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]

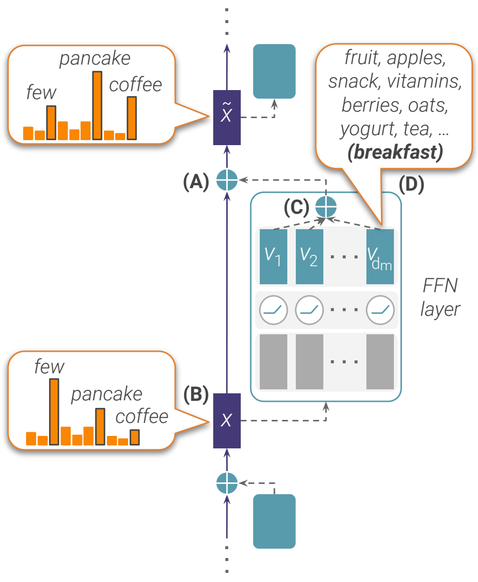

Figure 1: Illustration of our findings. Feed-forward layers apply additive updates (A) to the token representation $\mathbf{x}$, which can be interpreted as a distribution over the vocabulary (B). An update is a set of sub-updates induced by parameter vectors $\mathbf{v}_1, ..., \mathbf{v}_{d_m}$ (C),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

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 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)

[*] Equal contribution.
[†] Work done during an internship at AI2.
[1]Our codebase is available at `https://github.com/aviclu/ffn-values`.

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.,

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

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):

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

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]

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

where $\mathbf{e}_w$ is the embedding of w, and $Z\left(\,\cdot\,\right)$ 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:

[3]As in Eq.1, LN is omitted. In App. A.1, we verify empirically that our findings hold also when LN is applied.

- 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