Towards Tracing Factual Knowledge in Language Models Back to the Training Data

Ekin Akyürek[†] **Tolga Bolukbasi** **Frederick Liu** **Binbin Xiong**

Ian Tenney **Jacob Andreas**[†] **Kelvin Guu**

Google Research [†]MIT CSAIL

Abstract

Language models (LMs) have been shown to memorize a great deal of factual knowledge contained in their training data. But when an LM generates an assertion, it is often difficult to determine *where* it learned this information and whether it is true. In this paper, we propose the problem of *fact tracing*: identifying which training examples taught an LM to generate a particular factual assertion. Prior work on *training data attribution* (TDA) may offer effective tools for identifying such examples, known as "proponents". We present the first quantitative benchmark to evaluate this. We compare two popular families of TDA methods — *gradient*-based and *embedding*-based — and find that much headroom remains. For example, both methods have lower proponent-retrieval precision than an information retrieval baseline (BM25) that does not have access to the LM at all. We identify key challenges that may be necessary for further improvement such as overcoming the problem of gradient saturation, and also show how several nuanced implementation details of existing neural TDA methods can significantly improve overall fact tracing performance. [1]

1 Introduction

Research has shown that language models (LMs) acquire significant amounts of world knowledge from the massive text corpora on which they are trained (Petroni et al., 2019; Raffel et al., 2020). This development has enabled exciting advances in knowledge-intensive NLP tasks such as open-domain question answering (Roberts et al., 2020) and knowledge base population (Petroni et al., 2019). LMs have also been shown to generate factually incorrect statements (Lee et al., 2018; Tian et al., 2019), which is problematic for many applications where trustworthiness is important. Hence, there is an urgent need to understand exactly how LMs acquire and store knowledge so that we may improve their accuracy and coverage.

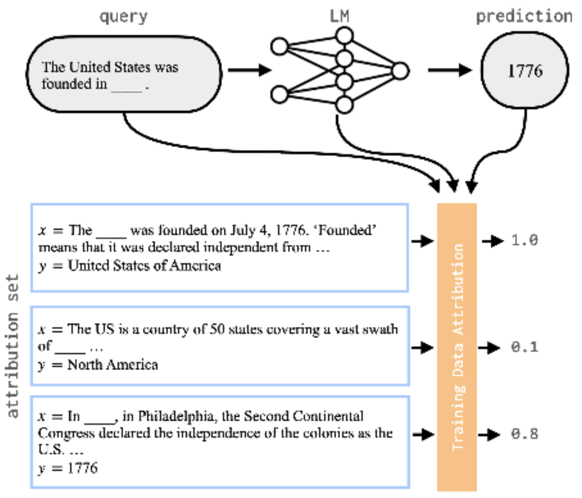

query
The United States was founded in ____ .
LM
prediction
1776
attribution set
x = The ____ was founded on July 4, 1776. 'Founded' means that it was declared independent from …
y = United States of America
x = The US is a country of 50 states covering a vast swath of ____ …
y = North America
x = In ____, in Philadelphia, the Second Continental Congress declared the independence of the colonies as the U.S. …
y = 1776
Training Data Attribution
1.0
0.1
0.8

Figure 1: FTRACE benchmark for tracing a language model's predictions back to training examples ("proponents"): We provide two fact attribution datasets: one with real facts (FTRACE-TREx) and one with synthetic facts (FTRACE-SYNTH). We evaluate commonly studied attribution methods, including gradient-based and embedding-based approaches for their ability to identify true proponents.

Training Data Attribution Ultimately, a language model's "knowledge" must derive from its training data. But there has been little research

[1]Code for the experiments is released at https://github.com/ekinakyurek/influence, and the datasets can be downloaded from https://huggingface.co/datasets/ekinakyurek/ftrace. Correspondences to akyurek@mit.edu

on attributing an LM's factual assertions back to specific training examples — a task we call *fact tracing*. Training data attribution methods (TDA) are the main literature concerned with linking predictions back to specific training examples (known as "proponents"). Influence functions (Hampel, 1974; Koh and Liang, 2017) and TracIn (Pruthi et al., 2020) are among the first methods to do this for neural networks, by estimating the marginal effect of a training example on the loss of a test-time example. However, most work on TDA has focused on classification and regression tasks that do not necessarily involve fine-grained factual information (Han et al., 2020; Hara et al., 2019).

Several obstacles have limited research on fact tracing for large, pre-trained LMs. First, since pre-training corpora are very large, it has not been clear how to obtain ground truth labels regarding which pre-training example was truly responsible for an LM's prediction. Second, TDA methods have traditionally been computationally prohibitive. In this paper, we present one of the first computationally tractable studies of fact tracing for LMs. To do so, we construct:

(1) Two specially designed evaluation datasets, FTRACE-TREx and FTRACE-Synth, which contain unambiguous ground-truth information about the origin of specific facts.

(2) A tractable procedure for evaluating fact-tracing methods on large-scale LMs.

Obtaining Ground Truth Proponents To establish (1) ground truth data for fact tracing, we propose a new recipe, which we call "novel fact injection". First, suppose that we can identify a set of "facts" that the pre-trained LM does *not* know — we call these "novel facts". We can convert each novel fact into an LM training example, and then fine-tune the LM on these extra examples until it memorizes the novel facts (i.e. "injecting" them into the LM). With a few caveats, we now know that the LM must have learned these facts from our newly injected examples. We also know which examples are responsible for teaching each fact, since we constructed each example from a particular fact. Hence, we now have ground-truth "proponents" for every novel fact, and can evaluate any TDA method on its ability to identify these proponents – i.e. to retrieve the true proponents out of a large set of training examples.

We implement this recipe using the TREx dataset (Elsahar et al., 2018) as our source of novel facts. TREx is a large text corpus where each sentence has been comprehensively annotated with the facts that it expresses, in the form of relational knowledge tuples. To identify novel facts present in TREx, we filter for knowledge tuples that the pre-trained LM did not already know, as tested using masked LM prompting. The sentences in TREx expressing these tuples are then "injected" via fine-tuning and labeled as proponents. We call this setup **FTRACE-TREx**.

There are two caveats for the above setup. First, we must be careful about how we define what an LM "knows". For example, if an LM generates a particular assertion with 10% probability, does this count as "knowing" or not? Second, some facts can be indirectly inferred from other facts. For example, suppose we want to know how an LM learned that Barack Obama was born in Hawaii. It could learn this from a literal mention of the fact: "Obama was born in Hawaii", or indirectly infer it from "Obama was born in Honolulu". TREx dataset mostly identifies literal proponents (the former), but not indirect proponents (the latter).

To address these two issues, we introduce an additional, more controlled setup, **FTRACE-Synth**, featuring synthetically generated novel facts that could not have possibly been known by the pre-trained LM, and which also have no correlation with any existing facts – making indirect inferences impossible.

Mitigating Computational Cost To mitigate (2) the high computational cost of most TDA methods, we propose a simple reranking setup that is commonly used in information retrieval (IR) experiments. Rather than running a TDA method over all training examples, we run it only over a small subset of "candidate" examples that is guaranteed to include the ground truth proponents as well as some "distractor" examples that are not true proponents. In this way, a TDA method always has the opportunity to identify the true proponents while still facing challenging distractors, which enables us to differentiate the performance of multiple methods.

Key Results Having developed data and quantitative evaluation methods for fact tracing, we use them to evaluate two popular families of TDA methods: gradient-based methods (such as Pruthi et al., 2020), and embedding-based methods (Rajani et al., 2020). As a reference point, we also compare these TDA methods against a simple baseline: BM25 (Robertson et al., 1995; Lv and Zhai, 2011), a standard IR technique that simply selects proponents by retrieving training examples that have high lexical overlap with the query.

To measure the full potential of existing TDA methods, we optimize over key design choices including checkpoint selection, layer selection, embedding normalization and more. Furthermore, we significantly improve TracIn by introducing a simple but novel variant that accounts for training optimizer momentum (Shazeer and Stern, 2018). Even with these improvements, we find that much headroom remains, as all TDA methods still underperform BM25 on the FTRACE-TREx dataset. This does not imply that BM25 is optimal for the task, but rather demonstrates clear ways in which TDA methods could improve. On our more controlled FTRACE-SYNTH, we observe that TDA methods are significantly more competitive, especially when we introduce lexical variation in the way facts are expressed. We conclude that significant headroom remains for TDA methods to successfully address the new task of fact tracing.

2 Retrieval Methods

We begin with a formal description of the different TDA methods we study in this paper: *gradient-based* methods (Koh and Liang, 2017; Pruthi et al., 2020) and *embedding-based* methods (Rajani et al., 2020). To contextualize the performance of these two families of approaches, we also describe a widely used information retrieval baseline, BM25, which uses surface lexical similarity and thus tells us how effectively we can perform fact tracing without even having to access a model.

2.1 Gradient-based Attribution

Influence functions (Hampel, 1974; Koh and Liang, 2017) provide one of the first and best-known attribution methods. Given a training example $z = (x, y)$ and a test example $z_{\text{query}} = (x_{\text{query}}, y_{\text{query}})$, influence functions seek to estimate the change in the loss on z_{query} given an ϵ increase in the weight of a particular training example z at training time. Computing the influence of a training example z involves first estimating the change in the optimal parameters $\hat{\theta}$, given that the example z is up-weighted by ϵ in the training objective, then calculating how much the loss on z_{query} changes w.r.t. the parameter change. The resulting influence score for convex loss functions is shown to be:

$$\mathcal{I}(z, z_{\text{query}}) = \\ -\nabla_\theta L\left(z_{\text{query}}, \hat{\theta}\right)^\top H_{\hat{\theta}}^{-1} \nabla_\theta L\left(z, \hat{\theta}\right) \quad (1)$$

where $\nabla_\theta L(z, \theta)$ denotes the gradient of the loss function on example z evaluated at model parameters θ, and $H_{\hat{\theta}}$ denotes the Hessian of the training objective evaluated at the final converged model parameters, $\hat{\theta}$ (see Koh and Liang (2017) for the derivation). In this form, influence functions can be roughly viewed as the weighted dot product of the gradients for z_{query} and z, where the weight is the inverse Hessian of the training objective at $\hat{\theta}$. Due to the complexity of inverse Hessian calculation, the naive computational complexity is $\mathcal{O}(np^2 + p^3)$ (n is dataset size, p is parameter size). Even after the sampling approximations proposed in Koh and Liang (2017), the cost is still too high to directly apply influence functions for fact tracing.[2]

Therefore, we turn to a more recent TDA method that has demonstrated both better tractability and strong empirical results: TracIn (Pruthi et al., 2020), which seeks to estimate influence by asking a credit-assignment question rather than a counterfactual perturbation question. During training, when we take a gradient step on training example z (input, output) at time t, we ask how much the loss changes on test example z_{query}. TracIn employs a first-order Taylor approximation to answer this question, yielding the following estimate, which is simply the dot product of gradients at a particular step t:

$$\mathcal{I}_{\text{t}}(z, z_{\text{query}}) = \nabla_\theta L\left(z_{\text{query}}, \theta_{\text{t}}\right)^\top \nabla_\theta L\left(z, \theta_{\text{t}}\right) \quad (2)$$

If we have taken K gradient steps on the training example, this yields the total influence:

$$\mathcal{I}(z, z_{\text{query}}) = \\ \sum_{k=1}^{K} \nabla_\theta L\left(z_{\text{query}}, \theta_{\text{t}(k)}\right)^\top \nabla_\theta L\left(z, \theta_{\text{t}(k)}\right) \quad (3)$$

where $\text{t}(k)$ denotes the training step at which we took the k^{th} gradient step on training example z.

[2] Schioppa et al. (2022) propose more tractable approximations for Hessian based influence, but the memory requirement of the proposed method is still infeasible without projecting gradients into lower dimensions. Please refer to (Basu et al., 2021) for additional shortcomings.

The sum over time steps is generally approximated by using some fixed set of training checkpoints, which need not coincide with the actual steps where z was visited. A known issue is that gradient similarity may be dominated by outlier training examples with large gradients. A simple fix proposed in previous work (Barshan et al., 2020; Han and Tsvetkov, 2021) is to unit-normalize the gradients, effectively replacing the dot product in Equation (2) with cosine similarity:

$$\mathcal{I}(z, z_{\text{query}}) = \sum_{k=1}^{K} \frac{\nabla_\theta L\left(z_{\text{query}}, \theta_{\mathrm{t}(k)}\right)^\top \nabla_\theta L\left(z, \theta_{\mathrm{t}(k)}\right)}{\|\nabla_\theta L\left(z_{\text{query}}, \theta_{\mathrm{t}(k)}\right)\| \|\nabla_\theta L\left(z, \theta_{\mathrm{t}(k)}\right)\|} \tag{4}$$

We hereafter refer to $\mathcal{I}$ in Equation (4) as TRACIN.

2.2 Embedding-based Attribution

Hidden representations of neural networks are known to embed high-level features that are often useful for similarity search. While not as theoretically justified, prior work (Rajani et al., 2019) has found that such representations can outperform gradient-based methods in model interpretability tasks. Following prior work, we extract the intermediate layer outputs of a Transformer language model, and average over decoding time-steps to obtain a single vector representation for any example. In our experiments, we consider representations at different layers of the Transformers, as well as their concatenations. Similar to the case of gradient-based methods, the association between a training example and a model prediction is defined by a cosine product:

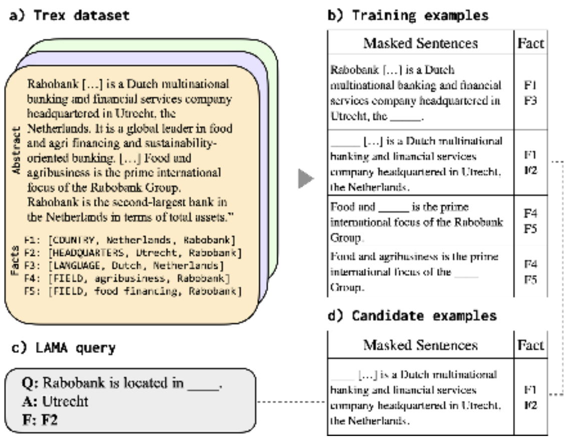
$$\mathcal{I}(z, z_{\text{query}}) = \frac{LM_{\text{inter.}}(z)^\top LM_{\text{inter.}}(z_{\text{query}})}{\|LM_{\text{inter.}}(z)^\top\| \|LM_{\text{inter.}}(z_{\text{query}})\|} \tag{5}$$

where $LM_{\text{inter.}}$ denotes a layer's output in LM. We refer to $\mathcal{I}$ in Equation (5) as EMBED.

2.3 Baseline: BM25

In the previous sections, we used attribution methods to define a model-specific similarity function between examples. But it is also possible to identify facts in a model-agnostic way: In the classic IR literature, word-overlap based methods have been shown to be both simple and effective.

Among these approaches, BM25 (Robertson et al., 1995; Lv and Zhai, 2011), the best performing variant, has been consistently used as a baseline for information retrieval benchmarks (Thakur et al., 2021) . When using BM25, we consider an example as a bag of words consist of the input and the output words. The score is proportional to token overlap between the query and the candidate, inversely weighted with the frequency of such tokens, and the importance of weights regulated by hyperparameters (Appendix A).


a) Trex dataset
Abstract
Rabobank [...] is a Dutch multinational banking and financial services company headquartered in Utrecht, the Netherlands. It is a global leader in food and agri financing and sustainability-oriented banking. [...] Food and agribusiness is the prime international focus of the Rabobank Group. Rabobank is the second-largest bank in the Netherlands in terms of total assets."
Facts
F1: [COUNTRY, Netherlands, Rabobank]
F2: [HEADQUARTERS, Utrecht, Rabobank]
F3: [LANGUAGE, Dutch, Netherlands]
F4: [FIELD, agribusiness, Rabobank]
F5: [FIELD, food financing, Rabobank]
b) Training examples
c) LAMA query
Q: Rabobank is located in ____.
A: Utrecht
F: F2
d) Candidate examples

Masked Sentences	Fact
Rabobank [...] is a Dutch multinational banking and financial services company headquartered in Utrecht, the ____.	F1 F3
____ [...] is a Dutch multinational banking and financial services company headquartered in Utrecht, the Netherlands.	F1 F2
Food and ____ is the prime international focus of the Rabobank Group.	F4 F5
Food and agribusiness is the prime international focus of the ____ Group.	F4 F5

Masked Sentences	Fact
____ [...] is a Dutch multinational banking and financial services company headquartered in Utrecht, the Netherlands.	F1 F2

Figure 2: Dataset Creation: From the original TREx (Elsahar et al., 2018) data, we construct masked sentences and annotate their facts by using provided fact annotations. We identify proponents by matching the TREx sentences expressing the same fact. (The outputs of the masked examples are omitted in the figure.)

3 Fact Tracing Datasets

We propose two datasets to measure fact tracing approaches: FTRACE-TREX, a natural language dataset with real facts derived from the TREx dataset, and FTRACE-SYNTH, a synthetic dataset with novel facts using made-up entities and relations. For each dataset, we define an **attribution set** containing all LM training examples that might be considered proponents and a **query set** containing test examples, each annotated with their ground truth proponents from the attribution set. The examples in these sets consist of masked inputs, outputs and the fact annotations.

3.1 FTRACE-TREx

We create an attibution set using TREx (Elsahar et al., 2018) and query set using LAMA (Petroni et al., 2019) datasets. TREx consists of DBPedia (Brümmer et al., 2016) abstracts, $a_i \in A$. Each abstract contains a set of sentences, $s_j = a_{ij}$, and each sentence is associated with a set of facts, $F(s_j)$. For each fact $f \in F(s_j)$, TREx annotates

	FTRACE-TREx		FTRACE-Synth	
Statistics	**Attribution**	**Query**	**Attribution**	**Query**
Length	1,560,453	31,479	3,190,000	10,000
Unique Facts	552,381	31,479	50,000	5,000
Avg. #proponents	–	83	–	62
Facts per example	8.28	1	2	1
Unique Predicates	488	41	37	37
Unique Objects	49,166	2,266	5,000	5,000
Unique Subjects	310,197	29,464	5,000	5,000

Table 1: FTRACE-TREx: We extract 1M masked examples from TREx (Elsahar et al., 2018), and match them with 27k queries from LAMA (Petroni et al., 2019) to construct our fact tracing benchmark. FTRACE-Synth: To evaluate influence methods on completely novel facts, we propose a synthetic benchmark consists of made-up entities and relations. Refer to Appendix C.4 and Appendix C.5 for examples.

the exact positions where the subject and object respectively appear in the sentence s_j.

We wish to convert these sentences into training examples that can teach a language model about the facts stated within them. To do so, we construct cloze-style language modeling examples as in masked language modeling (Devlin et al., 2019) or span corruption (Raffel et al., 2020). In particular, for each fact f in a sentence s, we mask out either the subject or the object, and train the model to predict it. The two resulting examples $\text{mask}_{\text{sub}}(s, f)$ and $\text{mask}_{\text{obj}}(s, f)$ are marked as "proponents" of the fact, as shown in Figure 2.

The LAMA dataset is anchored to the same fact tuples used by TREx. For each fact tuple, LAMA provides a template-generated sentence expressing the fact. Similar to TREx, we convert this sentence into a cloze-style example by either masking out the subject or object. Hence, we now have two sets of examples (TREx and LAMA) that express the same facts. We treat the TREx examples as our attribution set and the LAMA examples as our test set. Since we wish to trace influence from LAMA back to TREx, we sometimes refer to LAMA examples as "queries" and TREx examples as "retrieval candidates." For any LAMA example, we define the ground-truth proponents as simply the TREx examples that express the same fact.

One ambiguity remains regarding ground truth in TREx sentences that express multiple facts. Suppose a TREx sentence expresses facts f_1 and f_2, and we generate cloze examples for both f_1 and f_2. The example $\text{mask}_{\text{sub}}(s, f_1)$ is clearly a proponent of f_1, but it is perhaps also a proponent of f_2, since the text supporting f_2 is still present after masking. Ultimately, we care about whether attribution methods can retrieve the right *sentence* from the attribution set, not a particular *masking* of that sentence. In our evaluations (described next), we evaluate a method's ability to retrieve at the *sentence level*, with the score of a sentence defined as the max score over all maskings of that sentence.

In total (Table 1), we match approximately **448k** TREx sentences with **31k** LAMA queries. On average, each TREx example expresses three facts, and each LAMA example has 83 proponents (including different maskings of the same sentence).

3.2 FTRACE-Synth

In a dataset with real facts, two factors can negatively impact TDA methods compared to baselines such as BM25: First, many of the facts in FTRACE-TREx may already be known by a pretrained LM.[3] In such cases, the LM will not learn the fact from TREx, and TDA methods should not be expected to identify examples in TREx as proponents. We refer to this as the "saturation" problem, since the model's performance already saturated on the fact before fine-tuning, leaving no signal for TDA methods to detect. Second, real corpora like TREx and LAMA have lexical overlap between query and attribution examples (overlapping surface forms; see Section 3.1) which can favor counting-based methods like BM25.

To better evaluate TDA methods in isolation, we create a synthetic dataset, FTRACE-Synth with facts that are guaranteed to be novel. First, we create random entities with a total number comparable to TREx. Then, we randomly relate those entities with each other using the same set of relations from the TREx dataset.

Entities Our entity list consists of 5,000 synthetic entities each uniquely identified by a number. To reduce the lexical overlap between examples in the dataset, we use 4 surface forms per entity – 2 forms with Arabic numerals, 2 forms with Roman numerals. For example, the fourth entity appears with the following surface forms: ["4-entity", "entity-4", "IV-entity", "entity-IV"].

Relations The dataset includes a set of 37 relations (Appendix B) borrowed directly from TREx. Additionally, we paraphrase each relation to create

[3]Even if the answer is not ranked first among model outputs, the correct prediction may be "close to the surface" on these examples, and the contribution of fine-tuning may be small, even if the predictions flip.

diversity and to reduce the lexical overlap between attribution and query examples.

Attribution Set Each example in the attribution corpus expresses two facts to parallel the multi-fact nature of TREx examples.

Input: entity-MMCLXXIV is the official language of $\_\_\_\__1$, CMXCVII-entity is the writing place of $\_\_\_\__2$
Output: 1:3082-entity, 2:entity-MMMCCC

The attribution corpus includes 50,000 individual facts. By masking different entities as well as combining different facts, we can generate 3,190,000 masked examples for the attribution corpus.

Query Set Similar to LAMA, each example in the query corpus queries a single fact expressed as a masked example, for example:

Input: entity-3300 was written in _____.
Output: entity-CMXCVII

We generate 5,000 such facts by assigning random relations between different entities, with two surface forms for each, resulting in 10,000 examples. As a result, each fact in this query set has 62 proponents in the attribution corpus, and every entity appears in 10 relations on average.

4 Experimental Setup

Our experiments aim to answer the questions of (1) whether TDA methods can be used as effective fact tracing tools (compared to simple IR baselines), (2) which configurations make them most effective (exploring many variations), and (3) analyzing the weaknesses of TRACIN, in particular its sensitivity to *when* the knowledge is learned (the aforementioned “saturation” hypothesis).

4.1 Reranking Evaluation

Ideally, an attribution method would score a given test query against every training example, and we can sort all examples by their influence score. This would enable evaluation with standard IR methods like recall@10 and mean reciprocal rank (MRR) $\frac{1}{|Q|}\sum_{q\in Q}\frac{1}{\mathrm{rank}_q}$, where rank_q is the rank of the first true proponent for the query, and Q denotes the candidate set. However, most attribution methods are computationally intractable for scoring all training sentences in large datasets. Although we can reduce the complexity of some of these methods through the use of random projections (Pruthi et al., 2020), such lossy approximations would render our results less conclusive, as it would be unclear whether an outcome is due to the intrinsic quality of a method or the quality of the projection.

Therefore, to achieve computational tractability while avoiding such confounds, we propose a simple reranking setup: instead of scoring all examples, we can score a carefully selected subset that still enables meaningful comparisons. We call this the “candidate set”. It is the union of four sets:

1. all true proponents for a query: $\mathcal{P}(z_{\text{query}})$,
2. the top-100 retrievals from BM25: $\mathrm{BM25}(z_{\text{query}})$,
3. 100 random examples that share the same target y as the query: $\mathcal{D}_y = \{(x, y) \text{ s.t. } y = y_{\text{query}}\}$, and
4. 100 randomly sampled examples: $\mathcal{D}_{\text{random}}$,

with random samples fixed across all evaluations. Note that MRR on this particular candidate set is an **upper-bound** on the MRR over the full attribution set. Because it includes all proponents but fewer distractors, rank is guaranteed to be closer to 1 in the MRR definition. Also, including $\mathcal{P}(z_{\text{query}})$ is necessary to ensure that the model has the opportunity to retrieve all proponents. $\mathrm{BM25}(z_{\text{query}})$ ensures that we have “distractors” with high lexical overlap, and $\mathcal{D}_y$ is included because we observed that TDA methods have a tendency to retrieve examples with the same output as the query.

Our candidate set includes all top retrievals from BM25, so the results for BM25 are exact. When combined with the fact that reranking MRRs always upper-bound full retrieval MRRs, our setup guarantees that any method that underperforms BM25 on reranking also underperforms on full retrieval.

Slicing examples The gradient-based methods require careful treatment when considering models that go through two separate stages: pre-training and fine-tuning. For example, if a model has already obtained zero loss on an example at the start of fine-tuning, then the gradient will be near-zero throughout fine-tuning, and computing influence using only fine-tuning checkpoints will yield an uninformative influence score for any query. We refer to this problem as “saturation.” To mitiagate saturation, we evaluate TDA methods on a subset of queries we label **Finetune-learned (FL)**, where the model failed before fine-tuning (the answer is not in top-3 beam-search predictions), but succeeded afterward (the answer is ranked first in the beam). We referred to this set as "novel facts" in Section 1.[4]

4.2 Model

We use MT5-base, a commonly used encoder-decoder language model (Xue et al., 2021) to evaluate the aforementioned neural TDA methods. MT5 was pre-trained on the MC4 corpus, which includes all of Wikipedia, and therefore was exposed to the knowledge expressed in FTRACE-TREx. The pre-trained MT5 model achieves 24.3% top-3 accuracy when predicting answers to the TREx queries. Fine-tuning MT5 on our FTRACE-TREx training set increases accuracy to 47.42%, suggesting that there are still many facts MT5 did not know after pre-training. For FTRACE-SYNTH, the pre-trained model gets 0 accuracy as expected, and the fine-tuned model obtains 81%[5].

To evaluate TRACIN, we approximate Equation (3) by choosing three checkpoints that are uniformly spaced out in terms of their training loss (specifically, inverse perplexity), to ensure that we cover significant parts of training while favoring regions with greater loss reduction. Note that we use pre-training checkpoints when evaluating the pre-trained model, and fine-tuning checkpoints when evaluating the fine-tuned model; see Appendix A for details. We calculate the gradient w.r.t the average negative likelihood of the true output token sequence. To evaluate embedding-based fact tracing, we use representations from the final checkpoint of the model.

For both gradient and embedding-based methods, we present the best layer combination among the different concatenations of layers studied in (Section 5.2).

5 Results

5.1 Top-level comparisons

In Table 2, we present a top-level comparison of the three main methods discussed (gradient-based, embedding-based, and BM25). Hyperparameters for all methods have been optimized. As we discuss

Methods	**MRR**		**Recall@10**	
Random-Target	14.50 ±0.95		10.32 ±1.54	
BM25	**77.55** ±1.50		60.89 ±2.31	
	Finetuned	**Pretrained**	**Finetuned**	**Pretrained**
TRACIN	48.56 ±4.40	62.38 ±1.99	56.02 ±0.67	57.54 ±1.25
EMBED	64.29 ±1.32	60.59 ±1.13	57.89 ±1.38	54.91 ±0.32
TRACIN + EMBED	58.52 ±3.83	67.66 ±0.22	61.72 ±0.08	61.59 ±1.35

Table 2: Top Level Results: Best scores for each method and model on the **Finetune-learned** slice of **FTRACE-TREx**. We present average sentence-level retrieval results over 3 random selections of 200 queries. We found that BM25 performs best in MRR outperforming neural methods. Table 6 shows detailed MRRs on predicate, subject, and object level of candidate examples.

in subsequent sections, TDA hyperparameters have a significant effect on performance.

We optimized TRACIN by rescaling gradients with Adafactor accumulators (Shazeer and Stern, 2018), applying unit-normalization to the gradients (see Table 3) and selecting the best layer configuration (Section 5.2). To sanity check that TDAs are doing more than matching the query's output label, we compare to a RANDOM-TARGET baseline that outputs a score of 1 for all training examples with the same output label. This baseline is indeed substantially worse than either method.

Despite extensive optimization for TRACIN and EMBED, however, we found that BM25 with no tuning still outperforms neural TDAs in MRR and Recall@10. TRACIN slightly outperforms EMBED for pretrained model but significantly underforms EMBED for the finetuned model. When we ensemble TRACIN and EMBED (by summing their influence scores) there are improvements on both metrics, demonstrating that their success is somewhat orthogonal. We provide example retrievals from all three models in Appendix C.

Surprisingly, pre-trained TRACIN outperforms fine-tuned TRACIN in this dataset, as we discuss more in Section 5.3.

We do not seek to measure all benefits of attribution methods, but rather to assess one **expected** function (fact-tracing), as promised by their stated goal (tracing a model's prediction back to data). The fact that even the best TDA method obtains MRR of 67.66 and Recall@10 of 61.59 showcases the significant *absolute* headroom that remains for attribution methods . BM25 results are only a little better, and are provided mainly as a reference point. Next, we analyze what choices contributed to the current status of TDA methods with a detailed ex-

[4]In Appendix C.2, we present additional results for **Pretrain-learned (PL)** examples, which went from failing to successful during pre-training rather than fine-tuning.

[5]We accept an answer if any of the surface forms of the correct entity is the output.

	MRR		Recall@10	
Change	**Finetuned**	**Pretrained**	**Finetuned**	**Pretrained**
Adafactor → no-Adafactor	–3.83 ±4.81	–7.20 ±2.25	–11.29 ±2.05	–2.36 ±1.63
unit-norm → no-norm	–3.36 ±4.89	–32.90 ±2.13	–10.82 ±2.24	–28.06 ±1.46
multi-ckpt → single-ckpt	+0.51 ±6.42	+0.60 ±2.23	–6.95 ±4.72	+5.44 ±1.61
no [eos] → [eos]	+5.50 ±4.63	–24.77 ±3.82	+12.96 ±1.59	–19.93 ±3.49

Table 3: Our experiment with various configurations for best layer of the TracIn evaluted in Finetune-Learned set of FTRACE-TREx: For each change from the best configuration (the first row), we report the best result by optimizing free hyper-parameters. The normalization and the usage of the accumulator smoothing was effective in our top level TracIn results. We compare maximum scored checkpoint scores to our original multi-checkpoint results, we found that the best checkpoint performs slightly better than multi-heckpoint results in MRR. The including the the end of sentence token in the target side hurts pretrained MT5 model since it is originally trained to predict multiple answer.

	MRR	**Precision @10**	**Recall@10**
Random-Target	36.47 ±2.84	30.43 ±4.00	2.45 ±0.32
BM25	87.69 ±1.71	52.02 ±2.65	4.20 ±0.21
TracIn	100.00 ±0.00	99.50 ±0.14	8.02 ±0.01
Embed	99.58 ±0.24	97.12 ±0.53	7.83 ±0.04
TracIn + Embed	100.00 ±0.00	98.07 ±0.18	7.91 ±0.01

Table 4: Synthetic Dataset Results: Best scores for fine-tuned model on the **Finetune-learned** slice of FTRACE-Synth. On completely novel facts, the TracIn upperbound is higher than the other methods. Since we control the lexical overlap, BM25 underperforms neural methods. We present average sentence-level retrieval results over 2 random selections of 200 queries. The upper-bound scores on neural methods are higher in the synthetic data than BM25 as we reduce the lexical overlap. The TracIn upperbound performs best in all the metrics.

ploration of hyperparameters.

5.2 Which Transformer layers provide the most reliable attribution signal?

Some layers of a language model may be specialized for operations that have no relation to factual information. For example, previous probing work (Tenney et al., 2019) shows the existence of layers that focus on syntax rather than on knowledge. The contribution of such layers to TracIn may introduce noise. In Figure 3, we conduct an experiment where we sweep over various subsets of layers.

For TracIn, the best-performing layer is the embedding layer of the model — this result, also observed in Yeh et al. (2022), is surprising, as most prior work used only the last layer. In Embed, the best performing layer is again the output of the embedding layer. These results suggest that much of the effectiveness of embedding-based methods derives from their models of lexical similarity. Conversely, for TracIn, the embedding layer also includes contextual information since the gradient signal propagates back through the entire model.

Additional Model Variants Section 5.1 mentioned several design choices for TracIn. We performed a systematic evaluation of those choices. In Table 3, given the set of configurable options in the table, we set a given option to a particular value and then optimize remaining parameters.

Unit-normalized gradients drastically outperform the dot product. Next, we considered the role of Adafactor during training. The TracIn equation arises from considering parameter updates at a specific step. The true parameter updates were not raw gradients, but gradients that had been rescaled by Adafactor accumulators. Using these rescaled gradient for TracIn performs much better. Also, surprisingly, using the single best-performing checkpoint is sometimes better than using multiple checkpoints. We provide the individual checkpoint results in Table 5.

5.3 FTRACE-Synth and Saturation

As mentioned earlier, TracIn monitors the change in a model's performance on a test query over the course of training — and therefore is likely to fail if a test query's loss is already zero (saturated) at the start of the training period monitored by TracIn. In addition, because the pre-trained model sees very similar sentences and information in the pre-training corpora, the influence could be distributed over multiple examples, such that the signal from each candidate is weak. These confounding factors may apply to FTRACE-TREx. Therefore, we also evaluate TDA methods on our synthetic dataset, FTRACE-Synth, which controls for all these issues. We fine-tune the same model on FTRACE-Synth and perform the same evaluation in Table 4. The results suggest that when the aforementioned factors are controlled, the reranking upper-bound for gradient-based TDA method is better than BM25 and slightly better than embedding-based TDA methods. This result verifies that TDA methods might have advantages over standard IR methods, despite falling short in a more realistic, applied scenario.

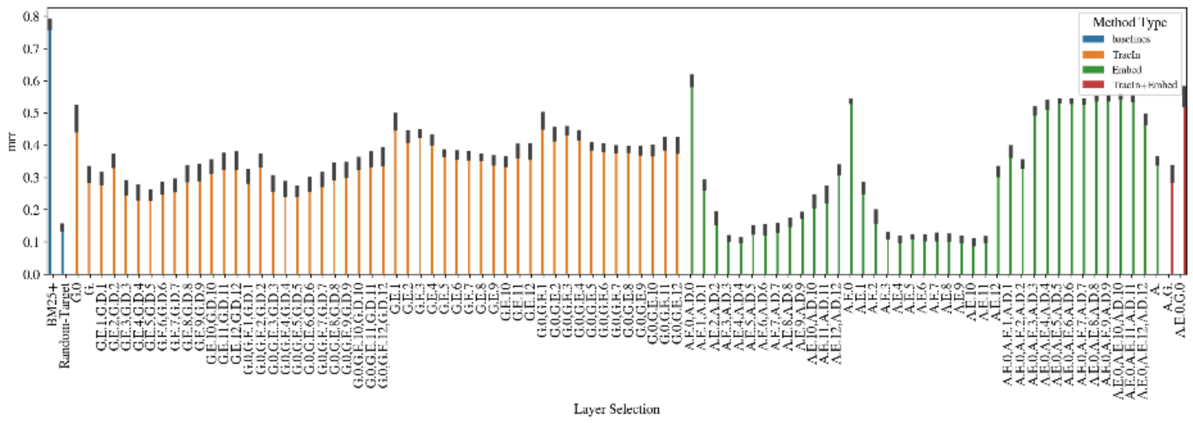


Figure 3: Mean reciprocal rank for TRACIN with different layers and and EMBED from different intermediate layers: In **G.0**, gradient of embedding layer is used. In **G.** and **A.**, respectively, gradients and embeddings of all layers are used. **A.E.0** and **A.D.0** corresponds to embedding layer's output in the encoder and decoder part of the model respectively. Comma-separated labels denote ensembling by summing the scores of the corresponding layers. We report results for 3 random seeds (error bars with standard deviation) of 200 queries where queries learned between pretraining checkpoints. In neural methods, using only the embedding layer or its output performs the best, while underperforming the baseline method BM25.

6 Related work

Information Retrieval To define our fact tracing task, we employ standard concepts from the information retrieval (IR) literature: a retrieval + reranking setup, and standard retrieval metrics. However, while IR focuses on retrieving any document that satisfies a user's query, our benchmark specifically aims to identify examples that caused a particular model to make a particular prediction. This focus on model-specific causality distinguishes us from prior IR work (Thakur et al., 2021; Nguyen et al., 2016). Our evaluation setup should be *easier* than generic IR benchmarks because we are only evaluating on predictions we know the LM gets right.

Language Models as Retrievers Language models have been successfully used in numerous IR applications. Karpukhin et al. (2020) use language model embeddings to warm-start neural retrievers for knowledge-intensive tasks. Guu et al. (2020) and Lewis et al. (2020) show that language modeling and information retrieval can be jointly learned in a manner that benefits both tasks. Our work uses TDA-based retrieval methods to help users understand the behavior of the LMs themselves.

Attribution Methods Recent work has tried to explain neural model behavior in many different ways: (1) attributing a prediction back to specific features in the input (Simonyan et al., 2014; Sundararajan et al., 2017; Han et al., 2020), (2) attributing to specific model parameters (Dai et al., 2022; Mitchell et al., 2022), (3) probing for competence at linguistic sub-tasks (Tenney et al., 2019), and finally (4) attributing back to training examples (Pruthi et al., 2020; Koh and Liang, 2017).

However, work in the last category (Han et al., 2020; Guo et al., 2021; Zhang et al., 2021) has been limited, mainly focusing on classification and regression tasks that do not involve questions about factuality or world knowledge. Consequently, these methods have primarily been used as a data cleaning technique, leaving the question of fact tracing unexplored (Han et al., 2020; Hara et al., 2019).

7 Conclusion

We introduced a new dataset and benchmark for *fact tracing*: the task of tracing language models' assertions back to the training examples that provided evidence for those predictions. We evaluated *gradient*-based and *embedding*-based TDA methods and found that they perform worse than a standard IR baseline (BM25) even in settings that favor TDA methods. We investigated the effects of layer selection, model checkpoints and fine-tuning. Our ablative analysis suggests that saturation is an important factor affecting the performance of current methods. Much is needed to improve these methods before they can be reliably used for fact tracing. We hope that this benchmark will enable future research on fact tracing, by mitigating computational challenges and establishing a principled ground truth.

Acknowledgments

We would like to thank Zhuyun Dai, Keith B. Hall, Ji Ma for their helpful discussions and feedbacks.

Limitations and Impact Statement

Our experiments focus on a single representative language model, MT5-base; it is possible that our findings about the effectiveness of attribution methods for fact tracing would differ substantially when applied to language models with very different architectures or trained on different datasets. Because of the candidate set construction scheme described in Section 4.1, these results only upper-bound the performance of evaluated methods, and it is also possible that they are even less effective than reported here. The ground truth labels in FTRACE-TREx extracted from TREx where the fact annotations are *semi-automatically* annotated, can have labeling errors.

The FTRACE dataset includes content from Wikipedia, some of which has not been vetted for factual accuracy. It is possible that by redistributing this content we will propagate misinformation. We plan to mitigate this harm with a datasheet that explicitly communicates FTRACE's role as an evaluation tool, and not as a reliable source of information. Apart from the dataset, we anticipate no ethical issues associated with the techniques described in this publication.

References

Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. 2020. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*, pages 1899–1909. PMLR.

Samyadeep Basu, Phil Pope, and Soheil Feizi. 2021. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*.

Martin Brümmer, Milan Dojchinovski, and Sebastian Hellmann. 2016. Dbpedia abstracts: A large-scale, open, multilingual nlp training corpus. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 3339–3343.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-rex: A large scale alignment of natural language with knowledge base triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. 2021. FastIF: Scalable influence functions for efficient model interpretation and debugging. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10333–10350, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Frank R Hampel. 1974. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393.

Xiaochuang Han and Yulia Tsvetkov. 2021. Influence tuning: Demoting spurious correlations via instance attribution and instance-driven updates.

Xiaochuang Han, Byron C. Wallace, and Yulia Tsvetkov. 2020. Explaining black box predictions and unveiling data artifacts through influence functions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5553–5563, Online. Association for Computational Linguistics.

Satoshi Hara, Atsushi Nitanda, and Takanori Maehara. 2019. Data cleansing for models trained with sgd. *Advances in Neural Information Processing Systems*, 32.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of* *the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR.

Katherine Lee, Orhan Firat, Ashish Agarwal, Clara Fannjiang, and David Sussillo. 2018. Hallucinations in neural machine translation. In *Workshop on Interpretability and Robustness for Audio, Speech, and Language at NeurIPS*.

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. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Yuanhua Lv and ChengXiang Zhai. 2011. Lower-bounding term frequency normalization. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 7–16.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale. In *International Conference on Learning Representations*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. Ms marco: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. 2020. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nazneen Fatema Rajani, Ben Krause, Wengpeng Yin, Tong Niu, Richard Socher, and Caiming Xiong. 2020. Explaining and improving model behavior with k nearest neighbor representations. *arXiv preprint arXiv:2010.09030*.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline M Hancock-Beaulieu, Mike Gatford, et al. 1995. Okapi at trec-3. *Nist Special Publication Sp*, 109:109.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. 2022. Scaling up influence functions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8179–8186.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604. PMLR.

K Simonyan, A Vedaldi, and A Zisserman. 2014. Deep inside convolutional networks: visualising image classification models and saliency maps. pages 1–8. Proceedings of the International Conference on Learning Representations (ICLR).

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. Bert rediscovers the classical nlp pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Ran Tian, Shashi Narayan, Thibault Sellam, and Ankur P Parikh. 2019. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv:1910.08684*.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 483–498, Online. Association for Computational Linguistics.

Chih-Kuan Yeh, Ankur Taly, Mukund Sundararajan, Frederick Liu, and Pradeep Ravikumar. 2022. First is better than last for training data influence. *arXiv preprint arXiv:2202.11844*.

Wei Zhang, Ziming Huang, Yada Zhu, Guangnan Ye, Xiaodong Cui, and Fan Zhang. 2021. On sample based explanation methods for NLP: Faithfulness, efficiency and semantic evaluation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5399–5411, Online. Association for Computational Linguistics.

Appendix

In this appendix, we will provide implementation details and additional results for the experiments.

A Implementation Details

BM25 We use the following BM25 formula:

$$\mathcal{I}(z, z_{\text{query}}) = \sum_{t \in z_{\text{query}}} \log\left(\frac{N+1}{N_t}\right) \times \left(\frac{(k_1+1) \cdot f(z,t)}{k_1 \cdot \left((1-b) + b \cdot \left(\frac{L(z)}{L_{\text{avg}}}\right)\right) + f(z,t)} + 1\right)$$

where, f(z, t) is the overlap count, N is the number of training examples, $L(z)$ is the length of the example, and L_{avg} is the average example length. k_1 and b are hyperparameters tha reweights the importance of the other terms in the formula. Robertson et al. (1995) provides the intuition behind this definition of relatedness.

We use a publicly available BM25+(Lv and Zhai, 2011) implementation written in python and released under `https://pypi.org/project/rank-bm25/`. We tokenize queries and retrieval examples by space and we remove masked tokens. We did not optimize any of the default hyper parameters.

MT5 Model We use intermediate checkpoints of MT5 model [6] (12 layers transformer with 580M parameters). We convert these checkpoints to Pytorch by using HugginFace's T5 converter. We use the tokenizer provided. In our datasetSection 3.1, we use `extra_id_0` for the mask token compatible with pretraining corpus of MT5..

TRACIN We calculate gradients by using Pytorch without batching examples and by using average negative likelihood over output sequence. We store each individual parameter's gradient (blocks of transformer) in a dictionary structure. Given a query and a retrieval example, we calculate scores Equation (4) for each parameter seperately that means we locally normalize each parameters' gradient in Equation (4). Then, to calculate a layer's or full model's score, we score individual scores corresponding to parameters in that layer. This enable us to sweep over different combination of layers as in Figure 3 without rerunning the model.

Pretrained MT5 model is trained until 80k gradient steps. We use checkpoints at 5100, 10200, 15300 steps. We fine-tune MT5 model on additional 60k gradient steps on TREx dataset. Then, we use checkpoints at 5000, 10000, 30000 steps.

We paralelize over checkpoints when calculating Equation (4). For each query, we spend approximately 15 minutes by using VOLTA V100 32 GB GPUs to get scores for all the retrieval examples in the ranking set (Section 4.1))

EMBED Transformer model's forward pass can be expressed as following pseudo code:

$$\begin{aligned} \text{enc}_0 &= \text{Embedding}(x) \\ \text{enc}_i &= \text{Encoder}_i(\text{enc}_{i-1}) i = 1..N \\ \text{dec}_0 &= \text{Embedding}(y) \\ \text{dec}_i &= \text{Decoder}_i(y, \text{enc}_N) i = 1..N \\ \mathcal{L} &= \text{NLL}(W_{\text{proj}} \text{dec}_N, y_{\text{query}}) \end{aligned} \tag{6}$$

We use enc_i and dec_i, and reduce (average) them over time-steps in input and outputs side respectively.

[6]https://github.com/google-research/multilingual-t5

B Synthetic Data Relation Templates

Below are the relation templates we use in the dataset. “0 and “1" are the slots for the entities. Paraphrases are delimited by “|" sign. Left paraphrase is the original surface for in the FTRACE-TREX dataset, right one is the additional paraphrase paraphrase.

```
{0} was born in {1} | {0}’s birth place is {1}
{0} died in {1} | {0} passed away in {1}
{0} is a subclass of {1} | {1} is superclass of {0}
The official language of {0} is {1} | {1} is the official language of {0}
{0} plays in {1} position | {1} is the play position of {0}
{0} was awarded the {1} | {1} given to {0}
{0} was originally aired on {1} | {1} is the first streamer of {0}
{0} was educated at the University of {1} | {0} studied in University of {1}
{0} shares border with {1} | {0} and {1} are neighbours
{0} is named after {1} | {1} was inspirational for the naming of {0}
The original language of {0} is {1} | {1} is the original language of {0}
{0} plays with {1} | {0} plays along with {1}
{0} is a member of {1} | {1} accepted {0} as a member
{0} works in the field of {1} | {1} is the work field of {0}
{1} participated in the {0} | {1} was a participant of {0}
{0} is a {1} by profession | {0}’s profession is {1}
{0} consists of {1} | {0} includes {1}
{0} is a member of the {1} political party | {0}’s political party was {1}
{0} maintains diplomatic relations with {1} | {0}’s diplomacy with {1}
{0} is produced by {1} | {1} produced {0}
{0} is a citizen of {1} | {0}’s home country is {1}
{0} was written in {1} | {1} is the writing place of {0}
{0} is located in {1} | {0} placed in {1}
{0} is developed by {1} | {1} developed {0}
{0} is the capital of {1} | the capital of {1} is {0}
{0} works for {1} | {0} works at {1}
{0} plays {1} music | {0} perform {1} music
{0} has the position of {1} | {0}’s position is {1}
{0} is represented by music label {1} | music label {1} represents {0}
{0} used to work in {1} | {1} is ex-workplace of {0}
{0} is affiliated with the {1} religion | {0} believes in {1} religion
{0} is owned by {1} | {1} owned {0}v
The native language of {0} is {1} | {1} is the native language of {0}
{0} and {1} are twin cities | {0} is twin city of {1}
{0} is a legal term in {1} | {0} is a legal definition in {1}
The headquarter of {0} is in {1} | {0}’s headquarter in {1}
{0} was founded in {1} | {0} was established in {1}
```

C Additional Results and Samples

C.1 Individual Checkpoints

	MRR		Recall@10	
	FT	PT	FT	PT
Multi	48.56±4.40	62.38±1.99	56.02±0.67	57.54±1.25
Ckpt1	49.07±4.67	54.77±1.26	49.07±4.67	54.77±1.26
Ckpt2	47.30±2.88	62.98±1.01	47.30±2.88	62.98±1.01
Ckpt3	48.69±5.19	60.29±3.34	48.69±5.19	60.29±3.34

Table 5: TRACIN results for individual checkpoints on Finetune-Learned set.

C.2 MRR Results with Submetrics

MRR on Finetune-Learned Subsets of FTRACE-TREx We provide submetrics for (**Finetuned-learned (FL)**) set.

Table 6: MRR Results with submetrics in fine-tuned learned set. (see Table 2)

	Sentence (Table 2)		Predicate		Subject		Object	
	FT	PT	FT	PT	FT	PT	FT	PT
Random-Target	14.50±0.95	14.50±0.95	14.71±0.89	14.71±0.89	98.14±0.72	98.14±0.72	63.56±2.53	63.56±2.53
BM25	77.55±1.50	77.55±1.50	79.26±2.82	79.26±2.82	88.25±1.80	88.25±1.80	85.71±1.22	85.71±1.22
TracIn	48.56±4.40	62.38±1.99	49.16±4.76	63.98±0.98	99.53±0.43	86.49±1.22	88.74±1.59	74.99±3.61
Embed	64.29±1.32	60.59±1.13	66.25±1.82	63.00±1.73	94.09±0.77	81.79±1.33	80.45±0.99	74.03±2.27
TracIn + Embed	58.52±3.83	67.66±0.22	59.24±3.88	69.49±0.92	97.92±0.50	82.03±1.61	71.94±2.44	79.15±1.55

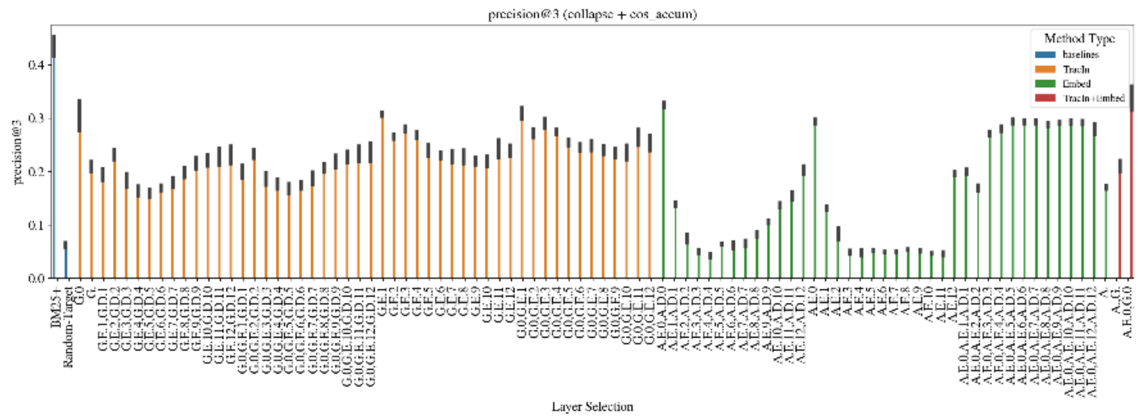
MRR on Pretrained-Learned Subsets of FTRACE-TREx We present additional results for (**Pretrain-learned (PL)**) examples where the model failed before the a checkpoint of pre-training, but changed during pre-training. We found that the average number of proponents in the PL set is 2.5x that of the FL set (since we expect that frequently mentioned facts will be learned first). These results suggest that it's difficult to control for when facts were learned without affecting the other statistics, and that direct comparisons between model performance on the PL and FL datasets may not be informative.

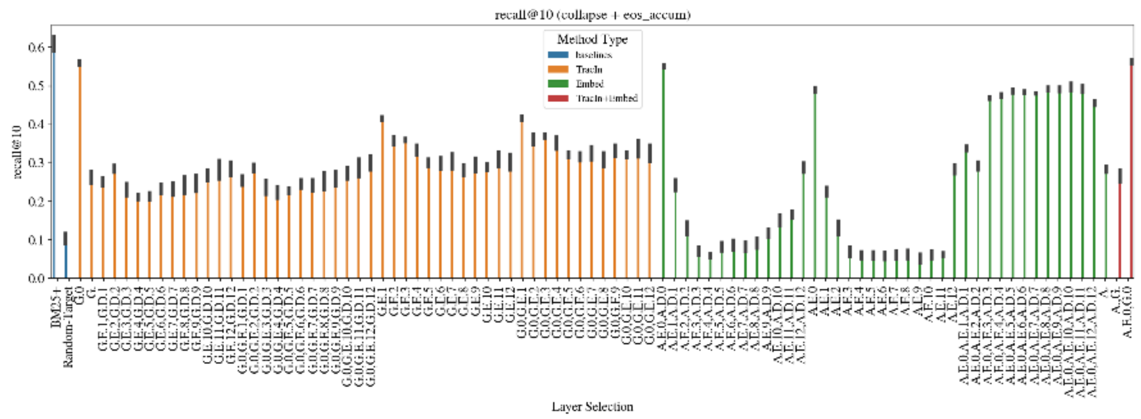
Table 7: MRR Results with submetrics in pretrained- learned set. (see Table 2)

	Sentence (Table 2)		Predicate		Subject		Object	
	FT	PT	FT	PT	FT	PT	FT	PT
Random-Target	15.83±1.42	15.83±1.42	15.62±1.29	15.62±1.29	98.36±0.79	98.36±0.79	61.88±1.99	61.88±1.99
BM25	77.62±3.24	77.62±3.24	77.20±3.56	77.20±3.56	91.24±0.83	91.24±0.83	88.07±1.39	88.07±1.39
TracIn	64.18±2.62	54.45±1.96	63.94±1.80	56.01±1.95	99.17±0.72	89.82±2.21	88.07±2.00	81.44±1.71
Embed	51.21±2.43	50.42±2.43	51.02±2.55	50.30±2.64	96.52±1.75	84.21±3.03	79.18±1.05	79.00±0.82
TracIn + Embed	65.91±2.88	55.40±1.98	66.04±2.60	57.09±2.24	97.95±0.25	86.21±2.16	84.09±2.00	82.01±1.78

C.3 Precision-Recall plots for FTRACE-TREx

We present accompanying precision and recall results for Figure 3.





C.4 Samples for FTRACE-TREx

Here, we provide example top-3 retrievals from TDAs for the FTRACE-TREx dataset. Long examples are truncated for display purposes. We provide label (whether the retrieved example includes the fact) next to the output of the retrieved example.

Embed	TracIn	BM25
Q: In late 2005, the $\underline{\hspace{2em}}_1$ broadcast a full series of Star Spell, again presented by Eamonn Holmes but Mishal Husain took over from Nina as word pronou... **A**: BBC **True**	**Q**: In late 2005, the $\underline{\hspace{2em}}_1$ broadcast a full series of Star Spell, again presented by Eamonn Holmes but Mishal Husain took over from Nina as word pronou... **A**: BBC **True**	**Q**: In late 2005, the $\underline{\hspace{2em}}_1$ broadcast a full series of Star Spell, again presented by Eamonn Holmes but Mishal Husain took over from Nina as word pronou... **A**: BBC **True**
Q: The Vicar of Dibley is a $\underline{\hspace{2em}}_1$ television sitcom created by Richard Curtis and written for actress Dawn French by Curtis and Paul Mayhew-Archer, wit... **A**: BBC **False**	**Q**: Tasneem Zehra Husain (also spelled as Tasneem Zehra Hussain), is a Pakistani $\underline{\hspace{2em}}_1$ and an Assistant Professor of Physics at the Lahore University of... **A**: theoretical physicist **False**	**Q**: In late 2005, the BBC broadcast a full series of Star Spell, again presented by Eamonn Holmes but $\underline{\hspace{2em}}_1$ took over from Nina as word pronouncer. **A**: Mishal Husain **True**
Q: Honigberg also recorded Homage to Rostropovich (1927–2007), a CD of solo cello works written for the legendary cellist; Frédéric Chopin's complete wor... **A**: piano **False**	**Q**: Abdul Aziz Bin Dato Haji Husain was born 18 July 1950 in Kuching, Sarawak, $\underline{\hspace{2em}}_1$. **A**: Malaysia **False**	**Q**: He now works for the BBC, presenting on the BBC News channel and $\underline{\hspace{2em}}_1$. **A**: BBC One **False**

Table 8: Mishal Husain works for $\underline{\hspace{2em}}_1$. (A: BBC)

C.5 Samples for FTRACE-Synth

Now, we provide the retrieved examples for FTRACE-SYNTH version of our dataset.

Embed	TracIn	BM25
Q: Clara Ellaline Hope Leighton (sometimes Clare Veronica Hope Leighton) (12 April 1898 - 4 November 1989) was an ____$_1$/American artist, writer and ill... **A**: English **False**	**Q**: He was educated in ____$_1$ and at the Quaker Leighton Park School. **A**: London **False**	**Q**: The ____$_1$ Kenneth Leighton (1929–1988) also wrote a Fantasia Contrappuntistica ("Homage to Bach", Op.24) for piano, which won the first prize at the... **A**: composer **True**
Q: Lillianne Brown Leighton (May 17, 1874 – March 19, 1956), known professionally as Lillian Leighton, was an ____$_1$ silent film actress. **A**: American **False**	**Q**: Kenneth ____$_1$ Bray (May 26, 1895 – January 9, 1953) was an Episcopal priest, teacher, sportsman and coach. **A**: Augustine **False**	**Q**: The composer ____$_1$ (1929–1988) also wrote a Fantasia Contrappuntistica ("Homage to Bach", Op.24) for piano, which won the first prize at the Bolzano... **A**: Kenneth Leighton **True**
Q: The composer Kenneth Leighton (1929–1988) also wrote a Fantasia Contrappuntistica ("Homage to Bach", Op.24) for ____$_1$, which won the first prize at ... **A**: piano **True**	**Q**: Leighton Road Evangelical Church is a nonconformist independent evangelical church located on the Gainsborough estate, ____$_1$ in the English county o... **A**: Ipswich **False**	**Q**: The composer Kenneth Leighton (1929–1988) also wrote a Fantasia Contrappuntistica ("Homage to Bach", Op.24) for ____$_1$, which won the first prize at ... **A**: piano **True**

Table 9: Query: Kenneth Leighton plays ____$_1$. (A: piano)

Embed	TracIn	BM25
Q: ____$_1$ given to 3692-entity,entity-2686 was awarded the ____$_2$ **A**: 1:entity-1138, 2:entity-MMMMDCLIII **True**	**Q**: ____$_1$ given to entity-MMDCLXXXVI,____$_2$ used to work in CXVI-entity **A**: 1:entity-MMMMDCLIII, 2:entity-1650 **True**	**Q**: ____$_1$ given to 3692-entity,____$_2$ was awarded the entity-MMMMDCLIII **A**: 1:entity-1138, 2:entity-2686 **True**
Q: entity-1138 given to ____$_1$,entity-2686 was awarded the ____$_2$ **A**: 1:3692-entity, 2:entity-MMMMDCLIII **True**	**Q**: entity-CCCII given to ____$_1$,MMMDLVI-entity given to ____$_2$ **A**: 1:entity-MMMMDCLIII, 2:entity-MDCCCLXXXVII **False**	**Q**: entity-CCCII given to ____$_1$,____$_2$ given to entity-MDCCCLXXXVII **A**: 1:entity-MMMMDCLIII, 2:MMMDLVI-entity **False**
Q: ____$_1$ given to 2686-entity,____$_2$ plays in entity-2658 position **A**: 1:MMMMDCLIII-entity, 2:MMMMCCLXIX-entity **True**	**Q**: ____$_1$ shares border with entity-DCCXXVII,entity-302 given to ____$_2$ **A**: 1:entity-MMMMCMXCVIII, 2:entity-MMMMDCLIII **False**	**Q**: entity-CCCII given to ____$_1$,MMMDLVI-entity given to ____$_2$ **A**: 1:entity-MMMMDCLIII, 2:entity-MDCCCLXXXVII **False**

Table 10: ____$_1$ given to entity-2686. (A: entity-MMMMDCLIII)

Embed	TracIn	BM25
Q: entity-MMMDLXXVI's diplomacy with $\_\_\_\__1$,entity-3193's birth place is $\_\_\_\__2$ **A**: 1:MMCDLX-entity, 2:entity-5 **True**	**Q**: 3132-entity maintains diplomatic relations with $\_\_\_\__1$,$\_\_\_\__2$ was awarded the entity-3701 **A**: 1:entity-3468, 2:entity-4097 **False**	**Q**: MMMDLXXVI-entity's profession is $\_\_\_\__1$,entity-CCLXI's diplomacy with $\_\_\_\__2$ **A**: 1:MXXX-entity, 2: 506-entity **False**
Q: MMMDLXXVI-entity's diplomacy with $\_\_\_\__1$,$\_\_\_\__2$ given to 2897-entity **A**: 1:entity-MMCDLX, 2:MCMLIII-entity **True**	**Q**: The original language of MMCCLXXIX-entity is $\_\_\_\__1$,3552-entity shares border with $\_\_\_\__2$ **A**: 1:MMCDLX-entity, 2:MMMMDLXV-entity **False**	**Q**: MMMDLXXVI-entity's diplomacy with $\_\_\_\__1$,MCMLIII-entity given to $\_\_\_\__2$ **A**: 1:entity-MMCDLX, 2: 2897-entity **True**
Q: MMMDLXXVI-entity's diplomacy with $\_\_\_\__1$,MCMLIII-entity given to $\_\_\_\__2$ **A**: 1:entity-MMCDLX, 2: 2897-entity **True**	**Q**: The official language of CMXCVII-entity is $\_\_\_\__1$,$\_\_\_\__2$ died in MMCDLX-entity **A**: 1:3215-entity, 2: 710-entity **False**	**Q**: MMMDLXXVI-entity's diplomacy with $\_\_\_\__1$,$\_\_\_\__2$ given to 2897-entity **A**: 1:entity-MMCDLX, 2:MCMLIII-entity **True**

Table 11: MMMDLXXVI-entity's diplomacy with $\_\_\_\__1$. (A: MMCDLX-entity)