation for the WebNLG dataset. Additional qualitative examples for different settings and metrics are provided in the Appendix § C.2.

| **Method** | **Latency (sec/sample)** | **FLOPs** |
|---|---|---|
| TracIn-CP | 153.0 | $1.1 \times 10^{13}$ |
| Simfluence | 0.1 | $1.6 \times 10^{1}$ |
| Ours | 0.2 | $5.3 \times 10^{6}$ |

Table 5: Inference latency and FLOPs of GPTfluence, Simfluence, and TracIn-CP.

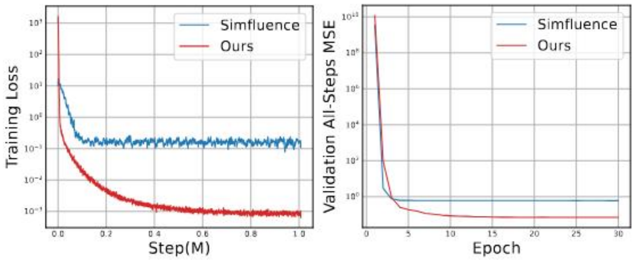



Figure 8: Comparison of our method and Simfluence with respect to **training loss** (Left) and **validation all-steps MSE** (Right).

- **Robustness across varying model sizes.** ***GPTfluence*** consistently surpassed Simfluence with increasing LLM size.
- **Unseen Data Generalization.** ***GPTfluence*** can generalize to unseen data, which includes simulating loss and performance metrics.
- **Computational Complexity.** ***GPTfluence*** exhibits a better convergence efficiency with acceptable inference latency.

# Experiments

## Analysis

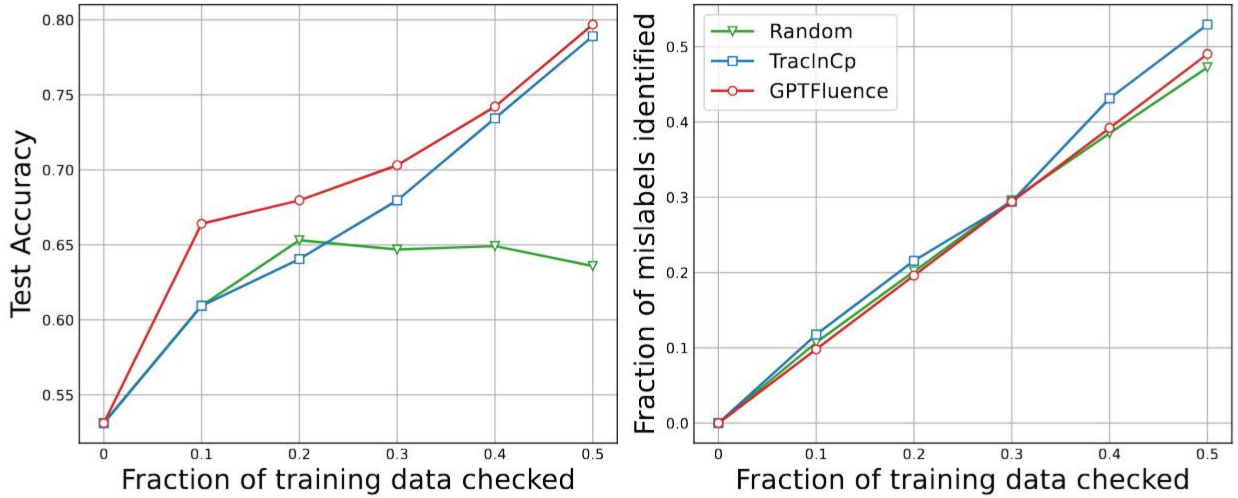
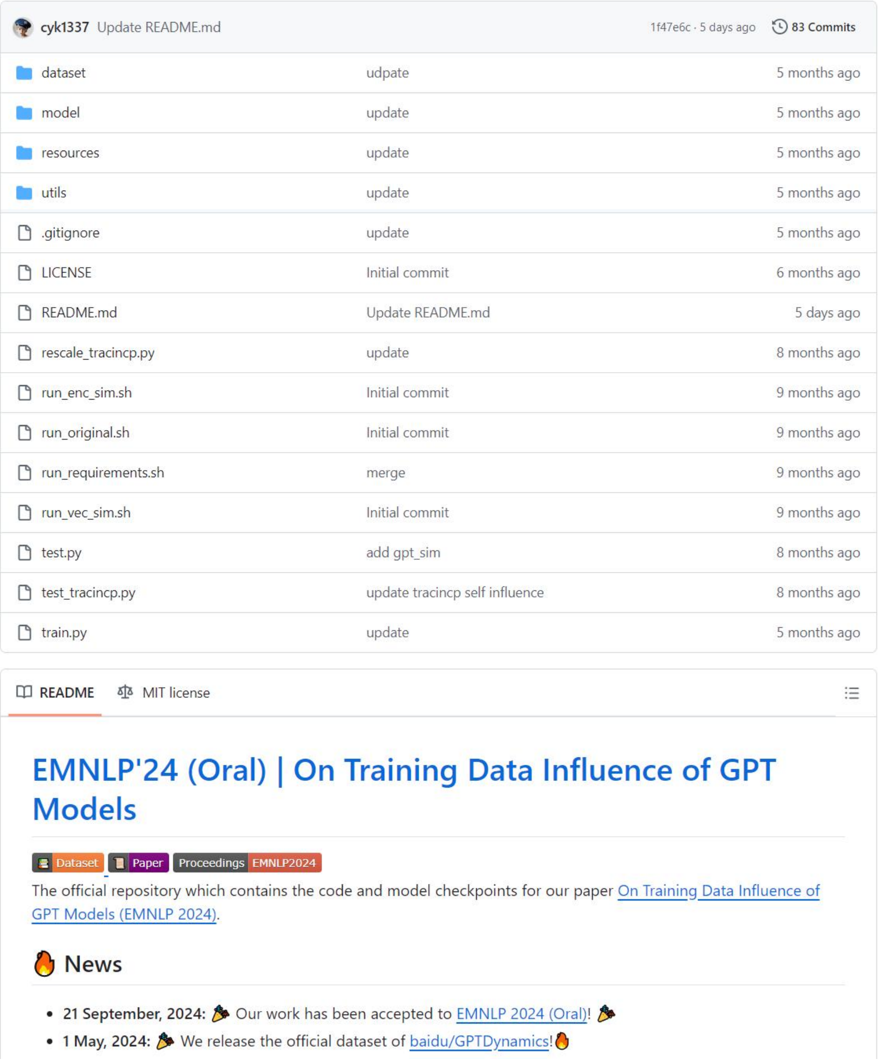



Figure 9: SST-2 Mislabelled Data Identification with GPTfluence, TracIn-CP and Random Selection.

- **Use Case: Mislabelled Data Identification.** *GPTfluence* shows a higher detection efficiency, with the most significant performance improvement when the checked fraction is low.

# Code & Datasets




*https://github.com/ernie-research/gptfluence?tab=readme-ov-file*

*https://huggingface.co/datasets/baidu/GPTDynamics*

# Thank you!