On the Representations of Entities in Auto-regressive Large Language Models

Victor Morand, Josiane Mothe, Benjamin Piwowarski

On the Representations of Entities in Auto-regressive Large Language Models

Victor Morand[1] **Josiane Mothe**[2] **Benjamin Piwowarski**[1]

[1]Sorbonne Université, CNRS,
Institut des Systèmes Intelligents et de Robotique (ISIR),
F-75005 Paris, France

[2]INSPE, UT2J, Université de Toulouse, IRIT
UMR5505 CNRS, F-31400 Toulouse, France

Abstract

Named entities are fundamental building blocks of knowledge in text, grounding factual information and structuring relationships within language. Despite their importance, it remains unclear how Large Language Models (LLMs) internally represent entities. Prior research has primarily examined explicit relationships, but little is known about entity representations themselves. We introduce entity mention reconstruction as a novel framework for studying how LLMs encode and manipulate entities. We investigate whether entity mentions can be generated from internal representations, how multi-token entities are encoded beyond last-token embeddings, and whether these representations capture relational knowledge. Our proposed method, leveraging *task vectors*, allows to consistently generate multi-token mentions from various entity representations derived from the LLMs hidden states. We thus introduce the *Entity Lens*, extending the *logit-lens* to predict multi-token mentions. Our results bring new evidence that LLMs develop entity-specific mechanisms to represent and manipulate any multi-token entities, including those unseen during training.[1]

1 Introduction

Despite the remarkable achievements of LLMs across a range of natural language processing tasks, the mechanisms underlying their ability to represent and manipulate knowledge remain opaque, making it challenging to enhance their interpretability and control. Among the various components of knowledge expression, entities are fundamental building blocks. They serve as anchors for factual information and relationships within text. A key challenge is therefore to understand how LLMs internally represent named entities (*e.g.* `Eiffel tower`) in a given context and how these representations persist, evolve, and encode information across model layers. Understanding how entity knowledge is stored and retrieved may enable us to better monitor when and why LLMs hallucinate or generate factually inconsistent responses.

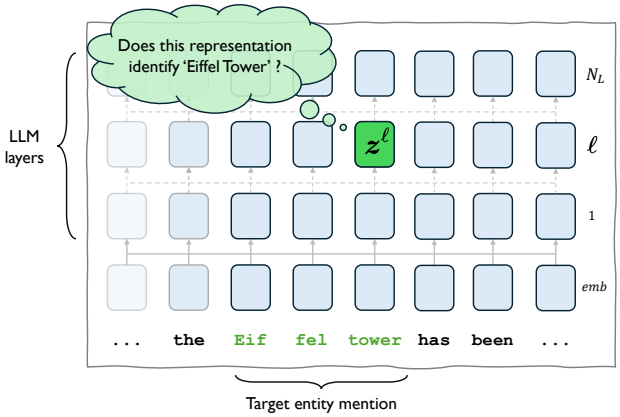


Figure 1: Representation extraction principle: the representation z^ℓ (in green) of the last token of a target entity mention - here *tower* for the named entity *Eiffel tower* - is extracted at a given layer ℓ of the transformer model.

Current interpretability research has primarily focused on entity relationships and factual knowledge rather than the internal representation of entities themselves (Geva et al., 2021, 2023; Hernandez et al., 2024; Niu et al., 2023). Many named entities span multiple tokens, yet prior work has largely used single-token probing. Our work seeks to fill this gap by exploring how LLMs build unified entity representations from multiple token.

In this work, we posit that LLMs compute layer-agnostic entity representations that can be isolated and manipulated. Our primary objective is to establish a direct association between these internal representations and named entities, focusing specifically on their mention in text. *To evaluate this association, we measure how accurately the corresponding mention can be generated from the representation at hand.* The core of our methodology is

[1]Code is available here, including all the hyper-parameters used in the experiments.

presented in Section 3.1, addressing the following research question:

RQ 1. *How well entity mentions can be decoded from their representation?*

Our results show that LLMs possess specific mechanisms for representing and manipulating entities, allowing them to consistently generate mentions from their internal entity representations. We also find that decoding capacity depends more on an entity frequency in the training data than on its token length.

Our next research question challenges the current assumption of using the last token's representation at a given layer. We address it in Section 4.

RQ 2. *Can we find better representations than the last token representations from LLMs?*

As alternatives, we propose averaging the mention token representations and cleaning the entity representation, and show those help the model to decode entity mentions.

Our last research question relates to the additional knowledge entity representations capture; it is developed in Section 5:

RQ 3. *Does the structure of the entity representation space capture (relational) knowledge?*

By transforming the representation of subject to related object entities with a linear transformation, we show that we can extract more than the sole entity mention from those representations.

We also introduce a practical application of our method called the *Entity Lens*. It allows to visualize which entity the model is "thinking" about at a given layer, extending *logit lens* (nostalgebraist, 2020) to generate multi-token entity mentions, instead of projecting a token's hidden state onto the vocabulary.

2 Related Work

The question of knowledge representation has been studied very early in the development of neural network approaches for language modeling. Word2Vec (Mikolov et al., 2013) shows that relationships between words (e.g., singular to plural, capital to country) could be represented as translations between word representations. This motivated the development of knowledge base representations where entities were linked by geometrical transformation – e.g., TransE (Bordes et al., 2013) that explicitly uses the translation observed in Word2Vec.

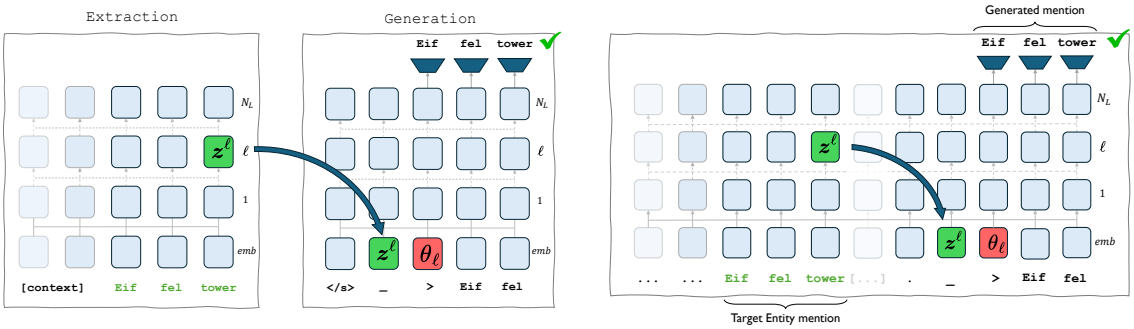
With the development of contextual word – and then token – embeddings, such as ELMo (Peters et al., 2019) and BERT (Devlin et al., 2019), the question of how knowledge is encoded and how entities are represented was temporarily put aside.

Then, early work (Petroni et al., 2019) showed that pre-trained LLMs can act as "knowledge bases" by retrieving factual information through prompt-based queries. To understand how this retrieval occurs in LLMs, Geva et al. (2021); Meng et al. (2022) hypothesized that feed-forward layers act as key-value memories, storing and retrieving factual associations during inference. Even if this idea has been challenged (Niu et al., 2023), Hernandez et al. (2024) showed that, within LLMs, simple relations can often be approximated by a linear mapping of the subject to the object entity representation at a middle decoder layer – echoing (Mikolov et al., 2013) in the context of LLMs. This suggests that LLMs operate, at least partially, within a structured entity representation space where entities can be manipulated. Recent research employing sparse auto-encoders supports this hypothesis, with Ferrando et al. (2025) identifying a feature within this entity representation space that quantifies how much the model "knows" about an entity.

This raises the question of how entities themselves are represented. Research suggests that earlier layers capture surface-level details, while deeper layers encode more abstract and task-specific features (Jawahar et al., 2019; Voita et al., 2019; Geva et al., 2023). While entities are processed across all layers, their representations—especially for multi-token entities—may not always be coherent or robust.

While probing methods are used to determine whether specific types of information (e.g., syntactic structure or factual content) can be extracted from the model's representations (Conneau et al., 2018; Tenney et al., 2019), these probes are not fine-grained and do not allow to understand how knowledge is processed and represented. To gain some insight on how LLM processes knowledge, a specific type of probe, called 'logit attribution' or 'logit lens' has been developed (nostalgebraist, 2020; Yu et al., 2023; Dalvi et al., 2019). With logit lens, the LLM hidden states are projected onto the vocabulary using the unembedding matrix, allowing to identify which tokens would be predicted at a given layer of the LLM.

Representing multi-token entities, like '`Eif|fel|_tower`', presents unique challenges



(a) Uncontextual entity mention decoding: The entity representation z^ℓ at layer ℓ is extracted in context (left, green), its mention is then generated using a learned *task vector* θ_ℓ that prompts the model to generate the mention from z^ℓ only.

(b) Contextual entity mention decoding: Also using a learned *task vector* θ_ℓ, the entity mention is now generated using both the extracted representation z^ℓ and the surrounding context from which it can be copied.

Figure 2: Our entity mention decoding method, in both uncontextual (left) and contextual (right) senarii.

for LLMs. Their internal representation may fail to capture the entity's full meaning consistently. Using the logit lens can help, but is insufficient to properly associate representations with the entire multi-token mention. This issue is further highlighted by (Liu, 2021), who found that tokenization granularity can significantly affect the quality of representations for multi-word expressions. *In this contribution, we extend logit lens by proposing the Entity lens that decodes full entity mentions from internal representations.* To address these challenges, researchers have explored strategies like attention-based aggregation (Clark et al., 2019), contextual subword pooling (Schick and Schütze, 2021), and token-level masking (Joshi et al., 2020). More recent methods such as Patchscopes (Ghandeharioun et al., 2024) and SelfIE (Chen et al., 2024) extends the limitations of the logit lens to produce multi-token explanations from an extracted representation. In contrast with generating general explanations, our work focuses on entity mention, allowing the use of quantitative metrics such as exact match. Our work specifically studies the problem of how to represent and manipulate multi-token entities.

3 Retrieving entity mentions from LLM representations

3.1 Methodology

In transformer-based language models (Vaswani et al., 2017), text is tokenized into a sequence of tokens $(t_1, \ldots, t_n) \in \mathcal{V}^n$, with $\mathcal{V}$ the vocabulary used by the tokenizer. These tokens are embedded into a sequence of initial *representations* $(z_1^0, \ldots, z_n^0) \in \mathbb{R}^d$, with d the model's representation space dimension. These representations are sequentially passed through the transformer layers: each layer $\ell \in \{1, \ldots, N_L\}$ generates a new series of representations $(z_1^\ell, \ldots, z_n^\ell) \in \mathbb{R}^d$, building on the representations from the preceding layer.

Decoding In this contribution, we aim at associating a given representation $z \in \mathbb{R}^d$ at hand with a named entity by trying to generate the mention from which the representation was extracted. Following the literature (Meng et al., 2022; Geva et al., 2023), z^ℓ is extracted from the last token of the entity mention at layer ℓ (See Figure 1). Section 4 extends our experiments to other representations. To generate a mention for the representation z^ℓ, we insert it bypassing the embedding layer and *prompt* the model to reconstruct the corresponding entity mention. This is done using a *soft prompt* or *embedding vector*, $\theta_\ell \in \mathbb{R}^d$, optimized for the task. This method is known as *Prompt Tuning* (Lester et al., 2021). Because this vector is functional rather than semantic, we refer to θ_ℓ as a *task vector*, inspired by Hendel et al. (2023).

Motivation of using Task Vectors Prompt tuning is a parameter-efficient method to prompt the model to perform a variety of tasks (Lester et al., 2021). It is however not the only method that can be used to generate text. Since our focus is on interpretability while keeping the LLM unchanged, fine-tuning is excluded. Probes have also been explored in this context (see for instance Pal et al. (2023)), but they do not easily allow to decode an arbitrary-lenght sequences of tokens, nor to prompt the model to retrieve a mention from a context. To the best of our knowledge, our task vector approach is the only method that can address all these

constraints and challenges of entity reconstruction. Our results furthermore show that this method performs well for the considered task.

Uncontextualized Decoding In this setting, the model's only input is the representation $\boldsymbol{z}^\ell$ along with the task vector θ_ℓ. The goal is to generate the entire entity mention. This allows us to measure how much information about the entity mention is retained in $\boldsymbol{z}^\ell$. This setting is illustrated Figure 2a.

Contextualized Decoding. In contextualized decoding, we include the textual context from which the representation was extracted before prompting the transformer with the task vector θ_ℓ. Figure 2b provides an overview of this methodology. In this setting, the model can just copy the mention from the context. This is however not a trivial task as the model must first identify the correct span within the context, *i.e.* essentially performing Named Entity Recognition (NER).

Evaluation In both settings, our setup allows to generate a complete mention from a given entity representation $\boldsymbol{z} \in \mathbb{R}^d$. We can then evaluate the reconstruction performance using the exact match metric (EM) between generated mentions and the original mentions from which the representations were extracted. Entity mentions are however known to be ambiguous, and a more general problem would consider all possible mentions for a given entity (*e.g.* `NYC` or `New York City`). The metric used in our uncontextual setup only measures the ability to reconstruct the specific mention from which the representation was extracted in the context, without accounting for possible additional ambiguous mentions. *Although well posed, the results are therefore only a lower bound for the more general problem*. For instance, `New York City` could be considered as an accurate generation from the representation of the mention `NYC`; our evaluation metric, however, considers it a failure. This limitation does not apply in the contextual setup, where the model is instructed to copy the mention directly from the provided context.

Task Vector Training In both settings, for each layer ℓ of the model, the task vector θ_ℓ is learned by maximizing the log-likelihood of generating the entity mention, given both the representation and θ_ℓ. We use cross-entropy, the standard loss for language modeling tasks. More implementation details and code can be found in the reproducibility statement, in Section A. Training a task vector for each layer allows us to evaluate which layers provide the most accurate entity representations.

3.2 Dataset and Experiment Setup

Data Our experiment involves extracting the representation of an entity mention within its context, and then trying to reconstruct the entity mention from this representation. This requires a dataset containing sentences with labeled spans of entity mentions: in other words, a NER dataset, for which we ignore the entity categories. We use the CoNLL-2003 dataset (Sang and De Meulder, 2003), a widely used NER benchmark. It provides a diverse set of named entities across various types, lengths and frequencies, making it well-suited for studying entity representations. In the CoNLL-2003 dataset and using the Pythia tokenizer, most entity mentions are tokenized into 2 or 3 tokens, while some span up to 8-12 tokens (see Figure 3). This diversity in token length allows us to explore the robustness of the models in handling both short and long multi-token entities, a key aspect of this study. More details on the dataset statistics can be found in Section A.

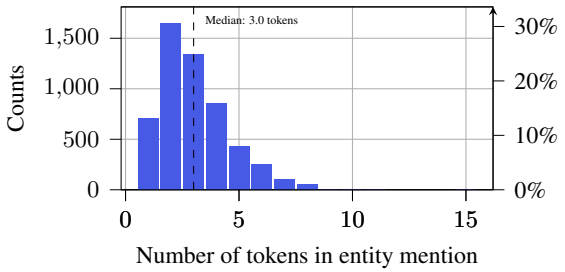


Figure 3: Distribution of the number of tokens for named entity mentions in the test set (PYTHIA tokenizer). 87% of them are tokenized with two or more tokens. The dashed line is the median token count (3).

Models We focus on decoder-only architectures, used in the vast majority of recent models, and known for their superior performance. We experiment with different models from two families. First, the PYTHIA family (Biderman et al., 2023), which includes several models of various sizes, trained with the same data and close settings. This allows the study of the impact of architecture parameters on performance, especially the model size. We also use the recent PHI family, a set of models trained with the latest techniques and curated data (Li et al., 2023). We use the available non-instruct versions, namely PHI-1.5 (Li et al., 2023) and PHI-2 (Javaheripi and Bubeck, 2023), allegedly trained on identical text data, with the particularity that

knowledge from PHI-1.5 has been distilled into PHI-2. We also use PHI-3[2] (Abdin et al., 2024), the next iteration of the PHI family which follows the LLAMA-2 architecture, and has the particularity of having a significantly smaller vocabulary size (32k tokens compared to 50k tokens for all other models considered in this work) and of being instruction fine-tuned – which is the only version available. We utilize models ranging from 140m to 7B parameters, which offers a fair range for studying the impact of model size, while limiting resource consumption. The architecture parameters of all the models we experimented with are detailed in appendix (Table 4).

3.3 Results

The results in both settings are presented in Figure 4a (uncontextual decoding) and Figure 4b (contextual). Without access to any context, some models (PYTHIA-6.9B, PHI-2 and PHI-3) are able to decode *exactly* the whole mention of up to 65% of the named entities from the test set. Upon analysis of the results, we observe that failed samples typically exhibit high semantic similarity with the original mention (see Table 3 in appendix for examples of failures). Model size unsurprisingly improves performance, supporting a well-known property that larger models memorize more knowledge, including named entities.[3] Figure 4a also shows that representations from the middle layers achieve better performance in mention decoding, corroborating findings reported in (Meng et al., 2022, 2023).

We unsurprisingly achieve significantly better generation results in the contextual setting where the model can copy the mention from the given context (Figure 4b). Near-optimal performance is reached: 93% EM when generating entity mentions using representations of their last tokens with PHI-2 (layer 20) – it was only 64% in the uncontextual setup. These results reveal a key new insight: LLMs do not, in general, store the entire entity mention within the representation of its last token. This challenges the common assumption that the final token's representation encapsulates the full entity meaning (Meng et al., 2022; Geva et al., 2023), and suggests that alternative encoding approaches may be needed to better capture multi-token entity representations.

Unlike sentence embedding models – which have been shown to encode almost all tokens from the input sentence (Morris et al., 2023) – autoregressive language models are not trained to encode tokens explicitly in their internal representations. Instead, if an entity is already mentioned in the context, the model can retrieve it thanks to the attention mechanism and simply copy it, as shown in various mechanistic interpretability studies (Wang et al., 2022; McDougall et al., 2023).

Multi-token entities and frequency Decoding the one-token entity `France` from its embedding or representation at any layer z^ℓ is much easier than decoding `Mand-el-bro-t` from the sole representation of token `'t'`. To further analyze how the number of tokens affects decoding performance, we split the test set based on tokenized mention length. Then, following the intuition that entity mentions frequent in the LLM training set should be easier to generate than rare ones, we also split the test set into quantiles based on mention n-gram frequency in *the Pile* (Gao et al., 2020).[4] We evaluate both settings across all models. In both cases, entity frequency – and not the number of tokens – is the main factor for accurate entity mention decoding (though the two are naturally correlated, see Figure 5 for results on the three PYTHIA models in the uncontextual setting).

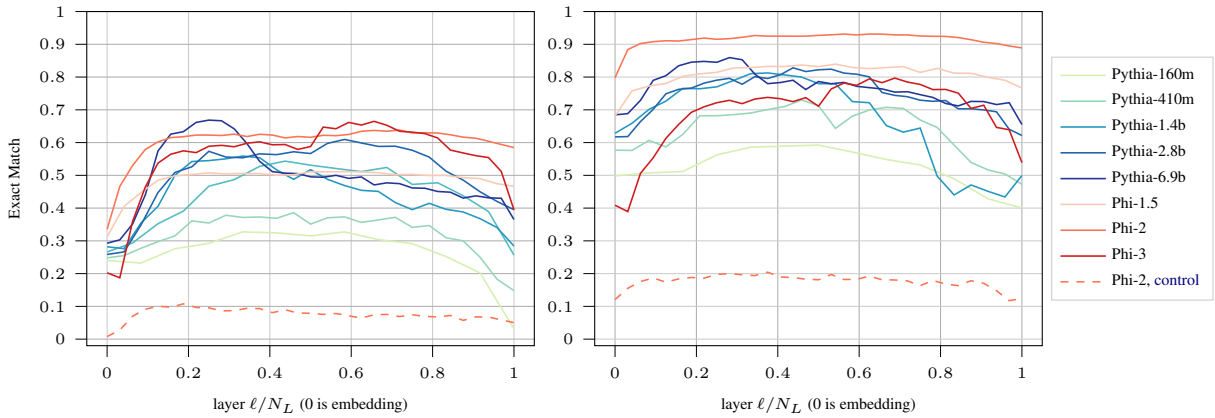
Baseline - Decoding any sequence We setup a control experiment to confirm that the unveiled mention decoding ability is specific to entities rather than a general capability of LLMs to decode prior tokens. Using the same methodology as in Section 3, we replace all entity mentions in our original dataset with randomly sampled sequences of three[5] tokens from the data, constraining the last token to be the end of a word for fairer comparison. We then train new task vectors for each model layer, so they instruct the LLM to *decode the 3-token sequence from the final-token's representation*. The results are presented in both settings in Figure 4, (control) restricted to the best-performing model (PHI-2) for clarity. In the uncontextual setup, this baseline reaches only 9% Exact Match (EM), compared to 65% when decoding entity mentions, strongly supporting our claim that LLMs detect and represent entities in a specific

[2] more specifically `Phi-3-mini-4k-instruct`.

[3] Best performance as a function of model size is reported in appendix, Figure 11.

[4] n-gram frequencies are obtained using the API from (Liu et al., 2024).

[5] Three is the median entity length in our data, see Figure 3. We also show in Section B.1 the generation results for arbitrary sequences of one and two tokens.



(a) Uncontextual mention decoding results by layer.

(b) Contextual mention decoding results by layer.

Figure 4: Context improves decoding. Better performances are obtained on representations extracted in middle layers. Curves present the rate of exact match (on y-axis) after training a task vector on representations extracted at layer ℓ (x-axis is normalized layer ℓ/N_L, as model have a different number of layers N_L).

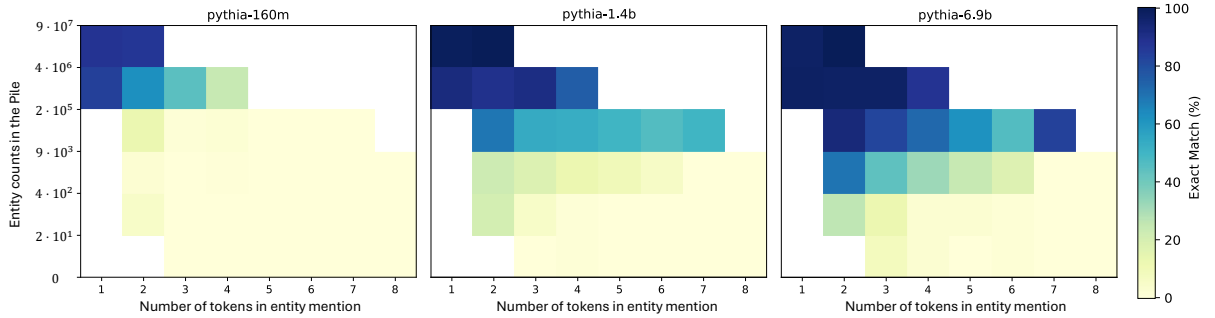


Figure 5: **Uncontextual** mention generation performance is higher for more frequent entities. Performance is analyzed by entity length and mention frequency in *the Pile* (Gao et al., 2020). For each model, we chose the layer with best exact match on the test set. Empty cells indicate fewer than five samples. See Section D.2 for full results across models and settings.

manner. In the contextual setup, the task seems trivial: The model must only find the extracted token in the context and copy it along with the two preceding tokens. However, results are surprisingly poor, with a maximum 19% EM (layer 9 of Phi-2, compared to 93% EM with 3-token entity mentions in the original contextual experiment). This further supports the idea that, unlike counting tokens, manipulating entity mentions –and potentially other meaningful units – is a natural task for which LLMs develop specialized circuits.

Conclusion Overall, our results in Section 3 bring strong evidence that LLMs develop specific mechanisms for representing and manipulating entities. In both settings, with only a single learned task vector, we successfully prompt the model to generate the correct entity mention from its representation. For named entities that are frequent in the LLM training set, the last token representation at middle layers is enough to retrieve their whole mention, just as if the latter was part of the vocabulary (Figure 5). For less common entities, LLMs however do not store the whole mention in its last token representation (even if they could, see the supplementary experiment in Section B.4). Still, when provided with the context, LLMs can very robustly detect and copy them, achieving near optimal performance in our setup (see Figure 4b). These results also demonstrate that representations are layer-agnostic since the LLM is still able to process them when injected at the embedding level, using only a simple task vector. Additional experiments investigating how those representations are built are detailed in Section B.3. We also confirm that task vectors generalize across different settings and layers (see Section B.2), further supporting our claim that they activate to a given extent the same specific circuits within LLMs.

4 Obtaining better representations

In this section, we question the optimality of using the last token representation of an entity. We

extend our initial experiments by using alternative representations for named entity mentions. For experiments, we chose to focus on PHI-2, because it is newer compared to PYTHIA, has a reasonable number of parameters allowing fast inference and got the best results in the first set of experiments.

Average representations The most common way to extract representations of sentences with language models is to *average* the representations of all their tokens (Jurafsky and Martin, 2009). As superpositions are the building blocks of LLMs (Elhage et al., 2022), this seems a natural choice. Formally, for a given named entity mention $e = (t_{e_1}, ..., t_{e_2})$, we compute its average representation at layer ℓ over the mention tokens, $\bar{\boldsymbol{z}}^\ell$ as $\bar{\boldsymbol{z}}^\ell = \sum_{i=e_1}^{e_2} \frac{\boldsymbol{z}_i^\ell}{e_2 - e_1 + 1}$.

Training a linear layer to *clean* the entity Representations are highly dependent on the context (Ethayarajh, 2019). In other words, for a given entity mention, the representations extracted from the inner layers of transformer models are very likely to contain, along with the representation of the entity, a lot of noise that comes from the context. Passing the extracted representation $\boldsymbol{z}$ into a linear layer that would clean or reinforce some relevant features might produce entity representations of better quality, in the sense that the model could better decode the entity mention from them. For both uncontextual and contextual settings, we therefore train a linear model with parameters (W, b) that is applied to the extracted representation $\boldsymbol{z}$ to obtain a *cleaned* representation $\mathcal{C}_{\boldsymbol{z}} = W\boldsymbol{z} + b \in \mathbb{R}^d$.

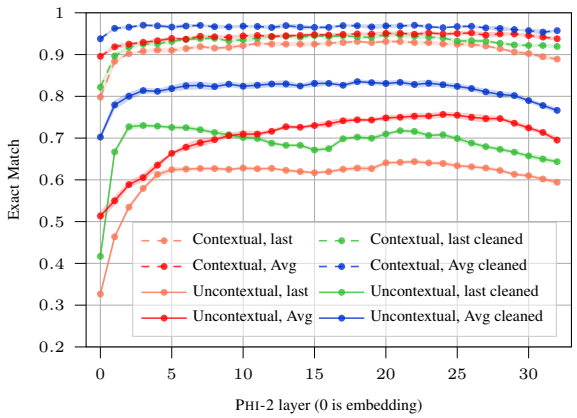


Figure 6: Entity mention generation with different representations. Representations are extracted as described in Section 3.1(last), by averaging (avg) or cleaning (clean) mention representations, as detailed Section 4. Each experiment was conducted 5 times to get an estimate of the variance. (not clearly visible since it is quite small)

Results Overall, we see in Figure 6 that in both uncontextual and contextual setups, the average representation of the tokens from a named entity mention allows to better decode it than the sole representation of the last token (See Figure 6 comparing the red and orange curves). The gain compared to representations of the last token is aligned with the conclusion of our uncontextual mention generation experiment (see Figure 4a), suggesting that the representation of the last token in auto-regressive LLMs does not *in general* encode all the tokens of the entity mention.

In the embedding layer (layer 0 in Figure 6) representations $\bar{\boldsymbol{z}}^0$ are just the average of embeddings. For uncontextual decoding, performance jumps to 51% compared to the 33% of exact matches obtained when generating only from the last token's embedding (solid red and orange curves) – this holds to a lesser extent for contextual decoding too. If the model can disentangle all the tokens from their superposed representation in $\bar{\boldsymbol{z}}^0$, it still has to figure out their order to retrieve the original mention.[6] Upper layers representations may therefore encode a notion of relative token position. The fact that last token representations from middle layers yield better results than the average of the token embeddings (up to 64% vs 51%) however shows that there is more than only token superposition in the representation of an entity mention.

Transforming the representation with a linear model before inserting it at the embedding layer also improves the reconstruction performance in both settings (Green and blue curves in Figure 6). The generation performance gain means that removing or boosting some relevant features – probably associated with non-entity representation subspaces, *common to all entity representation*, helps the model in generating the correct entity mention.

Overall, this demonstrates that using the sole last token representation at a given layer may not be the best choice to represent an entity. Further work is needed though to understand precisely the effect of the linear transformation and averaging.

5 Generalizing relation decoding and logit lens

Relation Extraction Here, we study a complementary question on whether the discussed entity representations can be manipulated (RQ 3). Recent work on the explainability of knowledge manipula-

[6] For instance, "ITALY" is decoded as "YALIT"

tion in transformers (Meng et al., 2022, 2023; Geva et al., 2023; Hernandez et al., 2024; Gottesman and Geva, 2024) support the hypothesis that, for a relation r linking subject and object entities s and o, the representation of the object $\boldsymbol{z}_o$ can be extracted from the representation of the subject $\boldsymbol{z}_s$.

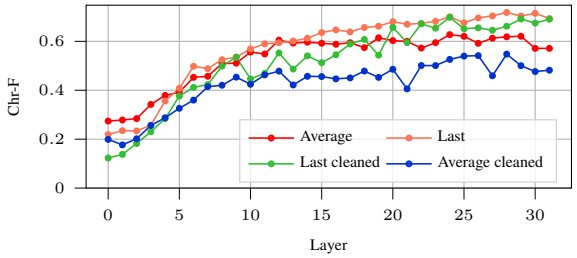


Figure 7: Linear relation decoding on the `Landmarks_to_country` dataset (links landmarks to their home country, *e.g* "`Eiffel tower`" and "`France`"). Averaging or cleaning the representations degrades the semantic information of raw representations from the last token of the entity mention.

We reproduce the idea from Hernandez et al. (2024) that for some basic relations, the association between $\boldsymbol{z}_s$ and $\boldsymbol{z}_o$ can be approximated by a linear model. Our work extends those results by properly accounting for multi-token entities. We train a linear model $\mathcal{L} : \boldsymbol{z}_s \mapsto W\boldsymbol{z}_s + b$ that aims to project the subject representation $\boldsymbol{z}_s$ on the object representation $\boldsymbol{z}_o$, i.e. $\mathcal{L}(\boldsymbol{z}_s) \approx \boldsymbol{z}_o$.

Experimental settings We use datasets from Hernandez et al. (2024) for which there are enough samples to train our linear model. Following their methodology, The data is filtered to keep only samples for which the relation is encoded in the model's parametric memory, guaranteeing that the object entity is represented somewhere. We then optimize our linear model parameters (W, b) using mean-square error on 50 training samples with stochastic gradient descent. We perform this procedure for each layer ℓ, using all studied representations ($\boldsymbol{z}^\ell$,$\bar{\boldsymbol{z}}^\ell$, $\mathcal{C}_{\boldsymbol{z}^\ell}$ or $\mathcal{C}_{\bar{\boldsymbol{z}}^\ell}$) for subject and object entities, allowing to identify which representations carry the most semantic information about an entity.

Results Our mention generation method (See Section 3.1) allows the generation of a mention for the obtained representation and then use *exact match* as a metric instead of only comparing the first token only as in (Hernandez et al., 2024; Geva et al., 2023). In practice, we present the results using the Character F-score (Chr-F), which is not binary and thus produces smoother plots. Figure 7 shows that, training a linear model on only 50 samples leads to over 72% Chr-F (74% EM) on the `Landmarks_to_country` test set. We show the application to other relation datasets with similar results in appendix (Figure 14). Moreover, if using average or cleaned representations yields better performance on mention decoding, this seems to be at the price of losing semantic information, particularly when it comes to encoding relations to other entities – as shown by the fact that averaging or cleaning representations degrades the results.

Entity Lens Thanks to our task vectors, we can generate a mention from representations of any token in a text. This allows visualizing which entity the model is "thinking" about when processing a token. We name this method the *Entity Lens*, generalizing the *logit lens* (nostalgebraist, 2020), that associates multi-token mentions with any given representation, using the learned task vectors from Section 3. For instance, applying the *Entity Lens* to the sentence "`The City of Lights iconic landmark`" shows that the model associates the mention "`City of Lights`" with Paris, and retrieves a representation associated with "`Eiffel Tower`" while processing the token "`landmark`". Figure 8 shows entities decoded from various hidden representations from PHI-2.

6 Conclusion

In this study, we have demonstrated that LLMs develop specialized mechanisms for representing and manipulating entities. Our experiments show that they can effectively be prompted using trained *task vectors* to generate complete mentions from entity representations. When given context, they reliably detect and copy mentions of entities, including those outside their parametric memory. Additionally, we showed that these representations can be semantically manipulated to decode basic relations, extending previous work. A direct application of our methodology, the *Entity Lens*, allows for instance to visualize which entity the model is "thinking" about at a given layer.

Overall, our results support the existence of an entity representation space within LLMs. This understanding paves the way for further research into how LLMs handle and manipulate knowledge, potentially leading to more explainable and controllable language models.

Layer	The	City	of	Lights	iconic	landmark
Emb	U.S. News & World Report 'ory'	City 'cks'	the United States of America 'enson'	Lights 'ham'	San Francisco Giants 'gr'	Hague 'olla'
6	P-1 'ory'	City 'wide'	City 'wide'	City of Lights 'metaphor'	iconic 'ized'	iconic landmark 'ry'
11	The 'ory'	City 'wide'	City of 'wide'	Paris, the City of Light 'green'	iconic 'ocl'	iconic landmark 'status'
21	B. 'first'	City 'wide'	City of 'New'	City of Lights ','	iconic 'city'	landmark ','
26	M 'first'	City 'of'	City of 'New'	Paris ','	Paris 'Paris'	landmark ','
32	The '"'	The City 'of'	City of 'P'	Paris 'is'	Eiffel Tower's iconic 'E'	Eiffel Tower ','

Figure 8: Example application of the *Entity Lens* on the sentence "`The City of Lights iconic landmark`", applied with with the task vectors trained on representations from PHI-2 in the uncontextual setup. PHI-2 associates "`City of Lights`" with Paris, "`landmark`" with the Eiffel Tower in this context. *The token predicted with the logit lens is also shown below.* Additional examples can be found in Section D.1.

7 Limitations

Generalization of Task Vectors Our method also assumes that the representations are layer-agnostic, meaning the LLM operates within a single representation space. More importantly, we assume that the task vector -the only learned parameters- is sufficient to instruct the LLM to decode the entity mention without providing any further information.

This discrepancy between contextual and uncontextual settings is reflected by the fact that learned task vectors are not the same across layers (see Figure 13 in appendix), although we see that they tend to generalize well to other layers (see in appendix, Figure 15). This design choice was motivated by the known performance of *prompt tuning* (Lester et al., 2021), which successfully trains embedding vectors to prompt the model into performing specific tasks.

Entity Lens Our experiments with task vectors, trained specifically on entities, extend the logit lens by enabling multi-token generation of entity mentions from any representation within the transformer. While our current implementation serves as a preliminary demonstration of this capability, further research is needed to fully explore its potential.

Currently, our approach generates only a single mention, whereas the traditional logit lens can retrieve the top-k mappings. This limitation could be addressed by employing beam-search generation, which would allow us to generate the "top-k" entity mentions for a given representation, thereby enhancing the versatility and applicability of our method. Additionally, the task vectors we train are layer-specific, whereas a more practical implementation would leverage one generalist task vector, which we leave for future work.

8 Acknowledgements

The authors acknowledge the ANR – FRANCE (French National Research Agency) for its financial support of the GUIDANCE project n°ANR-23-IAS1-0003 as well as the Chaire Multi-Modal/LLM ANR Cluster IA ANR-23-IACL-0007. This work was granted access to the HPC resources of IDRIS under the allocation 2024-AD011015440R1 made by GENCI.

References

Marah Abdin, Sam Ade Jacobs, A. A. Awan, Jyoti Aneja, Ahmed Awadallah, H. Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Singh Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, C. C. T. Mendes, Weizhu Chen, Vishrav Chaudhary, Parul Chopra, and 65 others. 2024. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2397–2430. PMLR.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Neural Information Processing Systems*.

Trenton Bricken, Chris Olah, and Aldy Tempelton. 2023. Towards Monosemanticity: Decomposing Language Models With Dictionary Learning.

Haozhe Chen, Carl Vondrick, and Chengzhi Mao. 2024. Selfie: self-interpretation of large language model embeddings. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

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.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. 2019. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press.

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.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Chris Olah. 2022. Toy Models of Superposition.

Kawin Ethayarajh. 2019. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 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 55–65, Hong Kong, China. Association for Computational Linguistics.

Javier Ferrando, Oscar Obeso, Senthooran Rajamanoharan, and Neel Nanda. 2025. Do i know this entity? knowledge awareness and hallucinations in language models. *Preprint*, arXiv:2411.14257.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling. *Preprint*, arXiv:2101.00027.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, Singapore. Association for Computational Linguistics.

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.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. 2024. Patchscopes: A Unifying Framework for Inspecting Hidden Representations of Language Models. In *Forty-First International Conference on Machine Learning*.

Daniela Gottesman and Mor Geva. 2024. Estimating knowledge in large language models without generating a single token. In *Proceedings of the 2024*

Conference on Empirical Methods in Natural Language Processing, pages 3994–4019, Miami, Florida, USA. Association for Computational Linguistics.

Roee Hendel, Mor Geva, and Amir Globerson. 2023. In-context learning creates task vectors. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9318–9333, Singapore. Association for Computational Linguistics.

Evan Hernandez, Arnab Sen Sharma, Tal Haklay, Kevin Meng, Martin Wattenberg, Jacob Andreas, Yonatan Belinkov, and David Bau. 2024. Linearity of relation decoding in transformer language models. In *Proceedings of the 2024 International Conference on Learning Representations*.

Mojan Javaheripi and Sébastien Bubeck. 2023. Phi-2: The surprising power of small language models.

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.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, 8:64–77.

D. Jurafsky and J.H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall Series in Artificial Intelligence. Pearson Prentice Hall.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: **phi-1.5** technical report. *arXiv preprint arXiv:2309.05463*.

Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. 2024. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. In *First Conference on Language Modeling*.

Weishu Liu. 2021. Caveats for the use of web of science core collection in old literature retrieval and historical bibliometric analysis. *Technological Forecasting and Social Change*, 172:121023.

Callum McDougall, Arthur Conmy, Cody Rushing, Thomas McGrath, and Neel Nanda. 2023. Copy Suppression: Comprehensively Understanding an Attention Head. *Preprint*, arXiv:2310.04625.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and Editing Factual Associations in GPT. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. 2023. Mass editing memory in a transformer. *The Eleventh International Conference on Learning Representations (ICLR)*.

Tomas Mikolov, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Proceedings of Workshop at ICLR*, 2013.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander Rush. 2023. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460, Singapore. Association for Computational Linguistics.

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

Jingcheng Niu, Andrew Liu, Zining Zhu, and Gerald Penn. 2023. What does the Knowledge Neuron Thesis Have to do with Knowledge? In *The Twelfth International Conference on Learning Representations*.

nostalgebraist. 2020. Interpreting GPT: The logit lens. *LessWrong*.

Koyena Pal, Jiuding Sun, Andrew Yuan, Byron Wallace, and David Bau. 2023. Future lens: Anticipating subsequent tokens from a single hidden state. In *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, page 548–560. Association for Computational Linguistics.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. 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 43–54, Hong Kong, China. Association for Computational Linguistics.

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.

Erik F. Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

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, Florence, Italy. 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.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. 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 4396–4406, Hong Kong, China. Association for Computational Linguistics.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2022. Interpretability in the Wild: A Circuit for Indirect Object Identification in GPT-2 small. *Preprint*, arXiv:2211.00593.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Qinan Yu, Jack Merullo, and Ellie Pavlick. 2023. Characterizing mechanisms for factual recall in language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9924–9959, Singapore. Association for Computational Linguistics.

A Reproducibility statement

All Task vector training have been trained using the CoNLL2003 NER Dataset, as detailed in Section 3.2. We performed 15 epochs on the train split using Adam Optimizer. (Kingma and Ba, 2015)

We provide a repository where we provide a rendered demo notebook, training code, all hyperparameters as well as some checkpoints.[7] We used the `transformer lens` (Nanda and Bloom, 2022), a wrapper around the `transformers` library (Wolf et al., 2020).

All experiments were conducted on cluster nodes with 80GB NVIDIA A100, 16 or 32GB NVIDIA V100 GPUs.

CoNLL Split	Train	Test
Number of samples	22449	11120
Number of unique Entities	7820	2521
Mean text length (in tokens)	26.4	26.4
Mean entity mention length	3 tok	3 tok

Table 1: Statistics of our dataset, processed from CoNLL2003 (Sang and De Meulder, 2003)

B Additional experiments

B.1 Generating random sequences of fixed token length

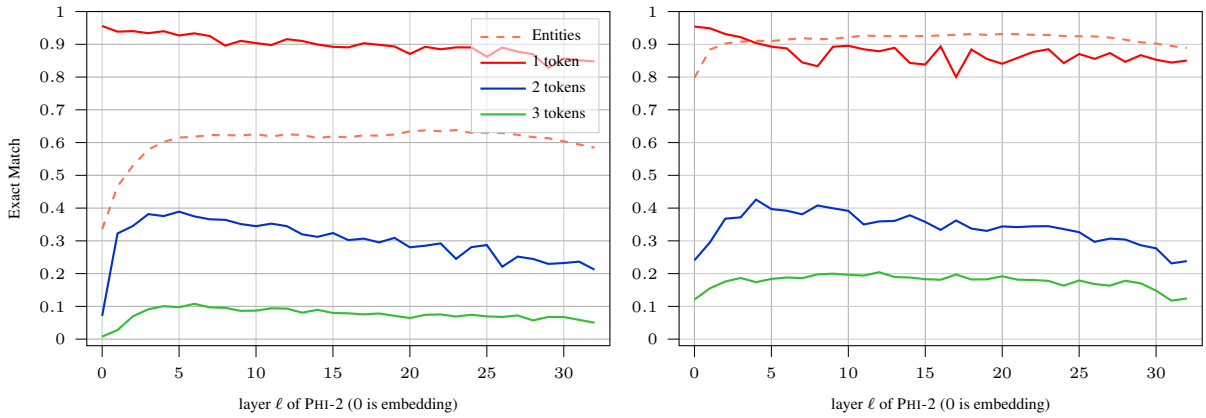
We provide Figure 9 all the results obtained for our control experiment described in Section 3.3

B.2 Generalization of learned Task Vectors

Other layers Each task vectors has been trained with representations from a specific transformer layer, this methodology allows to further analyze the impact of the layer on the quality of the representations. Even if they are not particularly similar (cf Figure 13), task vectors generalize well to other layers. This can be seen in Figure 15, where we apply the *Entity Lens* using the same task vector θ_{20} for all layers. The generated mentions are still consistent even if the representations are not extracted at layer 20.

Different setups Despite being similar, the two setups may imply a different generation mechanism from the model. In the uncontextual setup, we require the model to retrieve a mention from its parametric memory, whereas in the contextual

[7] https://github.com/VictorMorand/EntityRepresentations



(a) Uncontextual mention decoding results by layer.

(b) Contextual mention decoding results by layer.

Figure 9: Baseline – Decoding random sequences of fixed token length compared to decoding entity mentions with PHI-2 (end token is constrained to be the end of a word). If the model can easily retrieve one represented token, manipulating entity mentions is a more natural task then manipulating tokens. The observed behavior is similar on the other models considered.

setup we only require it to copy the right mention. Evaluating task vectors across tasks (Table 2) shows that, despite a 50% drop in performance, task vectors trained for one setting can be used in the other. This demonstrates that while the two processes are different, they however have some similarities. Investigating which is left for future work.

Task Vector	Uncontextual evaluation		Contextual evaluation	
	Exact match	Chr-F	Exact match	Chr-F
Phi-2 ℓ 20, uncontextual	64%	61%	15%	30%
Phi-2 ℓ 20, contextual	41%	40%	93%	94%
Random Vector	0%	13%	0%	17%

Table 2: Despite being trained for distinct tasks, task vectors do show some generalization to other generation settings.

B.3 Analyzing entity representations

Here, we explore how entity mention representations are built inside an LLM.

Successive representations We explore the convergence of the successive representations to the one that we extract at layer ℓ. We consider the cosine similarity of the entity representation $\boldsymbol{z}^\ell$ with the output from different layers, including Multi-head self-attention (Attn) and Multi-Layer Perceptron (MLP) sublayers. Sublayers iteratively *add* or *remove* information in the *residual stream*. There is no predominant layer for the construction of $\boldsymbol{z}^\ell$ (See Figure 10).

Causal analysis is an alternative mean to assess the contribution of one component of a transformer layer on the representation construction. We use

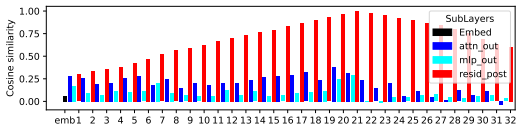


Figure 10: Similarity between the last token representations ($\boldsymbol{z}^\ell$) and intermediate representations from PHI-2. Different sublayers are shown, including outputs from the MLP and Multi Head Self-Attention (Attn). Application to a specific prompt : `'Port|ugal| called| up| Port|o| central| defender| Jo|ao| Manuel| P|into'`. The observed behavior is representative of the general one.

sublayer knockout, as done in (Geva et al., 2023), by zeroing out the output of one MLP or attention block of a given layer ℓ while computing the representation $\boldsymbol{z}^\ell$. We can measure how much this block contributes to this representation by comparing the similarity of the obtained representation with the original one. We observe that, apart from the first layer, no other blocks have a causal effect on the final representation $\boldsymbol{z}^\ell$, confirming the observation made above.

Conclusion Both our similarity and causal analysis experiments lead us to conclude that, as previous work also suggested (Meng et al., 2022; Geva et al., 2023), there is no clear location where the representation of an entity is "completed". The construction of entity representations are a *a smooth, iterative, and massively superposed process.* This aligns with recent contributions on superposition and feature disentanglement, and may be explained by the use of dropout during LLM pre-

training, nudging the model to develop redundant circuits (Elhage et al., 2022; Bricken et al., 2023).

B.4 Training Representations

We explore in this section what features in the representation are really used to generate a mention. To obtain the minimum information required to retrieve the mention, we optimize blank noise to make the model retrieve the right mention when prompted with a task vector trained in the context of our uncontextual mention generation experiment. By doing it several times and averaging the obtained vectors, we hope to keep only hat is needed to regenerate the mention.

Conclusion Our findings demonstrate that it is possible to train a vector to encode almost any mentions, within reasonable token limits. This confirms that the transformer's latent space has the capacity to store numerous tokens. LLMs however typically do not utilize this capability when they can access the context, as they can simply copy and paste the appropriate tokens from it.

C Textual Examples

Original	Inferred
Roberto Mancini	Carlo Mazzone
Pierre Van Hooydonk	Marc D'Haese
Guenther Huber	Peter Huber
Wenchang	Changsha
Michael Cornwell	Mark Calwell
IGLS	GLIS
Ole Einar Bjorndalen	Bjorn Dæhlie
Alba Berlin	Berlin
John Langmore	John Molyneaux
Patasse	Passeau
Rangoon	Yangon
M. Waugh	J. Waugh
Major	John Major
Lahd	Hlad
WARSAW	WAWRZAWA
Kim Pan Keun	Lee Dong-kook
David Boon	J. Boon
Berisha	Bushi
Gunn Margit Andreassen	Ingrid Bjørnson
Abel Balbo	Giuseppe Bologna

Table 3: Examples of failed generations sampled randomly from the the uncontextual mention generation results (Figure 4a) for PHI-2 at layer 15.

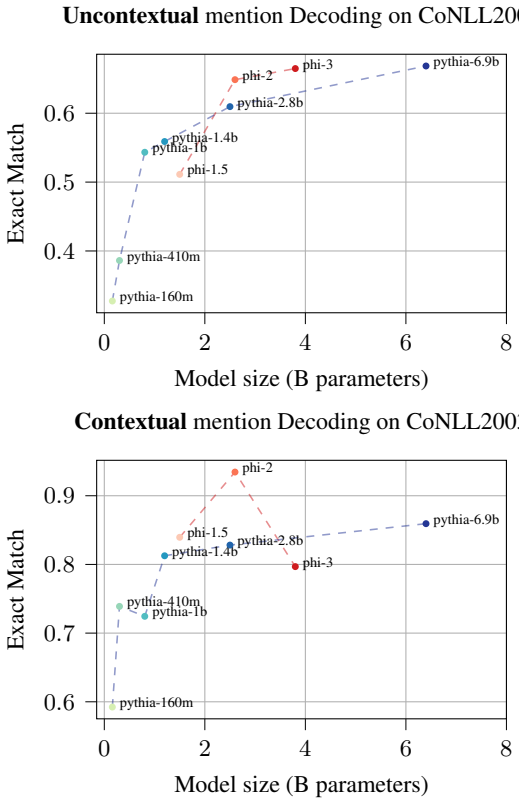


Figure 11: Aggregated Results comparing best performances depending on model size. Larger models demonstrate greater capability. Performance drop of the PHI-3 model can be explained by the use of a significantly smaller vocabulary size (32k vs 50k for all other models considered here) as well as the instruction tuning.

D Complementary Figures

D.1 Entity Lens visualizations

We provide here two example applications of the *Entity Lens*: Figure 8 for the uncontextual setup and Table 5 in the contextual setup. For any layer ℓ and for each token representation $\boldsymbol{z}_k^\ell$, we generate a mention with the layer-specific task Vector θ_ℓ. To test generalization capabilities through layers, we try Figure 15 to use the same task vector θ_{20} for all the layers, empirically validating nice generalization capabilities.

D.2 Performance as a function of Entity mention size and frequency

We provide here the complete results for the analysis conducted in Section 3.3. Reconstruction performance on test set splitted by mention frequence on *the Pile* and mention length with PYTHIAS models is shown Figure 17 in the *uncontextual* mention generation setup and Figure 16 in the *contextual* setup. Figure 18 gathers the results for the three

	n_params	n_layers	d_model	n_heads	act_fn	n_ctx	d_vocab	d_head	d_mlp
PHI-1.5	1.2B	24	2048	32	gelu	2048	51200	64	8192
PHI-2	2.5B	32	2560	32	gelu	2048	51200	80	10240
PHI-3	3.6B	32	3072	32	silu	4096	32064	96	8192
PYTHIA-160M	85M	12	768	12	gelu	2048	50304	64	3072
PYTHIA-410M	302M	24	1024	16	gelu	2048	50304	64	4096
PYTHIA-1B	805M	16	2048	8	gelu	2048	50304	256	8192
PYTHIA-1.4B	1.2B	24	2048	16	gelu	2048	50304	128	8192
PYTHIA-2.8B	2.5B	32	2560	32	gelu	2048	50304	80	10240
PYTHIA-6.9B	6.4B	32	4096	32	gelu	2048	50432	128	16384

Table 4: Characteristics of the models considered in this work.

	G	aston	Julia	and	Mand	el	bro	t	meet	,	the	latter	tells
Emb	Geston	Gaston	Gaston Julia	Mandelbrot	Mandel	Mandelbrot	Mandelbrot	Mandelbrot	Mandelbrot	Mandelbrot	Mandelbrot	Mandelbrot	Mandelbrot
ℓ 6	G	Gaston	Gaston Julia	Gaston Julia and Mandelbrot	Mand	Mandel	Mandelbrot	Mandelbrot	Meeting	Gaston Julia and Mandelbrot	Mandelbrot tells the latter	Mandelbrot	Mandelbrot tells
ℓ 11	G	Gaston	Gaston Julia	Gaston Julia	Mand	Mandelbrot	Mandelbro	Mandelbrot	Meeting	Gaston Julia and Mandelbrot	Gaston Julia	Mandelbrot	Gaston Julia
ℓ 16	G	Gaston	Gaston Julia	Mandelbrot	Mand	Mandel	Mandelbro	Mandelbrot	Meeting	Mandelbrot tells, Mandel	Mandelbrot	Mandelbrot	Mandelbrot tells
ℓ 21	G	Gaston	Gaston Julia	Mandelbrot tells	Mand	Mandelbrot	Mandelbro	Mandelbrot	Meeting	Mandelbrot tells	Mandelbrot tells the latter	Mandelbrot	Tell
ℓ 26	G	Gaston	Gaston Julia	Mandelbrot	Mand	Mandelbrot	Mandelbro	Mandelbrot	Meets	Mandelbrot tells, the latter	Mandelbrot	Mandelbrot	Tell
ℓ 32	Gaston	Gaston	Gaston Julia	Mandelbrot	Mandelbrot	Mandelbrot	Mandelbro	Mandelbrot	Mandelbrot tells meet	Mandelbrot	Mandelbrot	Mandelbrot	Mandelbrot tells

Table 5: Example application of the *Entity Lens*, applied with a task Vector trained on representations extracted in PHI-2 in the contextual setup. We input the sentence "`Gaston Julia and Mandelbrot meet, the latter tells`". We can notably see that the model does associate "`the latter`" with the right entity.

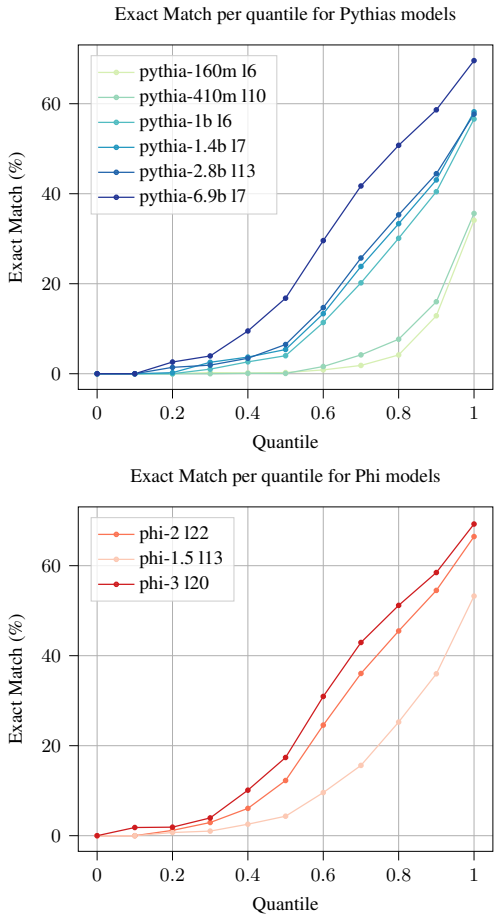
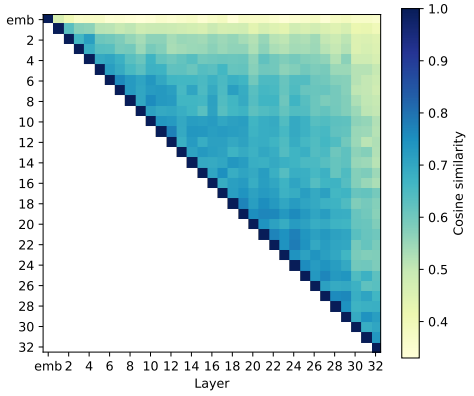
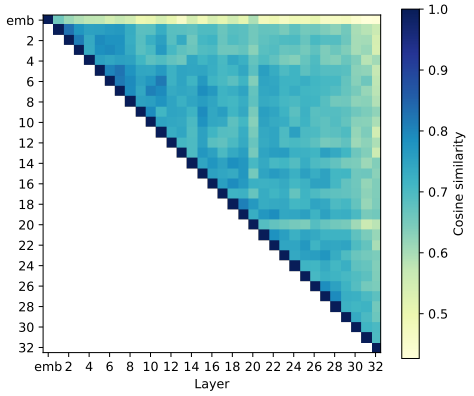


Figure 12: Performance on mention generation without context depending on quantiles of entity frequency in the Pile.

models from the PHI family.

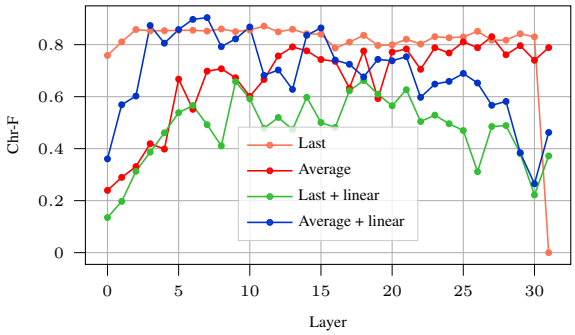


(a) Similarities between trained task vectors in the uncontextual setup

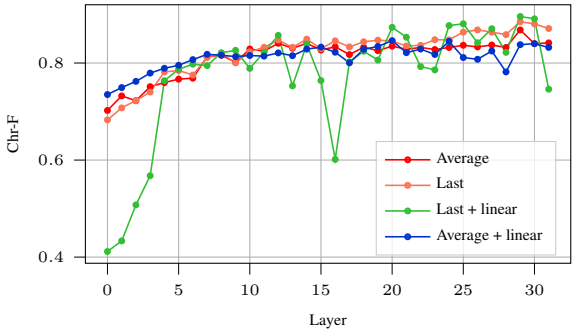


(b) Similarities between trained task vectors in the contextual setup

Figure 13: Cosine Similarity comparison of all trained task Vectors for PHI-2. Training at each layer seems to lead to a different task vector, although they are shown to generalize well (see Section B.2).



(a) Performance on `star_constellation`



(b) Performance on `person_native_language`

Figure 14: Chr-F performance on other datasets from Hernandez et al. (2024).

Layer	The	City	of	Lights	iconic	landmark
Emb	U.S. News & World Report 'ory'	City 'cks'	the United States of America 'enson'	Lights 'ham'	San Francisco Giants 'gr'	Hague 'olla'
6	P-1 'ory'	City 'wide'	City 'wide'	City of Lights 'metaphor'	iconic 'ized'	iconic landmark 'ry'
11	The 'ory'	City 'wide'	City of 'wide'	Paris, the City of Light 'green'	iconic 'ocl'	iconic landmark 'status'
21	B. 'first'	City 'wide'	City of 'New'	City of Lights ','	iconic 'city'	landmark ','
26	M 'first'	City 'of'	City of 'New'	Paris ','	Paris 'Paris'	landmark ','
32	The '"'	The City 'of'	City of 'P'	Paris 'is'	Eiffel Tower's iconic 'E'	Eiffel Tower ','

Figure 15: The *Entity Lens*, applied using only one task vector (θ_{20}, trained on representations extracted at layer 20 of PHI-2 in the uncontextual setup).The model still decodes relevant mentions from representations extracted at different layers, showing the generalizing capabilities of θ_{20} to representations at any layer. This further backs our claim that entity representations are layer agnostic.

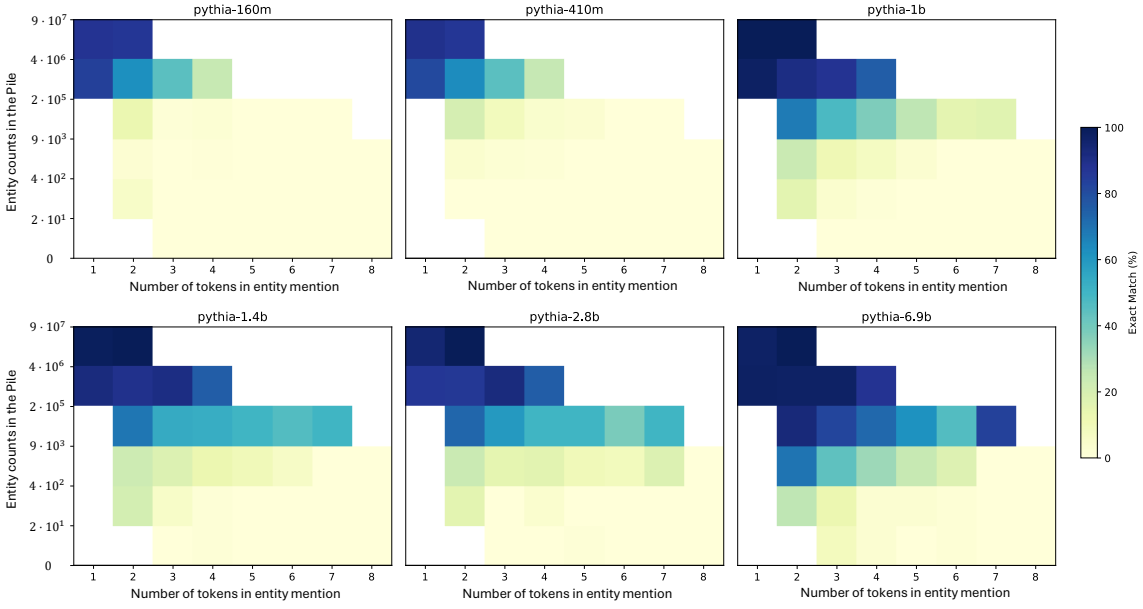


Figure 16: Reconstruction performance on test set for our **uncontextual** mention generation experiment. Performance is separated based on the number of tokens that need reconstruction, as well as the n-gram frequency of the mention in *the Pile* (Gao et al., 2020). For each model, we chose the layer with best exact match on the test set. Empty cells correspond to count/frequency settings with fewer than 5 samples, making it insufficient to compute performance.

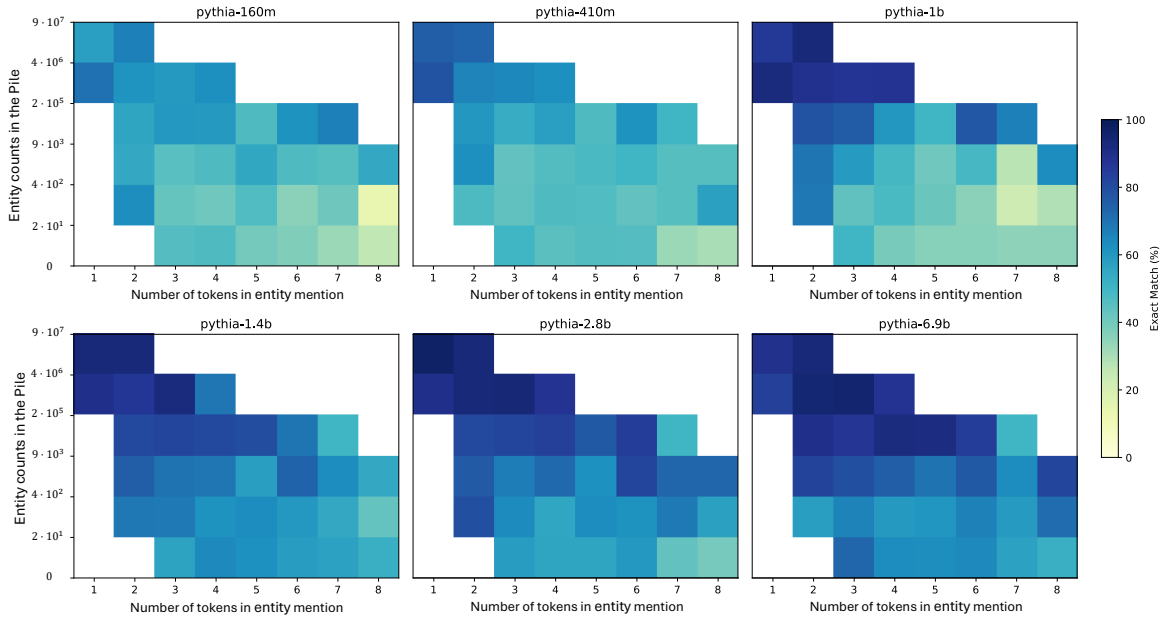
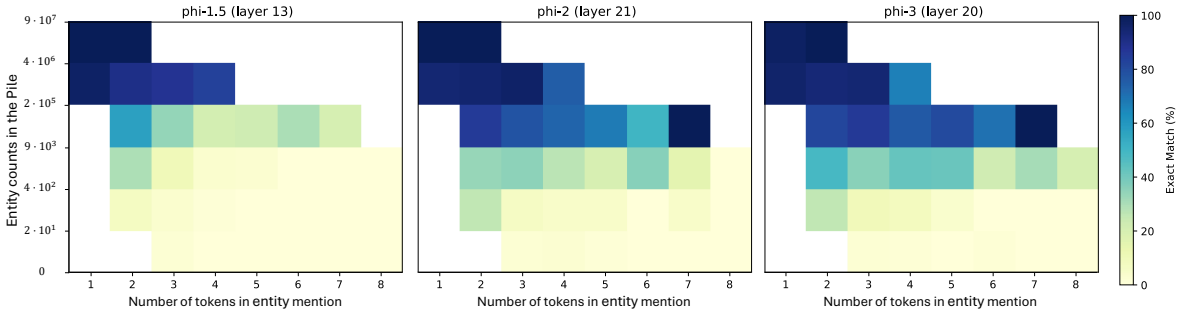
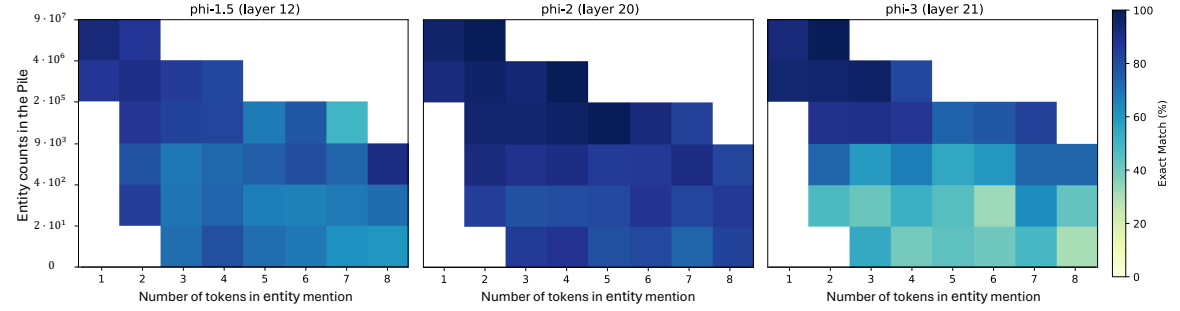


Figure 17: Reconstruction performance on test set for our **contextual** mention generation experiment. Performance is separated based on the number of tokens that need reconstruction, as well as the n-gram frequency of the mention in *the Pile* (Gao et al., 2020). For each model, we chose the layer with best exact match on the test set. Empty cells correspond to count/frequency settings with fewer than 5 samples, making it insufficient to compute performance.



(a) Performance analysis for the uncontextual mention generation experiment



(b) Performance analysis for the contextual mention generation experiment

Figure 18: Reconstruction performance depending on the number of tokens to reconstruct, as well as the n-gram frequency of the entity mention in *the Pile* (Gao et al., 2020). For each model, we chose the layer with best exact match on the test set.