You Are My Type! Type Embeddings for Pre-trained Language Models

Mohammed SAEED
EURECOM
mohammed.saeed@eurecom.fr

Paolo PAPOTTI
EURECOM
paolo.papotti@eurecom.fr

Abstract

One reason for the positive impact of Pre-trained Language Models (PLMs) in NLP tasks is their ability to encode semantic types, such as 'European City' or 'Woman'. While previous work has analyzed such information in the context of interpretability, it is not clear how to use types to steer the PLM output. For example, in a cloze statement, it is desirable to steer the model to generate a token that satisfies a user-specified type, e.g., predict a *date* rather than a *location*. In this work, we introduce Type Embeddings (TEs), an input embedding that promotes desired types in a PLM. Our proposal is to define a type by a small set of word examples. We empirically study the ability of TEs both in representing types and in steering masking predictions without changes to the prompt text in BERT. Finally, using the LAMA datasets, we show how TEs highly improve the precision in extracting facts from PLMs.

1 Introduction

Pre-trained language models (PLMs) based on transformers (Vaswani et al., 2017) have achieved state-of-the-art results in several downstream NLP tasks (Devlin et al., 2019; Liu et al., 2020). Being trained in a self-supervised fashion, such models convey, to a certain extent, linguistic (Puccetti et al., 2021; Lin et al., 2019) and factual knowledge (Rogers et al., 2020; Meng et al., 2022). Being able to faithfully extract the desired knowledge is a crucial aspect that has sparked lots of interest (Petroni et al., 2019; Bouraoui et al., 2020).

However, querying the PLM for information is not always reliable and requires more than a manually-written prompt as an input (Petroni et al., 2020). This is opposed to a standard knowledge graph (KG), where users formulate a structured SPARQL query specifying exactly what to expect at the output. For example, the query "`SELECT ?x WHERE wd:Q76 wdt:P26 ?x`" returns the spouse of Barack Obama, "Michelle Obama". In the PLM setting, the SPARQL query could be replaced by a natural-language prompt, such as "The spouse of Barack Obama is [MASK]". While the predictions of the prompt are reasonable (left-hand side of Figure 1), they do not reflect the requirement of getting instances of a specific type (names of people) in the output. In fact, in BERT's top-1 prediction on prompts where the desired output type is a MUSICAL INSTRUMENT (e.g., "Philip Glass plays [MASK]"), more than half of the predictions follow different types such as SPORT ("plays football") and CHARACTER ("plays Hamlet"), instead of the expected "plays piano". Indeed, differently from the KG with typed entities, the type information is dismissed from the input prompt, thus bringing no guarantee about the expected type.

While several works try to remedy this by engineering prompts to satisfy a desired type (Jiang et al., 2020; Shin et al., 2020; Zhong et al., 2021), or relying on external sources to enrich the prompt (Petroni et al., 2020), these approaches do not fully exploit the latent concepts encoded in the PLM (Dalvi et al., 2022). To fill up this gap, we introduce the notion of *Type Embeddings* (TEs). Similar to how positional embeddings in a PLM encode information about the position of a token in an input (Wang and Chen, 2020), TEs encode the expected type information of the output. The definition of a TE requires neither supervised training nor external resources as it simply uses the existing PLM token embeddings, e.g., people names, to obtain type information, e.g., for PERSON. TEs can be then naturally injected into the input embedding layer of a PLM to embody the expected type in the output (right-hand side of Figure 1). Driving the model towards the expected type can help in applications exploiting PLMs, such as data integration (Cappuzzo et al., 2020), data cleaning (Narayan et al., 2022), rule induction (Cui and Chen, 2021), and fact-checking (Lee et al., 2020).

Our contributions can be summarized as follows:

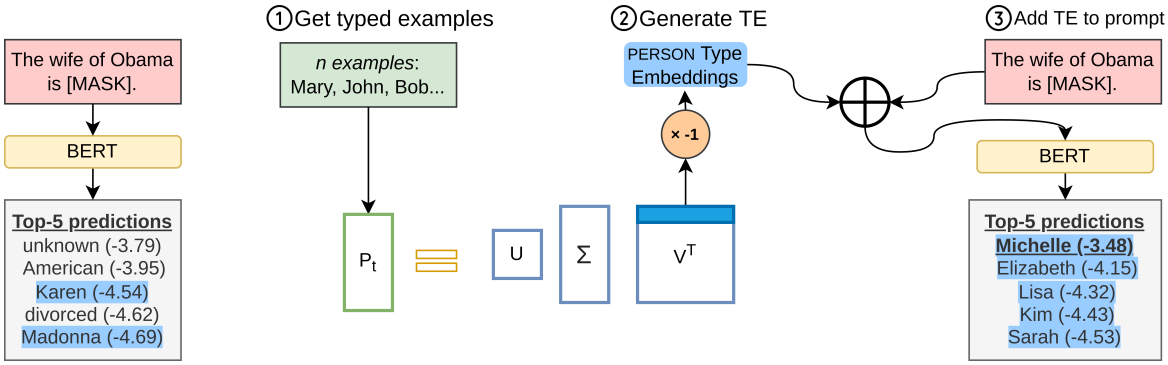

Figure 1: Top-5 predictions of BERT (with log probabilities) for a given prompt (left) and the changes when adding type information (right). Tokens following the desired type are colored. Correct answer is underlined.

- We introduce TYPE EMBEDDINGS (TEs), which, similar to positional embeddings, can be added to the input of PLMs and effectively encode type information. We show how to compute these embeddings using only labeled tokens that adhere to the specific type; the main idea is to remove the first singular vector of the token embedding matrix (Section 3).
- We propose methods to analyze type embeddings and evaluate their effectiveness by (i) measuring their semantic similarity to instances of the type, (ii) assessing the sensitivity of tokens to a given type, and (iii) analyzing layer-wise type classification (Section 4).
- We inject type embeddings into PLMs and show increase in performance for a factual probing dataset (LAMA) and alleviation of "type bias" for a prompt by steering the output type with TEs (Section 5).

We conclude the paper by discussing future directions, including the extension of our approach from types to more generic concepts (Section 6). Data and code for the paper are available at https://github.com/MhmdSaiid/TypeEmbedding.

2 Related Work

PLMs have been largely studied in the last years, with most analysis focusing on the attention mechanism (Voita et al., 2019; Vig and Belinkov, 2019) and on the role of embeddings (Rogers et al., 2020; Li et al., 2021; Clark et al., 2019).

However, none of those efforts study the notion of types that we introduce. One exception is the recent studies of how *concepts* are encoded in PLMs. One work analyzes BERT by clustering contextual representations across layers, followed by a manual annotation to label clusters with meaningful concepts (Dalvi et al., 2022). Another work starts from treating the feedforward network of a transformer as a key-value memory and studies how certain vectors encode concepts in the vocabulary space (Geva et al., 2022). Our effort is different in two ways. First, we do not require the labeling of artifacts from the PLM, but rather we rely on user-specified tokens to model their common type. Second, we focus on type, which is one semantic concept, leaving the others, such as syntactic, morphological, and lexical to future work (Section 6).

Our approach is related to the interpretation of a neural net's internal state in terms of a concept defined through a vector (Kim et al., 2018; Schrouff et al., 2021). The Concept Activation Vector (CAV) is derived from example images by finding the normal to the hyperplane separating examples without a concept and examples with a concept in the model's activation. CAVs separate examples with and without the target concept in a model's activation at a certain layer. By Testing with a CAV (TCAV), one can identify the importance of the color 'red' in fire-engine images for a neural network. We use CAVs on textual input, rather than on images, to measure how sensitive the model is to a type after adding its TE (Section 4.3). However, while CAV is a sensitivity measurement tool, TEs steer the target type in the model's output. A work sharing the same spirit as ours uses a vector to steer output in a PLM for style transfer between sentences (Subramani et al., 2022). However, our method requires only 10 tokens per type as opposed to 100 labeled sentences for style transfer, and it works also with GPT.

Our work introduces a new kind of type embeddings to enrich the input to the PLM, in analogy to positional embeddings (Wang and Chen, 2020; Wang et al., 2021a). We show the benefit of our solution on the LAMA benchmark (Petroni

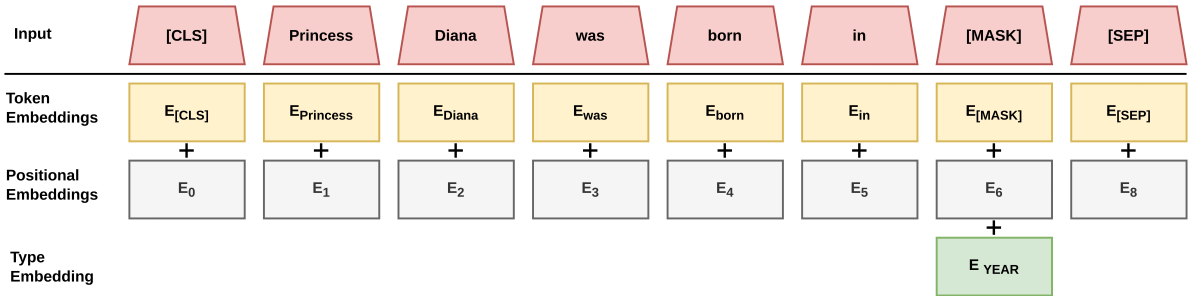

Figure 2: Input representation for a PLM. The YEAR type embedding (green box) is added to the [MASK] token.

et al., 2019), which contains cloze statements to query the PLM for a masked token. To enhance a PLMs' performance for such task, previous work improve prompts by mining or paraphrasing new prompts (Jiang et al., 2020), by adding trigger tokens (Shin et al., 2020), by finding vectors for prompts in the embedding space without restriction to the PLM's vocabulary (Zhong et al., 2021), or by combining multiple prompts (Qin and Eisner, 2021). As we simply add the type embedding to the input, our work is also different from approaches that pre-train an adapter to enhance PLMs' factual knowledge (Wang et al., 2021b) or rely on information retrieval to provide additional context for the prompt (Petroni et al., 2020). Finally, we steer the output while not changing the underlying model by triggering the neurons responsible for a prediction (Dai et al., 2022) or by producing an alternative model with edited facts (De Cao et al., 2021).

3 Type Embedding

In this section, we propose how to compute TEs from PLM token embeddings (Section 3.1), and how to use them (Section 3.2). Following the work on latent concepts in BERT (Dalvi et al., 2022), we focus on such model and report results on other PLMs in Table A2 in the Appendix.

3.1 Obtaining the TE

Given a type t, let the matrix $P_t \in \mathbb{R}^{n \times d}$ hold the token embeddings for n different tokens, where d is the dimension of the token embeddings. The n tokens are instances of a specific type t. We call these tokens *positively typed tokens*.

For our analysis of P_t, we apply *Singular Value Decomposition* (SVD). The SVD of an $m \times n$ matrix M factorizes it into $M = U\Sigma V^T$, where U is an $m \times m$ unitary matrix, Σ is an $m \times n$ diagonal matrix, and V is an $n \times n$ unitary matrix. We call the column vectors of U and V *singular vectors*. The diagonal values in Σ are called *singular values*. Assuming that M is a matrix where each row contains features of a data point, then the first singular vector of V, corresponding to the highest singular value, corresponds to the direction with maximum variance for the covariance matrix. In other words, it is the vector that contains the "common-part" of all data points.

The SVD of the matrix is $P_t = U\Sigma V^T$. The first column of the matrix V, $v^{(1)}$, is the first singular vector, which encodes information common between all n tokens. We hypothesize that this vector, unlike other singular vectors, contains non-type related information and needs to be removed from the input to promote type information encoded in the other singular vectors (more details in Section 4.1). A similar observation has been made for multilingual representations (Roy et al., 2020), where removing r singular vectors leaves semantic-related information in the input representations (Yang et al., 2021). Thus, the embedding to be added to promote type t is $E_t = -\lambda v^{(1)}$, where λ is a multiplier that is tuned on a hold-out dataset.

In practice, a type embedding is derived from a small set of tokens that are instances of the same type. Those can be provided by users, or obtained from existing typed resources such as KGs. In the rest of the chapter, the TEs are computed based on weighted sampling from KG entities. We query DBpedia (Auer et al., 2007) for tokens adhering to a specific type, keeping only those in the PLM's vocabulary, and use their node degree as the weight.

3.2 Using the TE

Assuming that a user has obtained the TE for the expected output type, the TE is simply added to the [MASK] input embedding, in analogy to token and positional embeddings. Figure 2 shows an example for a prediction where we enforce a YEAR type.

Depending on the task at hand, the TE can be added to one or more tokens. We found it more effective to add it only to the [MASK] token for

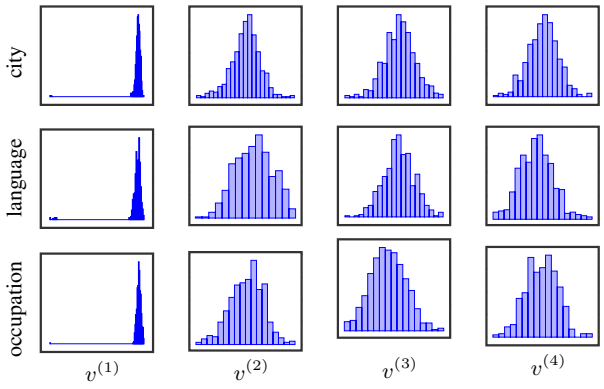

Figure 3: Distribution of the mean of the singular vectors across different types. We report the singular vectors with top-4 singular values. The distribution of the $v^{(1)}$ has the highest kurtosis.

MLM tasks, while for text generation it is more effective to add the TE to all tokens in the prompt. While we focus on MLM, we report preliminary results for text generation in Section 6.

4 Analysis of TEs

Having obtained a TE, we propose a series of analysis methods to assess its validity. We use the TE as a simple type retriever (Section 4.1), study the distribution of singular vectors (Section 4.2), analyze the effect of the TE w.r.t. the output and quantify the model's sensitivity w.r.t. typed tokens (Section 4.3), perform layer-wise classification to identify the desired type (Section 4.4), and measure TCAV of a model equipped with a TE (Section 4.5).

4.1 Similarity

As the TE is computed from token embeddings, the vector for E_t lives in the subspace formed by these embeddings. Therefore, we can use the TE to sort by distance token embeddings (through cosine similarity) as a qualitative confirmation that it reflects the desired type. Table 1 shows examples of TEs for three types (cities, years, and occupations) and the most similar token embeddings of BERT. This suggests that TEs could act as a standalone type retriever, to sort tokens according to type and analyze any biases in the tokens from which the TE is computed. Applying the method on the first singular vector $v^{(1)}$ (i.e., $-E_t$), we observe that the top retrieved tokens ('.', 'and', 'the', . . .) relate to syntax, suggesting that the first singular vector encodes syntactic aspects, in agreement with other work in multilingual representations (Roy et al., 2020), showing that such vectors encode non-semantic-related information (Yang et al., 2021).

4.2 Distribution of Singular Vectors

To understand the bias imposed by the first singular vector, we analyze the distributions of singular vectors, as it has been shown that distributions of singular vectors deviating from a Gaussian distribution contain bias (Shin et al., 2018).

From Figure 3, we see that the distribution of the singular vector $v^{(1)}$, corresponding to the largest singular value, clearly deviates from a Gaussian distribution, while others do not. This is indicated by the high kurtosis values for the first singular vectors. This suggests that this singular vector could represent a common bias that affects tokens (Shin et al., 2018). Note that since each singular vector is of dimension d, and to plot the histogram, we report the mean of the singular vector.

4.3 Effect of TE

We introduce two metrics for measuring TE's effectiveness.

Adversarial Accuracy. We expect that adding a TE to BERT causes the PLM to be more "type aware" in the associated task, i.e., adding the TE conveys type-related tokens in the output. For example, in an MLM task, adding the TE should rank higher the tokens following the associated type. In an NLG task, adding a TE should convey more type-related tokens in the generated text. We focus on the former and leave the latter for future work.

To validate this hypothesis, we check if the score of a positively typed token in an MLM task for a model with the associated TE is greater than that of a standard BERT model. Formally, given a model $\mathcal{M}_t$, with an MLM head that has been equipped with a TE E_t promoting a specific type t, we denote by $P^{(x)}_{\mathcal{M}_t}$ the normalized output score of the token x with model $\mathcal{M}_t$ and prompt pr. To assess the effectiveness of the TE, we compute this normalized probability to that of an adversary, a BERT model without any equipped TE. We define the metric *adversarial accuracy* (AA) as:

$$AA = \frac{|\{x \in X_{t^+} | P^{(x)}_{\mathcal{M}_t} > P^{(x)}_{\mathcal{M}_\emptyset}\}|}{|X_{t^+}|} \quad (1)$$

where $\mathcal{M}_\emptyset$ is a model without any TE, and X_{t^+} is a set of tokens adhering to the type t. A higher value indicates that the TE is able to promote PLM tokens following type t.

Adversarial Sensitivity. We also expect that adding the TE should make tokens following the

Type Emb.	Predictions
CITY	*Kazan*(.69), *Baku*(.67), *Cologne*(.67), Düsseldorf(.63), *Toulouse*(.62), Strasbourg(.62), *Bonn*(.61)
YEAR	*1823*(.85), 1834(.83), 1819(.82), *1755*(.82), 1825(.82), *1835*(.82), 1805(.82)
OCCUPATION	*geologist*(.76), *biologist*(.73), *theologian*(.72), *screenwriter*(.7), botanist(.69), linguist(.68), novelist(.67)

Table 1: Most similar token embeddings to a given Type Embedding with cosine similarity score in parentheses. Tokens in italic were used to compute the TE.

type more sensitive to the input TE. In other words, adding the TE in an MLM setting should cause these tokens to be more salient w.r.t. the input. To validate this hypothesis, we compare the sensitivity of a token w.r.t. the input in two models with and without a TE. If the former is greater than the latter, then the model is more sensitive to the typed token.

More formally, given a model $\mathcal{M}$, the output score of a token x is $P^{(x)}(X_{[MASK]})$[1]. With a first-order Taylor series expansion, we obtain $S^{(x)}_{\mathcal{M}} = P^t(X_{[MASK]}) - P^t(\mathbf{0}) \approx \frac{\partial P^t(X_{[MASK]})^T}{\partial X_{[MASK]}} X_{[MASK]}$, where $\mathbf{0}$ is the zero vector.

$S^{(x)}_{\mathcal{M}}$ is reminiscent of metrics used in the neural network pruning literature (LeCun et al., 1989; Molchanov et al., 2017). However, the metric is applied w.r.t. a vector rather than to the usual case of scalar, and we do not take the absolute value of the metric as we focus on comparing sensitivities of models and not measuring an absolute effect.

Finally, to test a TE, we compare the sensitivity to that of a standard BERT model. Similarly, we define *adversarial sensitivity* as the number of positive typed tokens whose sensitivity increased after adding TE to the number of positive typed tokens in a set X_{t^+}. More formally:

$$AS = \frac{|\{x \in X_{t^+} | S^{(x)}_{\mathcal{M}_t} > S^{(x)}_{\mathcal{M}_\emptyset}\}|}{|X_{t^+}|} \quad (2)$$

For both measures, we report results over a sample of 100 tokens, making sure that every one is an instance of type t and none of them has been used to derive the TE. We then compute the accuracy 10 times to get mean and standard deviation. To make sure that any change in the scores is due only to the TE, we set $pr = [MASK]$. This simple prompt neglects any contextual information that might affect PLM tokens, thus ensuring that any change is due to the TE.

Results for mean and standard deviation are reported in Table 2 for both AA and AS. For AA, TEs perform well in promoting tokens respecting a certain type. We observe a lower score for type CITY, which is likely due to (a) the large cardinality of the CITY type making it more difficult to model all required aspects of cities, and (b) coincidence of some city tokens with people names such as Morris, Salem, and Riley.

For AS, the TE has a small error margin. As we cannot expect token embeddings to capture all intricacies of a certain type, there are examples where the model fails the sensitivity test. Examples of failing tokens that did not show improvement in type sensitivity for CITY are Salvador and Blair, for LANGUAGE are Cherokee and Romani, and OCCUPATION are general and vicar.

[1]Other input tokens are omitted for brevity.

Type	AA	AS
CITY	.853 (.0166)	.82 (.014)
LANGUAGE	1 (0)	.860 (.012)
OCCUPATION	1 (0)	.893 (.018)

Table 2: Mean and standard deviation (in parentheses) of AA and AS for different types ($k = 1$).

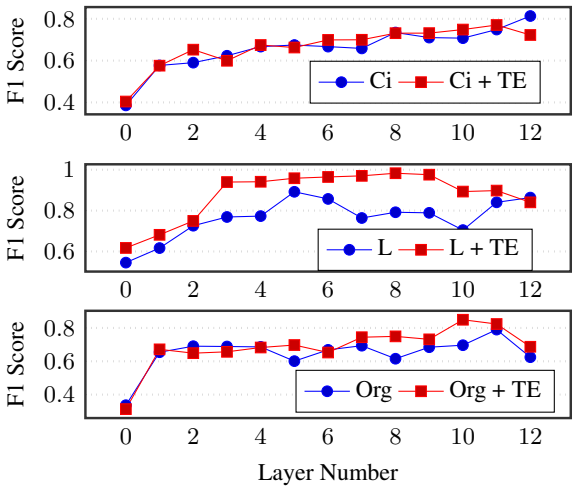

Figure 4: F1 scores of three classifiers trained and tested on layer-wise embeddings of CITY (Ci), LANGUAGE (L), and ORGANIZATION (Org) datasets.

4.4 Layer-wise Classification

As TEs are added at the input of the model, we postulate that adding TEs should help BERT identify types of input prompts more efficiently. For this, we train a layer-wise linear classifier on embed-

dings of input prompts, where positive instances are prompts belonging to a certain type t and negative instances are prompts of other types (examples in Table 3). For each type, we sample 100 positive and negative instances from other LAMA datasets (negative instances are sampled randomly from the remaining types), and train a layer-wise linear classifier. We repeat each experiment 10 times and report mean accuracy on a test set of the same size. Prompts appearing in the train set do not appear again in the test set. Results in Figure 4 show that adding TE gives most layer classifiers an increase in F1-score. The highest increase is usually at a layer in the middle, in agreement with other work (Dalvi et al., 2022), possibly because this is where a type is formed (Geva et al., 2021; Jawahar et al., 2019). The highest increase is for LANGUAGE, likely due to the smaller cardinality of the type compared to CITY and ORGANIZATION. We obtain from these the classifiers the CAVs needed for TCAV in the following section.

4.5 TCAV Sensitivity

A *Concept Activation Vector* (CAV) is a vector in the direction of the values of a concept's set of examples (Kim et al., 2018). For example, given images showing the concept of the red color (positive samples) and images without it (negative samples), a linear classifier is trained on the activation at each layer to separate positive and negative samples. The normal to the hyperplane separating the samples is the CAV. By using CAVs (with directional derivatives), one can measure the sensitivity of an input w.r.t. a concept by gauging the sensitivity of model predictions to changes in inputs towards the direction of a concept. Thus, given a set of datapoints representing a certain concept, *Testing with CAVs* (TCAVs) provides means to compute the model's conceptual sensitivity across the input (Kim et al., 2018). As a final analysis measure, we posit that a model equipped with a TE should have higher TCAV values across layers. For this, we compute layer-wise TCAV using the CAVs in Section 4.4. Figure 5 shows the TCAV values for types CITY and LANGUAGE, comparing a vanilla BERT model (k=0) and one equipped with TE (k>0) for the last 4 layers. As TCAV computes the model's conceptual sensitivity across a set of inputs, we observe that with the right TE, the importance of the type becomes more salient, i.e., the sensitivity of model predictions w.r.t. types, such as CITY at a certain layer, increases for a prompt and a TE associated with that type.

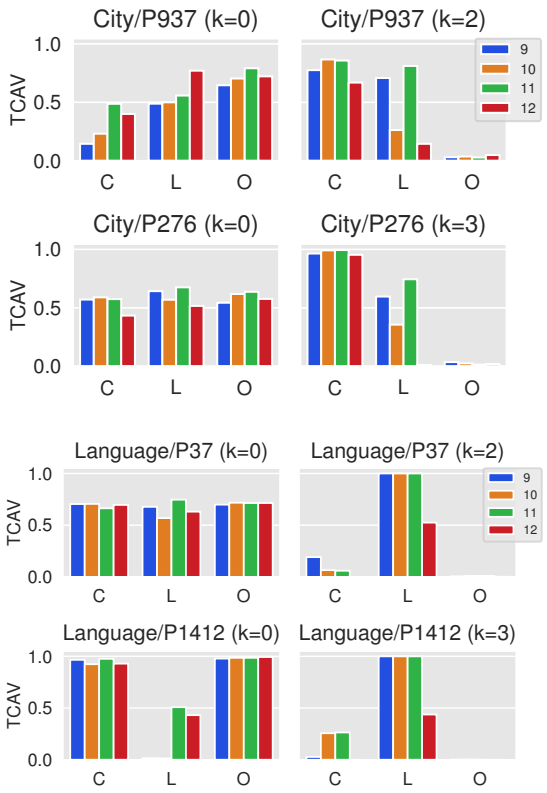

Figure 5: TCAV values for CITY (top) and LANGUAGE (bottom) datasets compared against the CITY (C), LANGUAGE (L), and ORGANIZATION (O) CAVs for layers 9-12 of BERT (left) and BERT+TE (right).

5 Experiments

The LAMA benchmark (Petroni et al., 2019) contains cloze statements to test PLMs' factual knowledge. First, we apply TEs to BERT and show increase in precision for most datasets (Section 5.1). We then enforce a change in the output with TEs (Section 5.2). Finally, we show the impact of the tokens that encode the TE (Section 5.3).

5.1 LAMA

We focus on the GRE and TREx datasets (ElSahar et al., 2018) as their prompts can be grouped into 17 output types from 38 datasets, with most examples covered by types CITY, LANGUAGE, and COUNTRY; examples for two types are in Table 3 (full list in Appendix A1). We remove prompts whose expected output is not in BERT's vocabulary and prompts containing more than one [MASK] token. This gives an upper bound on BERT's performance.

As stated in Section 3.1, the type embedding is computed with weighted sampling from KG enti-

Dataset	Prompt Example
P27	Albert II of Belgium is [MASK] citizen .
P1376	Cardiff is the capital of [MASK] .
P17	Cairo American College is located in [MASK] .
P131	Saharsa district is located in [MASK] .
P20	Fredegund died in [MASK] .
P937	Xavier Zubiri used to work in [MASK] .

Table 3: Examples of LAMA datasets grouped by output types COUNTRY (top) and CITY (bottom).

	P@1	P@10	P@50	P@100
B	.223	.509	.740	.845
BTo	.146	.327	.550	.640
PostTE	.248	.577	.819	.889
BTE (our method)	**.291**	**.606**	**.838**	**.899**

Table 4: Mean precision over all LAMA datasets compared to intrinsic baselines.

ties (10, by default). To tune the λ value of a TE, we use a hold-out dataset of 5% for each dataset, and choose the value that maximizes precision. We report results on a BERT BASE CASED model. Further experiments with other PLMs show similar trends (results in Table A2 in Appendix).

Intrinsic Evaluation. We compare BERT with TE (**BTE**) against standard BERT (**B**). As we assume that the user knows the desired output type, we also report for a baseline BERT + Token Type (**BTo**), which adds the expected type label (e.g., "the year") before the [MASK] token. We also report on a baseline **PostTE** which uses the TE at the output for re-ranking. The initial output score is added to the cosine similarity between the token embedding and the type embedding, controlled by a hyperparameter to adjust the importance of the similarity score. We choose the range of the hyperparameter to vary from 0 to 30 as in a similar work for natural language generation (Pascual et al., 2021). We also tested another baseline where we add the tokens used to derive the TE before the [MASK] token, as a signal of the desired types (Shin et al., 2020), but the results are lower than BTo.

Aggregated (macro) precision@k (P@k) results over all datasets are reported in Table 4 (full results in Table A3 in Appendix). On average, our proposal clearly improves the results. We see improvements across most of the types using TEs. However, we do observe reduction of precision in a few types, where the main reason being the greedy selection of a non-optimal value of λ. For type MANUFACTURER, setting $\lambda = 1$ (rather than $\lambda = 2$) improves

	P@1	P@10	P@50	P@100
LPAQA	.288	.607	.791	.855
BTE	**.317**	**.650**	**.868**	**.920**
OptiPrompt	**.469**	**.790**	**.922**	**.956**
BTE	.356	.697	.876	.930

Table 5: Mean precision over all LAMA datasets compared to extrinsic baselines. Unsupervised **BTE** outperforms **LPAQA**, which uses supervised learning. Supervised **OptiPrompt** obtains higher precision as it searches for prompts in the embedding space.

the results. For type SPECIALIZATION, while desired outputs such as mathematics and physics do exist in the KG samples, other nodes in the KG, such as teenager, Greek, and Sir have greater node degree and thus got selected in the sample for obtaining the TE. For the GROUP data, the value of λ for the TE was 0, meaning that adding the TE would hurt performance. Analyzing the predictions, we believe this is due to the bias in the TE imposed by the KG as most samples are related to sport groups (such as FIFA, UEFA, and CONCACAF) thus producing a TE biased towards sports group which negatively impacts the predictions. We discuss other sampling methods in Section 5.3. Finally, the YEAR dataset shows lower performance. We believe this is due to BERT's inability to precisely capture numeracy (Wallace et al., 2019). For **PostTE**, our method, of using the TE at the input, produces better results, as using the TE at the output does not allow for the fusion between factual and type knowledge in the model. **PostTE** does push typed tokens to higher rankings (indicating also the effectiveness of TEs in modeling type), but adding TEs to the input is better in terms of performance. Plus adding TEs to the input is more universal:, as the output is usually controlled by the experiment type (binary classification, MLM, NLG,...), which might not always make it clear how to insert the TE, whereas the input is always fixed. One thing to note is that, with **PostTE**, out of the 38 different datasets used, 22/38 had an optimal value of λ to be zero. Meaning that for most datasets, it did not improve results, as opposed to our method which had only 5/38 datasets with optimal $\lambda = 0$.

Extrinsic Evaluation. We evaluate our model against two supervised baselines. The first one, **LPAQA** (Jiang et al., 2020), uses mining-based methods to identify possible prompts for a given relation. The second baseline, **OptiPrompt** (Zhong et al., 2021), searches real-valued input vectors that

Prompt	TE	P@1	P@10	P@50	P@100
DoB	-	0	0	0	0
	E_{city}	.153	.404	.613	.701
	E_{city}^{optim}	.194	.444	.614	.719
PoB	-	.244	.533	.728	.808

Table 6: Precision in predicting PoB (place of birth) for DoB (date of birth) prompts by adding CITY TE ($k = 5$). Results with TE are comparable to the PoB prompt.

maximize the likelihood of the gold label on the training set using a gradient-based searching algorithm. Results in Table 5 show that our approach does better with fewer prompts, as **LPAQA** requires at least 10X more prompts per example. For **OptiPrompt**, the supervised approach produces better results than our unsupervised method. However, the approach requires training data, which is not always available. In fact, the authors of the paper use only TREx relations as they can query the KG for more data, which is not the case for Google-RE datasets. Also, as the method uses 1000 data points for training, the authors had to rely on another KG to gather more samples. Our approach requires only 10 tokens per type. Finally, while training enhances performance, it also encodes certain regularities that models could exploit, such as being prone to over-predicting the majority class label, as reported for **OptiPrompt**, unlike our approach which keeps model parameters intact.

5.2 Switching Types in Prompts

LAMA authors provide manually written prompts that adhere to the desired type. For example, to get the PLACE OF BIRTH (PoB) of a person, they use the prompt "[X] was born in [Y].", while for the DATE OF BIRTH (DoB) of a person they use the prompt "[X] (born [Y])". These prompts follow from how sentences about date and place of birth are written in Wikipedia pages. In this experiment, we ponder whether TEs can enforce a different type given one of these two prompt structure. We use DOB prompts with the expected outcomes of POB, where the goal is to steer the type of the output to a different type. For example, given "Barack (born [MASK])" (prompt for DoB), we set as expected output "Honolulu" (PoB answer). We remove examples for which the expected output is not in BERT vocabulary and are left with 1139 prompts. We then add the TE for CITY during inference. The results are shown in Table 6. As expected, without any TE, the precision score is zero as the output type is heavily influenced by the prompt. Adding E_{city} to the input steers the model to change type and it outputs cities. However, the scores are still less than those of POB prompts. Since the prompt is biased towards a certain type, better results can be obtained by removing the projection of the year information onto the city TE. Our optimized TE is then $E_{city}^{optim} = E_{city} - \frac{E_{city}.E_{year}}{||E_{city}||_2||E_{year}||_2}E_{year}$, which indeed shows improved results in Table 6.

	P@1	P@10	P@50	P@100
BTE	.291	.606	.838	.899
Top10	.336	.660	.856	.907
Bot10	.235	.534	.764	.846
Unif	.250	.563	.798	.884

Table 7: Mean over all datasets for every method.

5.3 Token Sampling

We study the impact of how tokens for TEs are sampled by (i) changing the sampling method, and (ii) varying the number of tokens used.

Sampling Methods. We evaluate forms of obtaining tokens alternative to weighted sampling: (i) weighted sampling with node degrees as weights (*BTE*), (ii) using the Top-10 tokens w.r.t. node degree (*Top10*), (iii) using the Bottom-10 tokens (*Bot10*), and (iv) sampling uniformly without relying on node degree (*Unif*). We repeat the experiment in Section 5.1 with every sampling strategy and show results in Table 7. More detailed results are in Table A4 in the Appendix.

We observe that *Top10* and weighted sampling obtain comparable performance. While *Top10* gets better results for COUNTRY, ORGANIZATION, and GENRE, other types such as YEAR, SPECIALIZATION and MANUFACTURER show lower precision because of the bias coming from the most popular KG samples. For example, *Top10* samples only years in the 21^{st} century, specializations related to titles (duke, Sultan, and Sir rather than mathematics and physics), and it is biased towards car manufacturers (Fiat and Honda). Weighted sampling reduces such bias. For FOOTBALL POSITION, *Unif* does better as it has more variety in the sample with more tokens related to American football positions (quarter back and guard) rather than soccer positions only (goalkeeper and midfielder).

In some cases, the bias in the KG reflects the bias in the test data. For OCCUPATION, the TE using *Top10* does encode some bias as most tokens are related to artistic positions (musician, actor), but

n	P@1	P@10	P@50	P@100
0	.223	.509	.740	.845
5	.279	.611	.814	.873
10	.291	.606	.838	.899
15	.275	.617	.847	.894
20	.298	.644	.859	.905
50	.292	.631	.853	.906

Table 8: Average of precision of the datasets while varying the number of samples n to compute the TE.

this improves results as the same bias occurs also among the expected outputs.

Varying Size of Samples. To study the effect of the number of tokens used in deriving the TE, we repeat the experiment in Section 5.1, while varying the number of tokens n. Results are reported in Table 8. We observe that results peak between 10 and 20 samples, but even a small number of samples significantly improves the results compared to the original BERT without TE (n=0).

6 Conclusion and Future Work

We have introduced TEs as additional input for PLMs to better encode type information, proposed methods to analyze TEs, and tested them on the LAMA dataset. While initial results are promising, we identify two directions of research.

More Precise Type Embeddings. Further analysis on the examples can lead to better TEs. One direction is to use also negative samples to compute the TE. This implies learning a vector that separates between samples as CAVs do. However, adding negative samples can bring more bias in the TE. This could be alleviated by performing some statistical hypothesis testing, as with CAVs (Kim et al., 2018). Another way to improve the effectiveness of our proposal is to combine vectors. Assuming a taxonomy of the types, different TEs can be combined, for example by subtracting for the one at hand, say PERSON, the ones that are not super or sub types, such as CITY and YEAR, as discussed for DoB in Table 6.

From Types to Concepts. While we focus on types and TEs, our approach can be extended to include more generic concepts, as long as their tokens are in the PLM's vocabulary. This could help alleviate the stereotypical and toxic content found in PLMs (Ousidhoum et al., 2021). To test our idea, we report an example for the task of natural language generation, where we "de-toxify" text generated by an autoregressive language model. We use

	λ	Toxicity (↓) Toxicity pr.	Fluency (↓) Output ppl.	Diversity (↑) Di-1	Di-2	Di-3
Toxic Prompt	0	.687	4.727	.541	.455	.357
	-1	.389	6.340	.602	.476	.377
	-2	.356	17.564	.668	.509	.400
Non-toxic Prompt	0	.045	4.195	.801	.676	.528
	-1	.077	4.038	.782	.622	.484
	-2	.088	3.716	.840	.620	.475

Table 9: Results of detoxifying texts generated from a distilled GPT-2 model. λ indicates the value of the multiplier of the TE ($\lambda = 0$ for original PLM).

a distilled GPT-2 model (Radford et al., 2019) and the *RealToxicityPrompts* dataset that contains 100K sentence-level prompts derived from a corpus of English text (Gehman et al., 2020). We feed 10K samples to the model, thus producing the generated texts. We then measure the toxicity of such texts with the *Perspective API*[2]. We consider a text toxic if the toxicity probability returned by the API is >0.5 and obtain 460 toxic prompts. We then compute a "toxicity concept embedding" using 6 manually picked tokens that convey toxicity. To de-toxify the generated text, we set the multiplier λ to negative values. Instead of adding the embedding to the [MASK] token only, we found better results when adding it to all tokens in the prompt. We believe that adding the TE to all tokens helps to 'preserve' type information along the lengthy generation procedure, as opposed to MLM which decodes one token. We also test a sample of non-toxic prompts (same size as toxic prompts) to show the effect of our embedding. In addition to toxicity, we measure *fluency* (perplexity of generated continuations according to a larger PLM) and *diversity* (mean number of distinct uni-/bi-/trigrams, normalized by the length of text for each prompt), as in other works for text generation (Liu et al., 2021).

Results in Table 9 show a huge reduction in the toxicity probability with $\lambda = -1$, higher more diversity but slightly less fluency for the toxic prompt. Setting $\lambda = -2$ decreases further the toxicity probability, but at the expense of less fluency. For the non-toxic prompts, the toxicity results are nearly the same, with minor differences for fluency and diversity. Considering that a "concept vector" steers the generation of the PLM without any form of fine-tuning, it is promising to study the use of "plug-and-play" concept vectors. Examples are reported in Table B1 in the Appendix.

[2] https://perspectiveapi.com/

Limitations

Encoding types requires a set of tokens and their embeddings. As we turn to PLMs, we are restricted by the tokens in its vocabulary, which limits the number of possible types for TEs. In addition, while we use TEs for a factual dataset, the TE encodes only type information and no factual information. While results improve for LAMA with TE, the interaction of type information and factual knowledge of the PLM is not understood. Finally, one cannot decide on a clear sampling method to use for computing the TEs (assuming the existence of a knowledge source such as a KG). The best sampling is heavily dependent on the distribution of the gold labels in the test dataset.

Ethics and Broader Impact

We are aware of (*i*) the biases and abusive language patterns (Bender et al., 2021) that PLMs impose, and (*ii*) the imperfectness and the bias of using knowledge graphs. However, our goal in this paper is to study how PLMs can be made more 'type-aware'. For (*i*), there has been some work on debiasing PLMs (Liang et al., 2020), while for (*ii*), we use a KG in our work to have variety in the set of tokens, but could resort to user-specified ones validated by consensus to reduce the bias.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In *FAccT*, page 610–623. ACM.

Zied Bouraoui, José Camacho-Collados, and Steven Schockaert. 2020. Inducing relational knowledge from bert. In *AAAI*.

Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating embeddings of heterogeneous relational datasets for data integration tasks. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, pages 1335–1349. ACM.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Wanyun Cui and Xingran Chen. 2021. Open rule induction. In *Advances in Neural Information Processing Systems*.

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*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

Fahim Dalvi, Abdul Rafae Khan, Firoj Alam, Nadir Durrani, Jia Xu, and Hassan Sajjad. 2022. Discovering latent concepts learned in BERT. In *International Conference on Learning Representations*.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. 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 S. Hare, Frédérique 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, Miyazaki, Japan, May 7-12, 2018*.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Ke Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *ArXiv*, abs/2203.14680.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *ICML*.

Yann LeCun, John Denker, and Sara Solla. 1989. Optimal brain damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann.

Nayeon Lee, Belinda Li, Sinong Wang, Wen-tau Yih, Hao Ma, and Madian Khabsa. 2020. Language models as fact checkers? In *Proceedings of the Third Workshop on Fact Extraction and VERification (FEVER)*, pages 36–41, Online. Association for Computational Linguistics.

Bai Li, Zining Zhu, Guillaume Thomas, Yang Xu, and Frank Rudzicz. 2021. How is BERT surprised? layerwise detection of linguistic anomalies. 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 4215–4228, Online. Association for Computational Linguistics.

Paul Pu Liang, Irene Mengze Li, Emily Zheng, Yao Chong Lim, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Towards Debiasing Sentence Representations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5502–5515, Online. Association for Computational Linguistics.

Yongjie Lin, Yi Chern Tan, and Robert Frank. 2019. Open sesame: Getting inside BERT's linguistic knowledge. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 241–253, Florence, Italy. Association for Computational Linguistics.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. 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 6691–6706, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2020. RoBERTa: A Robustly Optimized BERT Pretraining Approach.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual knowledge in gpt. *arXiv preprint arXiv:2202.05262*.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2017. Pruning convolutional neural networks for resource efficient inference. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can foundation models wrangle your data?

Nedjma Ousidhoum, Xinran Zhao, Tianqing Fang, Yangqiu Song, and Dit-Yan Yeung. 2021. Probing toxic content in large pre-trained language models. 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 4262–4274, Online. Association for Computational Linguistics.

Damian Pascual, Beni Egressy, Clara Meister, Ryan Cotterell, and Roger Wattenhofer. 2021. A plug-and-play method for controlled text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3973–3997, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Fabio Petroni, Patrick Lewis, Aleksandra Piktus, Tim Rocktäschel, Yuxiang Wu, Alexander H. Miller, and Sebastian Riedel. 2020. How context affects language models' factual predictions. *AKBC*.

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.

Giovanni Puccetti, Alessio Miaschi, and Felice Dell'Orletta. 2021. How do BERT embeddings organize linguistic knowledge? In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 48–57, Online. Association for Computational Linguistics.

Guanghui Qin and Jas' Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. In *NAACL*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Anna Rogers, Olga Kovaleva, and Anna Rumshisky. 2020. A Primer in BERTology: What We Know About How BERT Works. *Transactions of the Association for Computational Linguistics*, 8:842–866.

Uma Roy, Noah Constant, Rami Al-Rfou, Aditya Barua, Aaron Phillips, and Yinfei Yang. 2020. LAReQA: Language-agnostic answer retrieval from a multilingual pool. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5919–5930, Online. Association for Computational Linguistics.

Jessica Schrouff, Sebastien Baur, Shaobo Hou, Diana Mincu, Eric Loreaux, Ralph Blanes, James Wexler, Alan Karthikesalingam, and Been Kim. 2021. Best of both worlds: local and global explanations with human-understandable concepts. *ArXiv*, abs/2106.08641.

Jamin Shin, Andrea Madotto, and Pascale Fung. 2018. Interpreting word embeddings with eigenvector analysis. In *Workshop on Interpretability and Robustness in Audio, Speech, and Language (IRASL)*. NeurIPS IRASL.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. Extracting latent steering vectors from pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 63–76, 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.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do NLP models know numbers? probing numeracy in embeddings. 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 5307–5315, Hong Kong, China. Association for Computational Linguistics.

Benyou Wang, Lifeng Shang, Christina Lioma, Xin Jiang, Hao Yang, Qun Liu, and Jakob Grue Simonsen. 2021a. On position embeddings in {bert}. In *International Conference on Learning Representations*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021b. K-Adapter: Infusing Knowledge into Pre-Trained Models with Adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418, Online. Association for Computational Linguistics.

Yu-An Wang and Yun-Nung Chen. 2020. What Do Position Embeddings Learn? An Empirical Study of Pre-Trained Language Model Positional Encoding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6840–6849, Online. Association for Computational Linguistics.

Ziyi Yang, Yinfei Yang, Daniel Cer, and Eric Darve. 2021. A simple and effective method to eliminate the self language bias in multilingual representations. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5825–5832, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [mask]: Learning vs. learning to recall. In *North American Association for Computational Linguistics (NAACL)*.

A LAMA

Dataset statistics are reported in Table A1. Detailed results on the datasets are reported in Table A3. A full inference run on all LAMA datasets takes on average approximately 5 minutes on Google Colab with a Tesla P100 with a batch size of 32. We vary λ from 0 to 5. We repeat the experiment in Section 5.1 with every sampling strategy and report results in Table A4. For LANGUAGE, all sampling methods outperform the weighted method. This is due to the non-optimal value of λ produced for one dataset that reduced the average value. Indeed, setting a more suitable value for λ, pushes the precision scores comparably to other sampling methods. Surprisingly, for RELIGIOUS POSITION, *Bot10* produces better results on all metrics except $P@1$.

Type	Total Size	Dataset	Size	Sample
Country (Co)	3796	P495	896	The Sharon Cuneta Show was created in [MASK] .
		P27	948	Albert II of Belgium is [MASK] citizen .
		P1376	196	Cardiff is the capital of [MASK] .
		P1001	665	National Congress of Honduras is a legal term in [MASK] .
		P530	174	Vanuatu maintains diplomatic relations with [MASK] .
		P17	917	Cairo American College is located in [MASK] .
Football Position (FP)	737	P413	737	Curt Flood plays in [MASK] position .
Manufacturer (Ma)	878	P176	878	iPod shuffle is produced by [MASK] .
Organization (Org)	837	P108	342	David Dimbleby works for [MASK] .
		P178	495	iPod Classic is developed by [MASK] .
Occupation (Occ)	915	P106	915	Murray Grand is a [MASK] by profession .
Year GRE (Y (GRE))	1821	date_of_birth	1821	Emily Ballou (born [MASK]).
Genre (Ge)	849	P136	849	Boyd Raeburn plays [MASK] music .
Group (Gr)	212	P463	212	Russian Football Union is a member of [MASK] .
Language (L)	4118	P407	756	The Pirate Bay was written in [MASK] .
		P103	954	The native language of Jan Davidsz. de Heem is [MASK] .
		P1412	921	Leone Caetani used to communicate in [MASK] .
		P37	707	The official language of Iitti is [MASK] .
		P364	780	The original language of Do Phool is [MASK] .
Specialization (Sp)	533	P101	533	John Archibald Wheeler works in the field of [MASK] .
Religious Position (RelP)	727	P39	727	John Joseph Williams has the position of [MASK] .
Record Label (Rec)	256	P264	256	Amr Mostafa is represented by music label [MASK] .
City (Ci (GRE))	3689	place_of_birth	2925	Jacques Autreau was born in [MASK] .
		place_of_death	764	Robert Jack died in [MASK] .
City (Ci)	6490	P131	774	Saharsa district is located in [MASK] .
		P20	844	Fredegund died in [MASK] .
		P937	864	Xavier Zubiri used to work in [MASK] .
		P19	704	James Jackson Putnam was born in [MASK] .
		P740	643	Standard Bank was founded in [MASK] .
		P190	283	Inverness and [MASK] are twin cities .
		P36	400	The capital of Realm of Stefan Dragutin is [MASK] .
		P159	700	The headquarter of Shelbourne F.C. is in [MASK] .
		P47	542	Campi Bisenzio shares border with [MASK] .
		P276	736	Hiroshima International Animation Festival is located in [MASK] .
Continent (Con)	964	P30	964	Dominion Range is located in [MASK] .
Musical Instrument MI	739	P1303	739	Kerry King plays [MASK] .
TV Network (TVN)	806	P449	806	The New Dick Van Dyke Show was originally aired on [MASK] .
Religion (Rel)	452	P140	452	Muhammad Ali Jinnah is affiliated with the [MASK] religion .

Table A1: LAMA datasets grouped by type. Each dataset belongs to the TREx dataset, unless otherwise stated by (GRE).

This is because most of the golden labels of the data related to religious positions for Christianity, while using *Top10* includes a position for Judaism (rabbi), which is not the case for *Bot10* and *Unif*. Finally, similar results are observed for GROUP and CONTINENT simply because there were less than 10 tokens for each type from the KG.

	P@1	**P@10**	**P@50**	**P@100**
Bl	0.245	0.523	0.729	0.811
BlTE	0.297	0.582	0.777	0.849
Rob	0.073	0.235	0.400	0.479
RobTE	0.177	0.331	0.481	0.589

Table A2: Mean over all datasets for Bert Large (Bl) and Roberta base (Rob) with and without TEs..

B Generated Text

Examples of text generated with TEs are reported in Table B1.

		P@1	P@10	P@50	P@100
Co	B	0.333	0.578	0.812	0.892
	BTo	0.092	0.269	0.427	0.520
	PostTE	0.323	0.549	0.838	0.888
	BTE	0.393	0.643	0.874	0.916
FP	B	0.003	0.234	0.500	0.701
	BTo	0.203	0.407	0.657	0.730
	PostTE	0.239	0.510	0.883	0.977
	BTE	0.276	0.500	0.826	0.896
Ma	B	0.865	0.945	0.982	0.988
	BTo	0.859	0.939	0.98	0.987
	PostTE	0.008	0.848	0.941	0.965
	BTE	0.564	0.923	0.970	0.978
Org	B	0.347	0.733	0.928	0.961
	BTo	0.248	0.393	0.447	0.464
	PostTE	0.347	0.733	0.928	0.961
	BTE	0.279	0.730	0.955	0.977
Occ	B	0.002	0.089	0.463	0.839
	BTo	0.0	0.023	0.305	0.441
	PostTE	0.012	0.188	0.619	0.951
	BTE	0.036	0.196	0.601	0.904
Y (GRE)	B	0.016	0.152	0.623	0.806
	BTo	0.002	0.102	0.499	0.783
	PostTE	0.016	0.152	0.623	0.806
	BTE	0.017	0.146	0.624	0.802
Ge	B	0.007	0.470	0.697	0.803
	BTo	0.0	0.087	0.636	0.743
	PostTE	0.594	0.690	0.834	0.844
	BTE	0.589	0.686	0.831	0.841
Gr	B	0.692	0.821	0.861	0.886
	BTo	0.025	0.214	0.652	0.801
	PostTE	0.692	0.821	0.861	0.886
	BTE	0.692	0.821	0.861	0.886
L	B	0.600	0.892	0.971	0.988
	BTo	0.168	0.436	0.622	0.716
	PostTE	0.445	0.750	0.965	0.982
	BTE	0.556	0.855	0.967	0.980
Sp	B	0.085	0.362	0.569	0.688
	BTo	0.0	0.008	0.138	0.291
	PostTE	0.085	0.362	0.569	0.688
	BTE	0.097	0.302	0.545	0.640
RelP	B	0.070	0.281	0.709	0.900
	BTo	0.0	0.023	0.219	0.372
	PostTE	0.010	0.503	0.938	0.962
	BTE	0.159	0.488	0.951	0.959
Rec	B	0.140	0.416	0.733	0.877
	BTo	0.062	0.342	0.539	0.642
	PostTE	0.095	0.218	0.539	0.621
	BTE	0.152	0.444	0.819	0.922
Ci (GRE)	B	0.142	0.353	0.566	0.657
	BTo	0.0	0.003	0.024	0.06
	PostTE	0.142	0.353	0.566	0.657
	BTE	0.144	0.365	0.572	0.663
Ci	B	0.287	0.570	0.757	0.831
	BTo	0.058	0.107	0.194	0.252
	PostTE	0.284	0.563	0.756	0.839
	BTE	0.299	0.577	0.768	0.849
Con	B	0.216	0.481	0.730	0.822
	BTo	0.613	0.864	0.969	0.995
	PostTE	0.477	0.885	0.963	0.995
	BTE	0.408	0.882	0.945	0.980
MI	B	0.064	0.390	0.553	0.614
	BTo	0.131	0.64	0.821	0.832
	PostTE	0.017	0.587	1.000	1.000
	BTE	0.017	0.593	0.991	1.000
TVN	B	0.210	0.858	0.986	0.997
	BTo	0.161	0.609	0.952	0.992
	PostTE	0.210	0.858	0.986	0.997
	BTE	0.200	0.877	0.993	0.997
Rel	B	0.107	0.536	0.883	0.967
	BTo	0.002	0.422	0.814	0.900
	PostTE	0.476	0.809	0.925	0.981
	BTE	0.352	0.876	0.998	1.000

Table A3: Average precision scores for different types of the LAMA dataset for BERT (B), BERT with additional typed tokens (BTo), TE applied at the output (PostTE), and BERT with TE (BTE).

		P@1	P@10	P@50	P@100
Co	Top10	0.407	0.708	0.891	0.930
	Bot10	0.326	0.608	0.836	0.903
	Unif	0.355	0.601	0.83	0.904
FP	Top10	0.277	0.564	0.861	0.91
	Bot10	0.0	0.04	0.684	0.746
	Unif	0.223	0.561	0.859	0.911
Ma	Top10	0.770	0.921	0.974	0.984
	Bot10	0.865	0.945	0.982	0.988
	Unif	0.865	0.945	0.982	0.988
Org	Top10	0.606	0.866	0.966	0.979
	Bot10	0.307	0.665	0.906	0.951
	Unif	0.275	0.588	0.897	0.951
Occ	Top10	0.087	0.547	0.849	0.921
	Bot10	0.002	0.089	0.48	0.845
	Unif	0.001	0.089	0.496	0.872
Y (GRE)	Top10	0.019	0.145	0.618	0.792
	Bot10	0.010	0.109	0.377	0.578
	Unif	0.018	0.147	0.625	0.803
Ge	Top10	0.582	0.703	0.84	0.842
	Bot10	0.006	0.416	0.703	0.806
	Unif	0.043	0.057	0.356	0.636
Gr	Top10	0.692	0.821	0.861	0.886
	Bot10	0.692	0.821	0.861	0.886
	Unif	0.692	0.821	0.861	0.886
L	Top10	0.61	0.905	0.981	0.992
	Bot10	0.612	0.894	0.975	0.989
	Unif	0.603	0.895	0.976	0.993
Sp	Top10	0.085	0.354	0.573	0.682
	Bot10	0.081	0.356	0.577	0.688
	Unif	0.087	0.391	0.581	0.686
RelP	Top10	0.300	0.600	0.955	0.959
	Bot10	0.0	0.651	0.961	0.965
	Unif	0.0	0.506	0.954	0.962
Rec	Top10	0.193	0.469	0.885	0.942
	Bot10	0.144	0.428	0.761	0.881
	Unif	0.148	0.436	0.761	0.881
Ci (GRE)	Top10	0.148	0.37	0.584	0.678
	Bot10	0.098	0.320	0.521	0.613
	Unif	0.084	0.309	0.525	0.628
Ci	Top10	0.306	0.580	0.773	0.851
	Bot10	0.280	0.548	0.734	0.805
	Unif	0.292	0.578	0.762	0.841
Con	Top10	0.408	0.882	0.945	0.980
	Bot10	0.408	0.882	0.945	0.980
	Unif	0.408	0.882	0.945	0.980
MI	Top10	0.074	0.699	0.849	0.996
	Bot10	0.064	0.39	0.553	0.614
	Unif	0.017	0.692	0.996	1.000
TVN	Top10	0.086	0.864	0.997	0.997
	Bot10	0.225	0.864	0.984	0.997
	Unif	0.299	0.897	0.986	0.997
Rel	Top10	0.396	0.879	0.998	1.000
	Bot10	0.119	0.590	0.914	0.988
	Unif	0.089	0.734	0.972	0.988

Table A4: Average precision scores for different types of the LAMA dataset with various sampling methods to compute the TE.

	Generation
GPT-2	I like a lot of the things that your parents did at the age of 9 that made you happy, and now you're in my life."The former wrestler said he is glad he didn't pass that test. "They thought I was a bad idea, as far as it goes in life," Mayweather said of his parents and co-stars. "But that went way overboard for me. I went from having a bad childhood and now living in the Bronx with my aunt and her little sister to a guy who's trying to put it all in the right place."That brought me back to the top of my wrestling career and I think the most special person is David Letterman, who gave me that opportunity...
GPT2 + COUNTRY TE	I like a lot of the things that your guys are doing in the dressing room that the team does, and that's what we do. Q: You have said in a couple of different emails that you wanted to go back to **Switzerland**, but that's a different subject as well. A: Well, you will return. I have spoken to the players and said, that way, to go back to **France** and see the success. Q: What's the biggest point of contention from last week? A: I was on the bench in the first half of the game against **France**, although today I was on the bench. But yesterday I was in the team too. It was a big point of contention. And then tomorrow I will go to **Switzerland** . I had a talk with the coaches this morning and they said to him there were plenty people in the dressing room. Q: Will the **U.S.** play at the **Emirates** ? A: That's a possibility for sure...
GPT-2 + YEAR TE	I like a lot of the things that she is doing at the moment, but that is for the time being." She also said she has a lot of work ahead in mind when he arrives on an episode of "The Simpsons." For a fuller picture of her as a show host and actor see "What Happened to the Kids on the Island of the Gummy"? A documentary record (**1990**), also by David R. Seidman. Also on H&M Records. The last time she appeared on a TV show was on "Saturday Night Live" in **2007** , and in "My Brother's Wedding" on ABC in August **2007** . The first time she appeared on "The Tonight Show, with Johnny Carson" was in **1996** . She also won the Tony Award with "The Little Princess" from the Writers Guild and the Outstanding Dramas Award in **2002** . When interviewed on the TV show she recalled that she has worked as the 'Gummy' and 'Grandma' of the show, and gave the following list of roles for "My Brother's Wedding, with Johnny Carson" of her own first year. The '**90s** came in the last season of "How I Met Your Mother" which she co-starred on...
GPT-2	I hate *RelG*, yet *RelG* are the ones who are allowed to carry out this murderous act by terrorising *RelG* in *Religious Country* and then going on to kill innocent *RelG* in the future."...
GPT-2 + AFFECTION **TE**	I hate *RelG*, yet *RelG* are the ones who are allowed to carry the torch of *Religion* and fight injustice and oppression. You and I will continue to fight this oppression till the end and in the long run, in every age we are going forward. You will do whatever is necessary to keep our sisters on the right path...

Table B1: We report a set of truncated generated texts using GPT-2 and how they change using TEs for COUNTRY and YEAR. We also try to remove hate speech using an AFFECTION embedding derived from keywords such as love and cheerful. We replace specific keywords by more general keywords. *RelG* stands for a religious group that has been used in the underlined prompt and has been hidden for ethical considerations.