Transformer Feed-Forward Layers Build Predictions by Promoting Concepts in the Vocabulary Space

Mor Geva[*,1]   **Avi Caciularu**[*,2,†]   **Kevin Ro Wang**[3]   **Yoav Goldberg**[1,2]

[1]Allen Institute for AI   [2]Bar-Ilan University   [3]Independent Researcher

morp@allenai.org,{avi.c33,kevinrowang,yoav.goldberg}@gmail.com

Abstract

Transformer-based language models (LMs) are at the core of modern NLP, but their internal prediction construction process is opaque and largely not understood. In this work, we make a substantial step towards unveiling this underlying prediction process, by reverse-engineering the operation of the feed-forward network (FFN) layers, one of the building blocks of transformer models. We view the token representation as a changing distribution over the vocabulary, and the output from each FFN layer as an additive update to that distribution. Then, we analyze the FFN updates in the vocabulary space, showing that each update can be decomposed to sub-updates corresponding to single FFN parameter vectors, each promoting concepts that are often human-interpretable. We then leverage these findings for controlling LM predictions, where we reduce the toxicity of GPT2 by almost 50%, and for improving computation efficiency with a simple early exit rule, saving 20% of computation on average.[1]

1 Introduction

How do transformer-based language models (LMs) construct predictions? We study this question through the lens of the feed-forward network (FFN) layers, one of the core components in transformers (Vaswani et al., 2017). Recent work showed that these layers play an important role in LMs, acting as memories that encode factual and linguistic knowledge (Geva et al., 2021; Da et al., 2021; Meng et al., 2022). In this work, we investigate how outputs from the FFN layers are utilized internally to build predictions.

We begin by making two observations with respect to the representation of a single token in the input, depicted in Fig. 1. First, each FFN layer
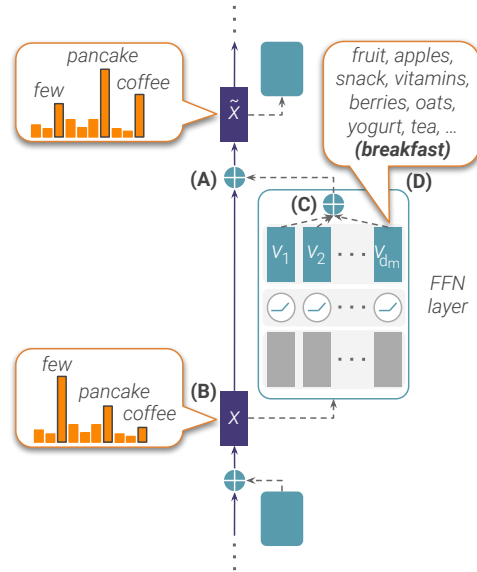
---

*Equal contribution.

[†] Work done during an internship at AI2.

[1]Our codebase is available at https://github.com/aviclu/ffn-values.

Figure 1: Illustration of our findings. Feed-forward layers apply additive updates (A) to the token representation $\mathbf{x}$, which can be interpreted as a distribution over the vocabulary (B). An update is a set of sub-updates induced by parameter vectors $\mathbf{v}_1, ..., \mathbf{v}_{d_m}$ (C), each can be interpreted as a concept in the vocabulary space (D).

induces an additive update to the token representation (Fig. 1,**A**). Second, the token representation across the layers can be translated *at any stage* to a distribution over the output vocabulary (Geva et al., 2021) (Fig. 1,**B**). We reason that the additive component in the update changes this distribution (§2), namely, *FFN layers compute updates that can be interpreted in terms of the output vocabulary*.

We then decompose the FFN update (§3), interpreting it as a collection of sub-updates, each corresponding to a column in the second FFN matrix (Fig. 1,**C**) that scales the token probabilities in the output distribution. Through a series of experiments, we find that (a) sub-update vectors across the entire network often encode a small-set of human-interpretable well-defined concepts, e.g. *"breakfast"* or *"pronouns"* (§4, Fig. 1,**D**), and (b) FFN updates rely primarily on token promotion

(rather than elimination), namely, tokens in the top of the output distribution are those pushed strong enough by sub-updates (§5). Overall, these findings allow fine-grained interpretation of the FFN operation, providing better understanding of the prediction construction process in LMs.

Beyond interpretation, our findings also have practical utility. In §6.1, we show how we can intervene in the prediction process, in order to manipulate the output distribution in a direction of our choice. Specifically, we show that increasing the weight of only 10 sub-updates in GPT2 reduces toxicity in its generations by almost 50%. Also, in §6.2, we show that dominant sub-updates provide a useful signal for predicting an early exit point, saving 20% of the computation on average.

In conclusion, we investigate the mechanism in which FFN layers update the inner representations of transformer-based LMs. We propose that the FFN output can be viewed as a collection of updates that promote concrete concepts in the vocabulary space, and that these concepts are often interpretable for humans. Our findings shed light on the prediction construction process in modern LMs, suggesting promising research directions for interpretability, control, and efficiency.

2 Token Representations as Evolving Distributions Over the Vocabulary

Modern LMs (Baevski and Auli, 2019; Radford et al., 2019; Brown et al., 2020) are transformer models primarily trained to predict the next token probability for a given input. Such LMs are composed of intertwined multi-head self-attention (MHSA) layers and FFN layers (Vaswani et al., 2017), with residual connections (He et al., 2016) between each pair of consecutive layers. The LM prediction is obtained by projecting the output vector from the final layer to an embedding matrix $E \in \mathbb{R}^{|\mathcal{V}| \times d}$, with a hidden dimension d, to get a distribution over a vocabulary $\mathcal{V}$ (after softmax).

Given a sequence $\mathbf{w} = \langle w_1, ..., w_t \rangle$ of input tokens, the model creates a contextualized representation $\mathbf{x}_i \in \mathbb{R}^d$ for each token $w_i \in \mathbf{w}$, that is being updated throughout the layers. In this work, we analyze the updates applied by the FFN layers and how they construct the model prediction. Concretely, each FFN layer $\ell = 1, ..., L$ processes $\mathbf{x}_i^\ell$ and produces an output $\mathbf{o}_i^\ell$, which is then added to $\mathbf{x}_i^\ell$ to

yield an updated representation $\tilde{\mathbf{x}}_i^\ell$:

$$\mathbf{o}_i^\ell = \mathsf{FFN}^\ell(\mathbf{x}_i^\ell)$$
$$\tilde{\mathbf{x}}_i^\ell = \mathbf{x}_i^\ell + \mathbf{o}_i^\ell$$

The updated representation $\tilde{\mathbf{x}}_i^\ell$ then goes through a MHSA layer,[2] yielding the input $\mathbf{x}_i^{\ell+1}$ for the next FFN layer. The evolving representation in this process (i.e. $\mathbf{x}_i^\ell \rightarrow \tilde{\mathbf{x}}_i^\ell, \forall \ell$) can be viewed as an information stream that is being processed and updated by the layers (Elhage et al., 2021). The output probability distribution is obtained from the final representation of the token, i.e.,

$$\mathbf{y} = \mathrm{softmax}(E\tilde{\mathbf{x}}_i^L). \qquad (1)$$

To analyze the FFN updates, we read from the representation *at any layer* a distribution over the output vocabulary, by applying the same projection as in Eq. 1 (Geva et al., 2021):

$$\mathbf{p}_i^\ell = \mathrm{softmax}(E\mathbf{x}_i^\ell)$$
$$\tilde{\mathbf{p}}_i^\ell = \mathrm{softmax}(E\tilde{\mathbf{x}}_i^\ell).$$

Note that $\tilde{\mathbf{p}}_i^L = \mathbf{y}$. Importantly, by linearity:

$$E\tilde{\mathbf{x}}_i^\ell = E\mathbf{x}_i^\ell + E\mathbf{o}_i^\ell,$$

implying that $\mathbf{o}_i^\ell$ can be interpreted as an additive update in the vocabulary space. However, we find that the projection of the FFN output $E\mathbf{o}_i^\ell$ to the vocabulary is not interpretable (§4). In this work, we take this a step further, and decompose the update $\mathbf{o}_i^\ell$ into a set of smaller sub-updates. By projecting the sub-updates to the vocabulary we find that they often express human-interpretable concepts.

In the rest of the paper, we focus on FFN updates to the representation of a single token in the sequence, and omit the token index for brevity, i.e. $\mathbf{x}^\ell := \mathbf{x}_i^\ell$ and $\mathbf{p}^\ell := \mathbf{p}_i^\ell$.

3 The FFN Output as a Collection of Updates to the Output Distribution

We now decompose the FFN output, and interpret it as a set of sub-updates in the vocabulary space.

FFN Outputs as Linear Vector Combinations. Each FFN at layer ℓ consists of two linear transformations with a point-wise activation function in between (bias terms are omitted):

$$\mathsf{FFN}^\ell(\mathbf{x}^\ell) = f\left(W_K^\ell \mathbf{x}^\ell\right) W_V^\ell,$$

---

[2]In some LMs, e.g. GPT2, a layer normalization (LN) (Ba et al., 2016) is applied to the representation $\tilde{\mathbf{x}}_i^\ell$. We omit it here and show it does not influence our interpretation in §3.

where $W_K^\ell, W_V^\ell \in \mathbb{R}^{d_m \times d}$ are parameter matrices, and f is a non-linearity function. Previous work proposed this module can be cast as an emulated neural key-value memory (Sukhbaatar et al., 2015, 2019), where rows in W_K^ℓ and columns in W_V^ℓ are viewed as keys and values, respectively (Geva et al., 2021). For an input $\mathbf{x}^\ell$, the keys produce a vector of coefficients $\mathbf{m}^\ell := f\left(W_K^\ell \mathbf{x}^\ell\right) \in \mathbb{R}^{d_m}$, that weighs the corresponding values in W_V^ℓ. Denoting by $\mathbf{k}_i^\ell$ the i-th row of W_K^ℓ and by $\mathbf{v}_i^\ell$ the i-th column of W_V^ℓ, we can then use the following formulation:

$$\mathsf{FFN}^\ell(\mathbf{x}^\ell) = \sum_{i=1}^{d_m} f(\mathbf{x}^\ell \cdot \mathbf{k}_i^\ell)\mathbf{v}_i^\ell = \sum_{i=1}^{d_m} m_i^\ell \mathbf{v}_i^\ell.$$

Therefore, *a FFN update can be viewed as a collection of sub-updates, each corresponding to a weighted value vector in the FFN output.*

Terminology. In the rest of the paper, we refer to the vectors $\mathbf{v}_i^\ell$ as *value vectors*, and to their weighted form $m_i^\ell \mathbf{v}_i^\ell$ as *sub-updates*. A transformer LM with $L = 10, d_m = 3000$ will have $30,000$ value vectors, and every token that passes through the transformer will weight these value vectors differently, resulting in $30,000$ sub-updates, where only a few of the sub-updates have high weights.

Interpreting Sub-Updates in the Vocabulary Space. Consider a sub-update $m_i^\ell \mathbf{v}_i^\ell$ for a given input, we can estimate its influence on the representation $\mathbf{x}^\ell$ (before the FFN update) by analyzing the change it induces on the output distribution. Concretely, we isolate the effect of $m_i^\ell \mathbf{v}_i^\ell$ on the probability $\mathbf{p}_w^\ell$ of $w \in \mathcal{V}$:[3]

$$\begin{aligned}
&p\left(w \mid \mathbf{x}^\ell + m_i^\ell \mathbf{v}_i^\ell, E\right) \\
&= \frac{\exp\left(\mathbf{e}_w \cdot \mathbf{x}^\ell + \mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell\right)}{Z\left(E(\mathbf{x}^\ell + m_i^\ell \mathbf{v}_i^\ell)\right)} \\
&\propto \exp\left(\mathbf{e}_w \cdot \mathbf{x}^\ell\right) \cdot \exp\left(\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell\right), \quad (2)
\end{aligned}$$

where $\mathbf{e}_w$ is the embedding of w, and $Z(\,\cdot\,)$ is the constant softmax normalization factor.

This implies that each sub-update $m_i^\ell \mathbf{v}_i^\ell$ introduces a scaling factor to the probability of every token w based on its dot product with $\mathbf{e}_w$. Specifically, having $\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell > 0$ increases the probability of w, and having $\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell < 0$ decreases it. This scaling factor can be split into two parts:

- The term $\mathbf{e}_w \cdot \mathbf{v}_i^\ell$ can be viewed as a *static score* of w that is independent of the input to the model. Thus, the projection $\mathbf{r}_i^\ell = E\mathbf{v}_i^\ell \in \mathbb{R}^{|\mathcal{V}|}$ induces a ranking over the vocabulary that allows comparing the scores by $\mathbf{v}_i^\ell$ w.r.t different tokens.

- The term m_i^ℓ is the *dynamic coefficient* of $\mathbf{v}_i^\ell$, which is fixed for all tokens for a given input. Thus, these coefficients allow comparing the contribution of value vectors in a specific update.

Overall, the scaling factor $\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell$ can be viewed as the effective score given by a value vector $\mathbf{v}_i^\ell$ to a token w for a given input.

In the next sections, we use these observations to answer two research questions of (a) What information is encoded in sub-updates and what tokens do they promote? (§4) and (b) How do FFN updates build the output probability distribution? (§5)

4 Sub-Updates Encode Concepts in the Vocabulary Space

We evaluate whether projection to the vocabulary provides a meaningful way to "read" FFN updates, and the extent to which sub-updates are interpretable based on their projections. To this end, we manually inspect the top-scoring tokens by value vectors and check if they express interpretable concepts. Concretely, we consider two representative LMs (details below), and for each vector $\mathbf{v}_i^\ell$ compute a ranking over the vocabulary by sorting the projection $\mathbf{r}_i^\ell$ (§3). Then, we try to detect patterns in the top-scoring tokens of each value vector.

Concepts Annotation Task. We let experts (NLP graduate students) annotate concepts by identifying common patterns among the top-30 scoring tokens of each value vector. For a set of tokens, the annotation protocol includes three steps of: (a) Identifying patterns that occur in at least 4 tokens, (b) describing each recognized pattern, and (c) classifying each pattern as either *"semantic"* (e.g., mammals), *"syntactic"* (e.g., past-tense verbs), or *"names"*. The last class was added only for WIKILM (see below), following the observation that a large portion of the model's vocabulary consists of names. Further details, including the complete instructions and a fully annotated example can be found in App. A.2.

Models. We conduct our experiments over two auto-regressive decoder LMs: The model of Baevski and Auli (2019) (dubbed WIKILM), a 16-layer LM trained on the WIKITEXT-103

---

[3] As in Eq.1, LN is omitted. In App. A.1, we verify empirically that our findings hold also when LN is applied.
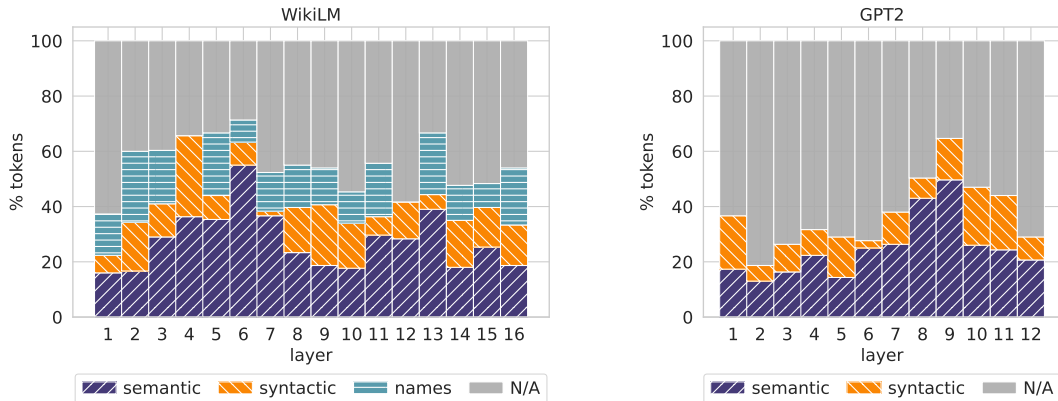
Figure 2: Portion of top-scoring tokens by value vectors in WIKILM and GPT2, that were associated with a semantic or syntactic concept, a name, or could not be matched to any concept ("N/A").

		Concept	Sub-update top-scoring tokens
GPT2	$\mathbf{v}_{1018}^{3}$	Measurement `semantic`	kg, percent, spread, total, yards, pounds, hours
	$\mathbf{v}_{1900}^{8}$	WH-relativizers `syntactic`	which, whose, Which, whom, where, who, wherein
	$\mathbf{v}_{2601}^{11}$	Food and drinks `semantic`	drinks, coffee, tea, soda, burgers, bar, sushi
WIKILM	$\mathbf{v}_{1}^{1}$	Pronouns `syntactic`	Her, She, Their, her, she, They, their, they, His
	$\mathbf{v}_{3025}^{6}$	Adverbs `syntactic`	largely, rapidly, effectively, previously, normally
	$\mathbf{v}_{3516}^{13}$	Groups of people `semantic`	policymakers, geneticists, ancestries, Ohioans

Table 1: Example value vectors in GPT2 and WIKILM promoting human-interpretable concepts.

corpus (Merity et al., 2017) with word-level tokenization ($|\mathcal{V}| = 267,744$), and GPT2 (Radford et al., 2019), a 12-layer LM trained on WEB-TEXT (Radford et al., 2019) with sub-word tokenization ($|\mathcal{V}| = 50,257$). GPT2 uses the GeLU activation function (Hendrycks and Gimpel, 2016), while WIKILM uses ReLU, and in contrast to GPT2, WIKILM does not apply layer normalization after FFN updates. WIKILM defines $d = 1024, d_m = 4096$ and GPT2 defines $d = 768, d_m = 3072$, resulting in a total of $65k$ and $36k$ value vectors, respectively. For our experiments, we sample 10 random vectors per layer from each model, yielding a total of 160 and 120 vectors to analyze from WIKILM and GPT2, respectively.

4.1 Projection of Sub-Updates is Meaningful

Real vs. Random Sub-Updates. We validate our approach by comparing concepts in top-tokens of value vectors and 10 random vectors from a normal distribution with the empirical mean and standard deviation of the real vectors. We observe that a substantially higher portion of top-tokens were associated to a concept in value vectors compared to the random ones (Tab. 2): 55.1% vs. 22.7% in WIKILM, and 37% vs. 16% in GPT2. Also, in both models, the average number of concepts per vector

was > 1 in the value vectors compared to ∼ 0.5 in the random ones. Notably, no semantic nor syntactic concepts were identified in WIKILM's random vectors, and in GPT2, only 4% of the tokens were marked as semantic concepts in the random vectors versus 24.9% in the value vectors.

Updates vs. Sub-Updates. We justify the FFN output decomposition by analyzing concepts in the top-tokens of 10 random FFN outputs per layer (Tab. 2). In WIKILM (GPT2), 39.4% (46%) of the tokens were associated with concepts, but for 19.7% (34.2%) the concept was *"stopwords/punctuation"*. Also, we observe very few concepts (< 4%) in the last two layers of WIKILM. We account this to extreme sub-updates that dominate the layer's output (§5.2). Excluding these concepts results in a considerably lower token coverage in projections of updates compared to those of sub-updates: 19.7% vs. 55.1% in WIKILM, and 11.8% vs. 36.7% in GPT2.

Overall, this shows that projecting sub-updates to the vocabulary provides a meaningful interface to the information they encode. Moreover, decomposing the FFN outputs is necessary for fine-grained interpretation of sub-updates.

	GPT2	WIKILM
FFN sub-updates	**36.7%**	**55.1%**
+ *stopwords concepts*	*37%*	*55.1%*
Random sub-updates	16%	22.7%
FFN updates	11.8%	19.7%
+ *stopwords concepts*	*46%*	*39.4%*

Table 2: Portion of top-scoring tokens associated with a concept, for FFN updates and sub-updates in WIKILM and GPT2, and for random vectors. For FFN updates/sub-updates, we show results with and without counting concepts marked as stopwords.

$\mathbf{p}^\ell$: cow, cat, **dog**, goat, horse, bear
$\tilde{\mathbf{p}}^\ell$: **dog**, cat, goat, horse, cow, bear
Saturation: dog is promoted from rank 3 in $\mathbf{p}^\ell$ to rank 1 in $\tilde{\mathbf{p}}^\ell$, to be the top-candidate until the last layer.

$\mathbf{p}^\ell$: **cow**, cat, dog, goat, horse, bear
$\tilde{\mathbf{p}}^\ell$: dog, cat, goat, horse, **cow**, bear
Elimination: cow is eliminated from rank 1 in $\mathbf{p}^\ell$ to 5 in $\tilde{\mathbf{p}}^\ell$.

Table 3: Example saturation and elimination events, after a FFN update (reference tokens are in bold text).

4.2 Sub-Update Projections are Interpretable

Fig. 2 shows a breakdown of the annotations across layers, for WIKILM and GPT2. In both models and across all layers, a substantial portion (40%-70% in WIKILM and 20%-65% in GPT2) of the top-tokens were associated with well-defined concepts, most of which were classified as *"semantic"*. Also, we observe that the top-tokens of a single value vector were associated with 1.5 (WIKILM) and 1.1 (GPT2) concepts on average, showing that *sub-updates across all layers encode a small-set of well-defined concepts*. Examples are in Tab. 1.

These findings expand on previous results by Geva et al. (2021), who observed that value vectors *in the upper layers* represent next-token distributions that follow specific patterns. Our results, which hold across *all the layers*, suggest that these vectors represent general concepts rather than prioritizing specific tokens.

Underestimation of Concept Frequency. In practice, we find that this task is hard for humans,[4] as it requires reasoning over a set of tokens without any context, while tokens often correspond to uncommon words, homonyms, or sub-words. Moreover, some patterns necessitate world knowledge (e.g. *"villages in Europe near rivers"*) or linguistic background (e.g. negative polarity items). This often leads to undetectable patterns, suggesting that the overall results are an underestimation of the true concept frequency. Providing additional context and token-related information are possible future directions for improving the annotation protocol.

Implication for Controlled Generation. If sub-updates indeed encode concepts, then we can not only interpret their contribution to the prediction, but also *intervene* in this process, by increasing the

weights of value vectors that promote tendencies of our choice. We demonstrate this in §6.1.

5 FFN Updates Promote Tokens in the Output Distribution

We showed that sub-updates often encode interpretable concepts (§4), but how do these concepts construct the output distribution? In this section, we show that sub-updates systematically configure the prediction via promotion of candidate tokens.

5.1 Promoted Versus Eliminated Candidates

Every sub-update $m_i^\ell \mathbf{v}_i^\ell$ either increases, decreases, or does not change the probability of a token w, according to the score $\mathbf{e}_w \cdot m_i^\ell \mathbf{v}_i^\ell$ (§3). This suggests three mechanisms by which tokens are pushed to the top of the output distribution – *promotion*, where sub-updates increase the probability of favorable tokens, *elimination*, where sub-updates decrease candidate probabilities, or a *mixture* of both. To test what mechanism holds in practice, we analyze the scores sub-updates assign to top-candidate tokens by the representation. To simplify the analysis, we focus on changes induced by the 10 most dominant sub-updates in each layer, that is, the 10 sub-updates $m_i^\ell \mathbf{v}_i^\ell$ with the largest contribution to the representation, as measured by $|m_i^\ell| \cdot ||\mathbf{v}_i^\ell||$ (see details in App. A.3).

For the experiments, we use a random sample of 2000 examples from the validation set of WIKITEXT-103,[5] which both WIKILM and GPT2 did not observe during training. As the experiments do not involve human annotations, we use a larger GPT2 model with $L = 24, d = 1024, d_m = 4096$.

We start by comparing the sub-updates' scores to a reference token in two types of events:

- *Saturation* (Tab. 3, up): The update $\mathbf{p}^\ell \to \tilde{\mathbf{p}}^\ell$ where the final token predicted by the model (i.e., $w = \text{argmax}(\mathbf{y})$) was promoted to be the top can-

---

[4] A sub-update annotation took 8.5 minutes on average.

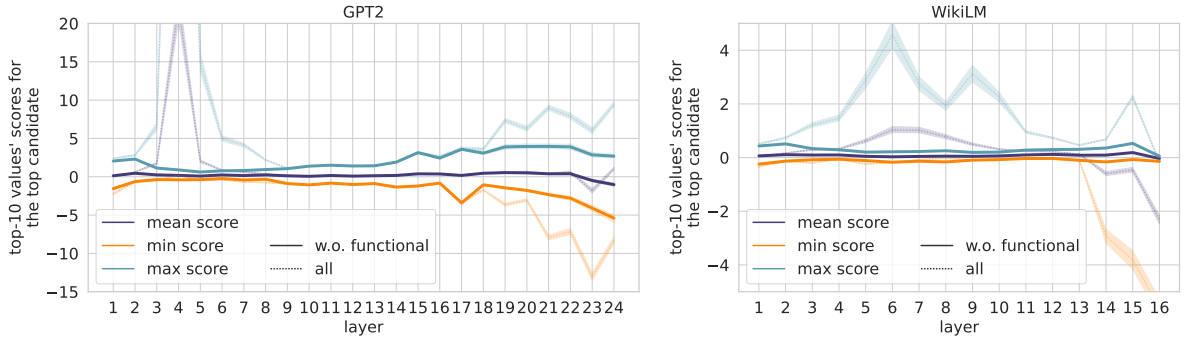[5] Data is segmented into sentences (Geva et al., 2021).

Figure 3: Mean, maximum and minimum scores assigned by the 10 most dominant sub-updates in each layer to the top-candidate token, in GPT2 (left) and WIKILM (right). Solid (dashed) lines exclude (include) functional value vectors. The y-axis in both plots is cut for readability, as the max. (min.) scores reach 100 (-6).

Sub-updates	Event	Max.	Mean	Min.
WIKILM, dominant	*saturation*	1.2	< 0.01	-0.8
	elimination	0.5	-0.01	-0.5
WIKILM, random	*saturation*	0.02	< 0.01	-0.02
	elimination	0.02	< 0.01	-0.02
GPT2, dominant	*saturation*	8.5	1.3	-4.9
	elimination	4.0	0.1	-3.6
GPT2, random	*saturation*	0.2	0.01	-0.2
	elimination	0.1	< 0.01	-0.1

Table 4: Maximum, mean, and minimum scores of reference tokens in saturation and elimination events, by the 10 most dominant and 10 random sub-updates.

didate until the last layer. We analyze saturation events induced by the FFN before the last layer, covering 1184 and 1579 events in WIKILM and GPT2, respectively.

- *Elimination* (Tab. 3, bottom): The update $\mathbf{p}^\ell \to \tilde{\mathbf{p}}^\ell$ with the largest increase in the top candidate's rank, i.e. where the top candidate was dropped behind other candidates to have a rank > 1. Overall, our analysis covers 1909 (WIKILM) and 1996 (GPT2) elimination events.

We compute the mean, maximum, and minimum scores of the reference token by the 10 most dominant sub-updates in each event, and average over all the events. As a baseline, we compute the scores by 10 random sub-updates from the same layer.

Tab. 4 shows the results. In both models, tokens promoted to the top of the distribution receive higher maximum scores than tokens eliminated from the top position ($1.2 \to 0.5$ in WIKILM and $8.5 \to 4.0$ in GPT2), indicating they are pushed strongly by a few dominant sub-updates. Moreover, tokens eliminated from the top of the distribution receive near-zero mean scores, by both dominant and random sub-updates, suggesting they are not being eliminated directly. In contrast to promoted tokens, where the maximum scores are substantially higher than the minimal scores (1.2 vs. -0.8 in WIKILM and 8.5 vs. -4.9 in GPT2), for eliminated tokens, the scores are similar in their magnitude (± 0.5 in WIKILM and 4.0 vs. -3.6 in GPT2). Last, scores by random sub-updates are dramatically lower in magnitude, showing that our choice of sub-updates is meaningful and that higher coefficients translate to greater influence on the output distribution.

This suggests that FFN updates work in a promotion mechanism, where top-candidate tokens are those being pushed by dominant sub-updates.

5.2 Sub-Updates Across Layers

To analyze the FFN operation in different layers, we break down the top-candidate scores per layer. Formally, let $w^\ell = \mathrm{argmax}(\mathbf{p}^\ell)$ be the top candidate at layer ℓ (before the FFN update) for a given input, we extract the scores $\mathbf{e}_{w^\ell} \cdot m_i^\ell \mathbf{v}_i^\ell$ by the 10 most dominant sub-updates and compute the mean, minimum and maximum scores over that set.

Fig. 3 shows that, in both models, until the last few layers (23-24 in GPT2 and 14-16 in WIKILM), maximum and minimum scores are distributed around non-negative mean scores, with prominent peaks in maximum scores (layers 3-5 in GPT2 and layers 4-11 in WIKILM). This suggests that the token promotion mechanism generally holds across layers. However, scores diverge in the last layers of both models, with strong negative minimum scores, indicating that the probability of the top-candidate is pushed down by dominant sub-updates. We next show that these large deviations in positive and negative scores (Fig. 3, dashed lines) result from the operation of small sets of functional value vectors.

Extreme Sub-Updates. To analyze the extreme FFN updates, we first cluster the value vectors to discover high-level trends. We use agglomerative clustering (Müllner, 2011) to learn 10k clusters for each model, based on the cosine distance matrix D, where $D_{(\ell_1, i_1),(\ell_2, i_2)} = 1 - cos(\mathbf{v}_{i_1}^{\ell_1}, \mathbf{v}_{i_2}^{\ell_2})$, $\forall i_1, i_2 \in \{1, \cdots, d_m\}$, $\forall \ell_1, \ell_2 \in \{1, \cdots, L\}$.[6] Then, we search for clusters that are frequently active in extreme updates, by (a) extracting sub-updates where the scores for the top-candidate pass a certain threshold (± 10 for GPT2 and ± 5 for WIKILM), and (b) counting the appearances of each cluster in the layer sub-updates.

In both models, a small set of homogeneous clusters account for the extreme sub-updates shown in Fig. 3, which can be divided into two main groups of value vectors: Vectors in the upper layers that promote *generally unlikely* tokens (e.g. rare tokens), and vectors that are spread over all the layers and promote *common* tokens (e.g. stopwords). These clusters, which cover only a small fraction of the value vectors (1.7% in GPT2 and 1.1% in WIKILM), are mostly active for examples where the input sequence has ≤ 3 tokens or when the target token can be easily inferred from the context (e.g. end-of-sentence period), suggesting that these value vectors might configure "easy" model predictions. More interestingly, the value vectors that promote unlikely tokens can be viewed as *"saturation vectors"*, which propagate the distribution without changing the top tokens. Indeed, these vectors are in the last layers, where often the model already stores its final prediction (Geva et al., 2021).

6 Applications

We leverage our findings for controlled text generation (§6.1) and computation efficiency (§6.2).

6.1 Zero-Shot Toxic Language Suppression

LMs are known to generate toxic, harmful language that damages their usefulness (Bender et al., 2021; McGuffie and Newhouse, 2020; Wallace et al., 2019). We utilize our findings to create a simple, intuitive method for toxic language suppression.

Method. If LMs indeed operate in a promotion mechanism, we reason that we can decrease toxicity by "turning on" non-toxic sub-updates. We find value vectors that promote safe, harmless concepts by extracting the top-tokens in the projections of all

the value vectors and either (a) manually searching for vectors that express a coherent set of positive words (e.g. *"safe"* and *"thank"*), or (b) grading the tokens with the Perspective API and selecting non-toxic value vectors (see details in App. A.4). We turn on these value vectors by setting their coefficients to 3, a relatively high value according to Fig. 3. We compare our method with two baselines:

1. Self-Debiasing (SD) (Schick et al., 2021): SD generates a list of undesired words for a given prompt by appending a *self-debiasing input*, which encourages toxic completions, and calculating which tokens are promoted compared to the original prompt. These undesired words' probability are then decreased according to a decay constant λ, which we set to 50 (default).

2. WORDFILTER: We prevent GPT2 from generating words from a list of banned words by setting any logits that would result in a banned word completion to $-\infty$ (Gehman et al., 2020).

Evaluation. We evaluate our method on the challenging subset of REALTOXICPROMPTS (Gehman et al., 2020), a collection of 1,225 prompts that tend to yield extremely toxic completions in LMs, using the Perspective API, which grades text according to six toxicity attributes. A score of > 0.5 indicates a toxic text w.r.t to the attribute. Additionally, we compute perplexity to account for changes in LM performance. We use GPT2 and, following Schick et al. (2021), generate continuations of 20 tokens.

Results. Finding the non-toxic sub-updates manually was intuitive and efficient (taking < 5 minutes). Tab. 5 shows that activation of only 10 value vectors (0.01%) substantially decreases toxicity ($\downarrow 47\%$), outperforming both SD ($\downarrow 37\%$) and WORDFILTER ($\downarrow 20\%$). Moreover, inducing sub-updates that promote "safety" related concepts is more effective than promoting generally non-toxic sub-updates. However, our method resulted in a perplexity increase greater than this induced by SD, though the increase was still relatively small.

6.2 Self-Supervised Early Exit Prediction

The recent success of transformer-based LMs in NLP tasks has resulted in major production cost increases (Schwartz et al., 2020a), and thus has spurred interest in early-exit methods that reduce the incurred costs (Xu et al., 2021). Such methods often use small neural models to determine when to stop the execution process (Schwartz et al., 2020b;

---

[6]We experimented with $k = 3e^2, 1e^3, 3e^3, 1e^4, 3e^4$, and choose $k = 1e^4$ based on manual inspection.

Model	Toxicity	Severe toxicity	Sexually explicit	Threat	Profanity	Identity attack	PPL
GPT2	58.5%	49.2%	34.1%	16.4%	52.5%	16.8%	21.7
↑ 10 Manual Pick	↓47% 30.8%	↓50% 24.8%	↓40% 20.4%	↓63% 6.0%	↓47% 27.9%	↓48% 8.8%	25.3
↑ 10 API Graded	↓10% 52.7%	↓11% 44%	↓3% 33.2%	↓19% 13.3%	↓9% 47.6%	↓9% 15.3%	23.8
SD	↓37% 37.2%	↓46% 26.4%	↓36% 21.7%	↓52% 7.8%	↓39% 32%	↓50% 8.4%	23.9
WORDFILTER	↓20% 46.9%	↓34% 32.4%	↓36% 21.9%	↓<1% 16.3%	↓38% 32.3%	↓13% 14.7%	-

Table 5: Evaluation results on the challenging subset of REALTOXICPROMPTS, showing the percentage of toxic completions for 6 toxicity attributes, as well as language model perplexity ("PPL").

Elbayad et al., 2020; Hou et al., 2020; Xin et al., 2020, 2021; Li et al., 2021; Schuster et al., 2021).

In this section, we test our hypothesis that dominant FFN sub-updates can signal a *saturation event* (§5.2), to create a simple and effective early exiting method that does not involve any external model training. For the experiments, we use WIKILM, where saturation events occur across all layers (statistics for WIKILM and GPT2 are in App. A.5).

Method. We devise a simple prediction rule based on a nearest-neighbours approach, using 10k validation examples from WIKITEXT-103. First, for every example, we map the top-10 dominant sub-updates at each layer to their corresponding clusters. Then, for every layer ℓ, we split all the sets of clusters at that layer into two sets, T^ℓ and N^ℓ, based on whether saturation occurred or not (e.g., T^5 stores all the sets that were active in a saturation event at layer 5). Given the top-10 clusters of an unseen example at some layer ℓ, we consider a higher overlap with T^ℓ than with $N^{\ell'}$, $\forall \ell' > \ell$ as a signal for early exit. Thus, during inference, we propagate the input example through the layers, and compute at each layer ℓ the intersection size between its top-10 active clusters and each of T^ℓ and $N^{\ell'}$, $\forall \ell' > \ell$. If the average and maximal intersection with T^ℓ exceeds those with $N^{\ell'}$, $\forall \ell' > \ell$, we halt the computation and declare early exiting.[7]

Baselines. We train layer-wise binary classifiers over the representation and FFN updates $\mathbf{x}^\ell$, $\mathbf{o}^\ell$, and $\tilde{\mathbf{x}}^\ell$, using logistic regression. As in our method, the labels are determined according to saturation events in the training data (see App. A.5). During inference, we execute the computation through the layers, and halt according to the layer classifier.

---

[7]This is a simplification. We split N^ℓ by saturation layers and require a bigger intersection with T^ℓ at all the layers.

Method	Accuracy	Saved Layers	
Binary classifiers using $\mathbf{x}^\ell$	94.4±6.4	18.8%	3.0±0.4
Binary classifiers using $\mathbf{o}^\ell$	92.9±5.4	19.4%	3.1±0.3
Binary classifiers using $\tilde{\mathbf{x}}^\ell$	94.4±6.4	18.8%	3.0±0.4
Sub-updates rule	94.1±1.4	20.0%	3.2±0.3

Table 6: Early exit evaluation results on WIKILM.

Evaluation. Each method is evaluated by *accuracy*, i.e., the portion of examples for which exiting at the predicted layer yields the final model prediction, and by *computation efficiency*, measured by the amount of saved layers for examples with correct prediction. We run each method with five random seeds and report the average scores.

Results. Tab. 6 shows that our method obtains a high accuracy of 94.1%, while saving 20% of computation on average without changing the prediction. Moreover, just by observing the dominant FFN sub-updates, it performs on-par with the prediction rules relying on the representation and FFN output vectors. This demonstrates the utility of sub-updates for predicting saturation events, and further supports our hypothesis that FFN updates play a functional role in the prediction (§5.2).

7 Related Work

The lack of interpretability of modern LMs has led to a wide interest in understanding their prediction construction process. Previous works mostly focused on analyzing the evolution of hidden representations across layers (Voita et al., 2019), and probing the model with target tasks (Yang et al., 2020; Clark et al., 2019; Tenney et al., 2019; Saphra and Lopez, 2019). In contrast, our approach aims to interpret the model parameters and their utilization in the prediction process.

More recently, a surge of works have investigated the knowledge captured by the FFN layers (Da et al., 2021; Jiang et al., 2020; Dai et al., 2022;

Yao et al., 2022; Meng et al., 2022; Wallat et al., 2020). These works show that the FFN layers store various types of knowledge, which can be located in specific neurons and edited. Unlike these works, we focus on the FFN outputs and their contribution in the prediction construction process.

Last, our interpretation of FFN outputs as updates to the output distribution relates to recent works that interpreted groups of LM parameters in the discrete vocabulary space (Geva et al., 2021; Khashabi et al., 2022), or viewed the representation as an information stream (Elhage et al., 2021).

8 Conclusions

Understanding the inner workings of transformers is valuable for explainability to end-users, for debugging predictions, for eliminating undesirable behavior, and for understanding the strengths and limitations of NLP models. The FFN is an understudied core component of transformer-based LMs, which we focus on in this work.

We study the FFN output as a linear combination of parameter vectors, termed values, and the mechanism by which these vectors update the token representations. We show that value vectors often encode human-interpretable concepts and that these concepts are promoted in the output distribution.

Our analysis of transformer-based LMs provides a more detailed understanding of their internal prediction process, and suggests new research directions for interpretability, control, and efficiency, at the level of individual vectors.

Limitations

Our study focused on the operation of FFN layers in building model predictions. Future work should further analyze the interplay between these layers and other components in the network, such as attention-heads.

In our analysis, we decomposed the computation of FFN layers into smaller units, corresponding to single value vectors. However, it is possible that value vectors are compositional in the sense that combinations of them may produce new meanings. Still, we argue that analyzing individual value vectors is an important first step, since (a) the space of possible combinations is exponential, and (b) our analysis suggests that aggregation of value vectors is less interpretable than individual value vectors (§4.1). Thus, this approach opens new directions

for interpreting the contribution of FFN layers to the prediction process in transformer LMs.

In addition, we chose to examine the broad family of decoder-based, auto-regressive LMs, which have been shown to be extremely effective for many NLP tasks, including few- and zero-shot tasks (Wang et al., 2022). While these models share the same building blocks of all transformer-based LMs, it will be valuable to ensure that our findings still hold for other models, such as encoder-only LMs (e.g. RoBERTa (Liu et al., 2019)) and models trained with different objective functions (e.g. masked language modeling (Devlin et al., 2019)).

Finally, our annotation effort was made for the evaluation of our hypothesis that sub-updates encode human-interpretable concepts. Scaling our annotation protocol would enable a more refined map of the concepts, knowledge and structure captured by LMs. Furthermore, since our concept interpretation approach relies on manual inspection of sets of tokens, its success might depend on the model's tokenization method. In this work, we analyzed models with two different commonly-used tokenizers, and future research could verify our method over other types of tokenizations as well.

Ethics Statement

Our work in understanding the role that single-values play in the inference that transformer-based LMs perform potentially improves their transparency, while also providing useful control applications that save energy (early-exit prediction) and increase model harmlessness (toxic language suppression). It should be made clear that our method for toxic language suppression only reduces the probability of toxic language generation and does not eliminate it. As such, this method (as well as our early-exit method) should not be used in the real world without further work and caution.

More broadly, our work suggests a general approach for modifying LM predictions in particular directions, by changing the weights of FFN sub-updates. While this is useful for mitigating biases, it also has the potential for abuse. It should be made clear that, as in the toxic language suppression application, our approach does not modify the information encoded in LMs, but only changes the intensity in which this information is exposed in the model's predictions. Moreover, our work primarily proposes an interpretation for FFN sub-updates, which also could be used to identify abusive inter-

ventions. Regardless, we stress that LMs should not be integrated into critical systems without caution and monitoring.

Acknowledgements

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Alexei Baevski and Michael Auli. 2019. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations (ICLR)*.

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 *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT)*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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.

Jeff Da, Ronan Le Bras, Ximing Lu, Yejin Choi, and Antoine Bosselut. 2021. Analyzing commonsense emergence in few-shot knowledge models. In *3rd Conference on Automated Knowledge Base Construction*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics (NAACL)*, pages 4171–4186, Minneapolis, Minnesota.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. 2020. Depth-adaptive transformer. In *International Conference on Learning Representations (ICLR)*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*. Https://transformer-circuits.pub/2021/framework/index.html.

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, 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.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the conference on computer vision and pattern recognition (CVPR)*.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. *Advances in Neural Information Processing Systems (NeurIPS)*.

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.

Daniel Khashabi, Xinxi Lyu, Sewon Min, Lianhui Qin, Kyle Richardson, Sean Welleck, Hannaneh Hajishirzi, Tushar Khot, Ashish Sabharwal, Sameer Singh, and Yejin Choi. 2022. Prompt waywardness: The curious case of discretized interpretation of continuous prompts. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3631–3643, Seattle, United States. Association for Computational Linguistics.

Lei Li, Yankai Lin, Deli Chen, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. 2021. CascadeBERT: Accelerating inference of pre-trained language models via calibrated complete models cascade. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 475–486, Punta Cana, Dominican Republic. 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. 2019. RoBERTa: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Kris McGuffie and Alex Newhouse. 2020. The radicalization risks of gpt-3 and advanced neural language models. *arXiv preprint arXiv:2009.06807*.

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

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. *International Conference on Learning Representations (ICLR)*.

Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*.

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.

Naomi Saphra and Adam Lopez. 2019. Understanding learning dynamics of language models with SVCCA. 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 3257–3267, Minneapolis, Minnesota. Association for Computational Linguistics.

Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in NLP. *Transactions of the Association for Computational Linguistics*, 9:1408–1424.

Tal Schuster, Adam Fisch, Tommi Jaakkola, and Regina Barzilay. 2021. Consistent accelerated inference via confident adaptive transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4962–4979, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. 2020a. Green AI. *Communications of the ACM*, 63(12):54–63.

Roy Schwartz, Gabriel Stanovsky, Swabha Swayamdipta, Jesse Dodge, and Noah A. Smith. 2020b. The right tool for the job: Matching model and instance complexities. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6640–6651, Online. Association for Computational Linguistics.

Noam M. Shazeer. 2020. Glu variants improve transformer. *ArXiv*, abs/2002.05202.

S. Sukhbaatar, J. Weston, and R. Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems (NIPS)*.

Sainbayar Sukhbaatar, Edouard Grave, Guillaume Lample, Herve Jegou, and Armand Joulin. 2019. Augmenting self-attention with persistent memory. *arXiv preprint arXiv:1907.01470*.

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.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008.

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.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. 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 2153–2162, Hong Kong, China. Association for Computational Linguistics.

Jonas Wallat, Jaspreet Singh, and Avishek Anand. 2020. BERTnesia: Investigating the capture and forgetting of knowledge in BERT. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 174–183, Online. Association for Computational Linguistics.

Thomas Wang, Adam Roberts, Daniel Hesslow, Teven Le Scao, Hyung Won Chung, Iz Beltagy, Julien Launay, and Colin Raffel. 2022. What language model architecture and pretraining objective works best for zero-shot generalization? In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 22964–22984. PMLR.

Ji Xin, Rodrigo Nogueira, Yaoliang Yu, and Jimmy Lin. 2020. Early exiting BERT for efficient document ranking. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 83–88, Online. Association for Computational Linguistics.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. BERxiT: Early exiting for BERT with better fine-tuning and extension to regression. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 91–104, Online. Association for Computational Linguistics.

Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. 2021. A survey on green deep learning. *arXiv preprint arXiv:2111.05193*.

Yilin Yang, Longyue Wang, Shuming Shi, Prasad Tadepalli, Stefan Lee, and Zhaopeng Tu. 2020. On the sub-layer functionalities of transformer decoder. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4799–4811, Online. Association for Computational Linguistics.

Yunzhi Yao, Shaohan Huang, Ningyu Zhang, Li Dong, Furu Wei, and Huajun Chen. 2022. Kformer: Knowledge injection in transformer feed-forward layers. *arXiv preprint arXiv:2201.05742*.
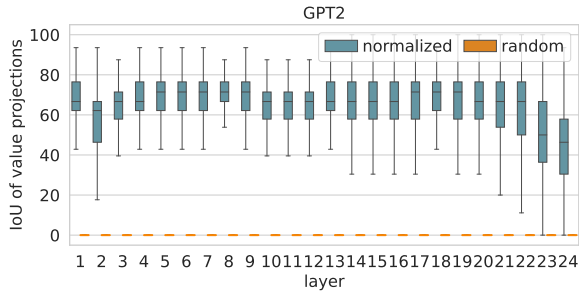
Figure 4: Similarity of projections to E, of GPT2 value vectors with and without layer normalization, and of value vectors and randomly-initialized vectors. We measure similarity of the top-30 tokens in each projection with IoU.

A   Appendix

A.1   Value Vectors Projection Method

Our interpretation method of sub-updates is based on directly projecting value vectors to the embedding matrix, i.e. for a value $\mathbf{v}$ and embedding matrix E, we calculate $E\mathbf{v}$ (§4). However, in some LMs like GPT2, value vectors in each layer are added to the token representation followed by a layer normalization (LN) (Ba et al., 2016). This raises the question whether "reading" vectors that are normalized in the same manner as the representation would yield different concepts.

To test that, we compare the top-30 scoring tokens by $E\mathbf{v}_i^\ell$ and by $E \cdot \text{LayerNorm}(\mathbf{v}_i^\ell)$, for $i = 1, ..., d_m$ and $\ell = 1, ..., L$, using Intersection over Union (IoU). As a baseline, we also compare $E\mathbf{v}_i^\ell$ with random vectors, initialized from a normal distribution with the empirical mean and standard deviation of the value vectors. Fig. 4 shows that LN does not change the projection substantially, with an overlap of 64.5% of the top-30 tokens on average, suggesting that the same concepts are promoted in both cases. This is in contrast to random values, which produce a $\sim 0\%$ overlap on average.

A.2   Concepts Annotation

We analyze the concepts encoded in sub-updates, by projecting their corresponding value vectors to the embedding matrix and identifying repeating patterns in the top-scoring 30 tokens (§3). Pattern identification was performed by experts (NLP graduate students), following the instructions presented in Fig. 5. Please note these are the instructions provided for annotations of WIKILM, which uses word-level tokenization. Thus, the terms "words" and "tokens" are equivalent in this case.

For value vectors in WIKILM, which uses a word-level vocabulary with many uncommon words, we additionally attached a short description field for each token that provides context about the meaning of the word. For the description of a token w, we first try to extract the definition of w from Wordnet.[8] If w does not exist in Wordnet, as often happens for names of people and places, we then search for w in Wikipedia[9] and extract a short (possibly noisy) description if the query was successful. A complete annotation example Tab. 7.

A.3   Sub-Update Contribution in FFN Outputs

In this section, we justify our choice along the paper of looking at the top-10 dominant sub-updates. The contribution of a sub-update $m_i^\ell \mathbf{v}_i^\ell$ to the FFN output is:

$$\text{contrib}(m_i^\ell \mathbf{v}_i^\ell) := \frac{|m_i^\ell| \|\mathbf{v}_i^\ell\|}{\sum_{j=1}^{d_m} |m_j^\ell| \|\mathbf{v}_j^\ell\|},$$

namely, its relative weight compared to the overall sum of weights of all sub-updates. The overall contribution of the top-10 dominant sub-updates is computed by summing their contributions. Note that we take the absolute value of the coefficients $|m_i^\ell|$, since some activation functions (e.g. GeLU (Hendrycks and Gimpel, 2016) in GPT2), can result in negative values of m_i^ℓ.

Empirically, we observe that in some cases sub-updates with negative coefficients do appear as part of the 10 most dominant sub-updates in GPT2. We further attribute this to the success of GeLU in transformer models (Shazeer, 2020), as it increases the expressiveness of the model by allowing reversing the scores value vectors induce over the vocabulary.

Fig. 6 depicts the contribution of the top-10 dominant sub-updates per layer for WIKILM and GPT2, using 2000 random examples from the WIKITEXT-103 validation set. Clearly, for all the layers, the contribution of the dominant sub-updates exceeds the contribution of random sub-updates. Observe that, even though they cover only 0.24% of the value vectors, the contribution of dominant sub-updates is typically around 5%, and in some layers (e.g. layers 8-16 in WIKILM and layer 1 in GPT2) it reaches over 10% of the total

---

[8]We use the NLTK python package.
[9]Using the wptools package https://pypi.org/project/wptools/.

Figure 5: Annotation instructions for the concepts identification task.

contribution. This demonstrates that analyzing the top-10 dominant sub-updates can shed light on the way predictions are built through the layers.

A.4 Toxic Language Suppression Details

The 10 manually selected value vectors were found by searching for non-toxic words, such as *"safe"* and *"peace"*, among the top-30 tokens in the vector projections to the vocabulary. We selected a small set of 10 value vectors whose top-scoring tokens were coherent and seemed to promote different kinds of non-toxic tokens. The list of manually picked vectors is provided in Tab. 8. Importantly, the search process of all vectors was a one-time effort that took < 5 minutes in total. We chose the value vectors in a greedy-manner, without additional attempts to optimize our choice.

To select 10 non-toxic value vectors based on an automatic toxicity metric, we used the Perspective API. Concretely, we concatenated the top-30 tokens by each value vector and graded the resulting text with the toxicity score produced by the API. Then, we sampled 10 random vectors with a toxicity score < 0.1 (a score of < 0.5 indicates a non-toxic text).

A.5 Early Exit Details

This section provides further details and analysis regarding our early exit method and the baselines we implemented.

Method Implementation. We consider 90% of the 10k examples for constructing T^ℓ and N^ℓ, and the remaining 10% examples are considered as the testing set. We used $k = 2e^2$ to cluster the top-10

dominant value vectors, but observed that other k values yielded similar results.

Baselines' Implementation. We train each binary classifier using 8k training examples, based on the standardized forms of each feature vector. We considered a hyperparameter sweep, using 8-fold cross-validation, with $l2$ or $l1$ regularization (lasso (Tibshirani, 1996) or ridge (Hoerl and Kennard, 1970)), regularization coefficients $C \in \{1e^{-3}, 1e^{-2}, 1e^{-1}, 1, 1e^1, 1e^2, 1e^3\}$, and took the best performing model for each layer. We also used a inversely proportional loss coefficient according to the class frequencies.

In order to achieve high accuracy, we further calibrate a threshold per classifier for reaching the maximal F_1 score for each layer. This calibration is done after training each classifier, over a set of 1000 validation examples.

Frequency of Saturation Events. We investigate the potential of performing early exit for WIKILM and GPT2. Tab. 9 and 10 depict the frequency of saturation events per layer, considering 10k examples from the WIKITEXT-103 validation set, for WIKILM and GPT2, respectively. In GPT2, 34.15% of the examples require the full computation using all the model layers, while for WIKILM, this holds for only 15.22% of the examples. Notably, early fixation events in GPT2 are less common than in WIKILM, possibly due to the larger number of layers the prediction construction is spread over. Hence, we use WIKILM for our experiments, as it has significantly higher compu-

patterns	word	description
1	front	the side that is forward or prominent
1	ahead	having the leading position or higher score in a contest
1	forward	the person who plays the position of forward in certain games, such as basketball, soccer, or hockey
1	preceded	be earlier in time; go back further
1	Before	earlier in time; previously
1	before	earlier in time; previously
1	rear	the back of a military formation or procession
1	fore	front part of a vessel or aircraft
2	Name	a language unit by which a person or thing is known
1	Past	the time that has elapsed
1	prior	the head of a religious order; in an abbey the prior is next below the abbot
1	anterior	a tooth situated at the front of the mouth
1	upperparts	standard terms for unambiguous description of relative placement of body parts
1	lead	an advantage held by a competitor in a race
1	backwards	at or to or toward the back or rear
1	aft	(nautical, aeronautical) situated at or toward the stern or tail
1	preceding	be earlier in time; go back further
1	upstream	in the direction against a stream's current
	hind	any of several mostly spotted fishes that resemble groupers
1	posterior	the fleshy part of the human body that you sit on
	Etymology	a history of a word
1	Pre	Wikimedia disambiguation page
	chin	the protruding part of the lower jaw
1	north	the region of the United States lying to the north of the Mason-Dixon line
1	east	the cardinal compass point that is at 90 degrees
2	surname	the name used to identify the members of a family (as distinguished from each member's given name)
1	Then	that time; that moment
2	name	a language unit by which a person or thing is known
1	northbound	moving toward the north
1	leading	thin strip of metal used to separate lines of type in printing

pattern id	description (optional)	semantic/syntactic
1	positions/ directions	semantic
2	naming	semantic

Table 7: An example annotation spreadsheet of the top-tokens by the value vector $\mathbf{u}_{1090}^6$ in WIKILM.
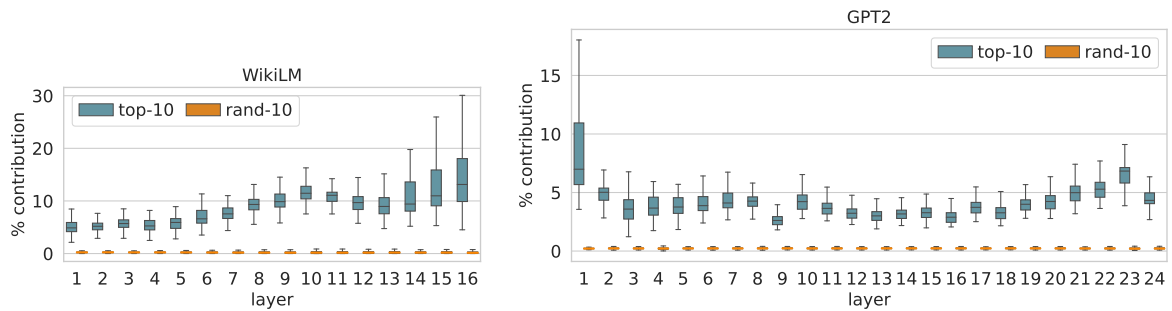

Figure 6: Relative contribution to the FFN output of the 10 most dominant and 10 random sub-updates in each layer, of WIKILM (left) and GPT2 (right).

tation saving potential, as well as more saturation events per layer.

44

Value	Top-10 Tokens
$\mathbf{v}_{1853}^{14}$	transparency, disclosure, clearer, parency, iquette, humility, modesty, disclosures, accountability, safer
$\mathbf{v}_{73}^{15}$	respectful, honorable, healthy, decent, fair, erning, neutral, peacefully, respected, reconc
$\mathbf{v}_{1395}^{15}$	safe, neither, safer, course, safety, safe, Safe, apologize, Compact, cart
$\mathbf{v}_{216}^{16}$	refere, Messages, promises, Relations, accept, acceptance, Accept, assertions, persistence, warn
$\mathbf{v}_{462}^{17}$	should, should, MUST, ought, wisely, Should, SHOULD, safely, shouldn, urgently
$\mathbf{v}_{3209}^{17}$	peaceful, stable, healthy, calm, trustworthy, impartial, stability, credibility, respected, peace
$\mathbf{v}_{4061}^{17}$	Proper, proper, moder, properly, wisely, decency, correct, corrected, restraint, professionalism
$\mathbf{v}_{2921}^{18}$	thank, THANK, thanks, thank, Thank, apologies, Thank, thanks, Thanks, apologise
$\mathbf{v}_{1891}^{19}$	thanks, thank, Thanks, thanks, THANK, Thanks, Thank, Thank, thank, congratulations
$\mathbf{v}_{3770}^{23}$	free, fit, legal, und, Free, leg, pless, sound, qualified, Free

Table 8: The 10 manually picked value vectors used for toxic language suppression and the top-10 tokens in their projection to the vocabulary. Repetitions in the projections are a result of special characters not being shown. These vectors were found by manually searching for non-toxic words such as *"safe"* and *"peace"* in the projections to the vocabulary.

Layer	% Examples	Layer	% Examples
1	6.70	9	2.96
2	5.25	10	3.78
3	13.74	11	4.74
4	3.13	12	7.45
5	1.02	13	10.79
6	1.07	14	9.88
7	1.86	15	9.81
8	2.60	16	15.22

Table 9: The percentage of saturation events per layer using WIKILM, for the WIKITEXT-103 validation set.

Layer	% Examples	Layer	% Examples
1	2.21	13	1.24
2	0.77	14	1.62
3	1.06	15	2.37
4	0.74	16	2.72
5	0.85	17	2.99
6	0.83	18	3.80
7	0.83	19	4.15
8	0.72	20	5.21
9	0.93	21	5.67
10	0.99	22	9.31
11	1.16	23	14.52
12	1.32	24	34.15

Table 10: The percentage of saturation events per layer using GPT2, for the WIKITEXT-103 validation set.