Knowledge Neurons in Pretrained Transformers

Abstract

Large-scale pretrained language models are surprisingly good at recalling factual knowledge presented in the training corpus (Petroni et al., 2019; Jiang et al., 2020b). In this paper, we present preliminary studies on how factual knowledge is stored in pretrained Transformers by introducing the concept of knowledge neurons. Specifically, we examine the fill-in-the-blank cloze task for BERT. Given a relational fact, we propose a knowledge attribution method to identify the neurons that express the fact. We find that the activation of such knowledge neurons is positively correlated to the expression of their corresponding facts. In our case studies, we attempt to leverage knowledge neurons to edit (such as update, and erase) specific factual knowledge without fine-tuning. Our results shed light on understanding the storage of knowledge within pretrained Transformers. The code is available at http://anonymous.url.

1 Introduction

Large-scale pretrained Transformers (Devlin et al., 2019; Liu et al., 2019; Dong et al., 2019; Clark et al., 2020; Bao et al., 2020) are usually learned with a language modeling objective on large-scale corpora, such as Wikipedia, where exists oceans of factual knowledge. Pretrained language models naturally play as a free-text knowledge base by predicting texts (Bosselut et al., 2019). Petroni et al. (2019) and Jiang et al. (2020b) probe factual knowledge stored in pretrained language models by fill-in-the-blank cloze queries. The evaluation shows that pretrained Transformers have a strong ability to recall factual knowledge without any fine-tuning. Roberts et al. (2020) use closed-book question answering to show that the larger a model is, the more knowledge it can store. However, most previous work focuses on evaluating the overall accuracy of text-form knowledge prediction. In this paper, we attempt to look deeper into pretrained Transformers and investigate how factual knowledge is stored.

As shown in Figure 1, we propose a knowledge attribution method to identify the neurons that express a relational fact, where such neurons are named knowledge neurons. Specifically, we view feed-forward network (i.e., two-layer perceptron) modules in Transformer as key-value memories (Geva et al., 2020). For the example in Figure 1, the hidden state is fed into the first linear layer and activates knowledge neurons; then, the second linear layer integrates the corresponding memory vectors. The key-value-memory nature (Geva et al., 2020) inspires us to propose the knowledge attribution method, which identifies knowledge neurons in feed-forward networks by computing the contribution of each neuron to the knowledge prediction.

Extensive analysis shows that the activation of the identified knowledge neurons is positively correlated to the knowledge expression, which shows the effectiveness of the proposed knowledge at-
We present two preliminary studies: updating facts, we find that knowledge neurons of a fact tend to express specific factual knowledge in pretrained Transformers by directly modifying FFN weights, the second linear layer knowledge neurons in the FFN module are responsible for expressing factual knowledge. Such surgery shows promising results, keeping a moderate influence on other knowledge.

In our case studies, we try to leverage knowledge neurons to explicitly edit factual knowledge in pretrained Transformers without any fine-tuning. We present two preliminary studies: updating facts, and erasing relations. After identifying the knowledge neurons, we perform a knowledge surgery for pretrained Transformers by directly modifying the corresponding parameters in feed-forward networks. Such surgery shows promising results, keeping a moderate influence on other knowledge.

Our contributions are summarized as follows:

- We introduce the concept of knowledge neurons and propose a knowledge attribution method to identify the knowledge neurons that express specific factual knowledge in the fill-in-the-blank cloze task.
- We conduct both qualitative and quantitative analysis to show that knowledge neurons are positively correlated to knowledge expression.
- We present preliminary studies of leveraging knowledge neurons to edit factual knowledge in Transformers, even without any fine-tuning.

2 Background: Transformer

Transformer (Vaswani et al., 2017) is one of the most popular and effective NLP architectures. A Transformer encoder is stacked with L identical blocks. Each Transformer block mainly contains two modules: a self-attention module, and a feed-forward network (abbreviated as FFN) module. Let $X \in \mathbb{R}^{n \times d}$ denote the input matrix, two modules can be formulated as follows:

$$Q_h = XW_h^Q, \ K_h = XW_h^K, \ V_h = XW_h^V, \ (1)$$
$$\text{Self-Att}_h(X) = \text{softmax}(Q_hK_h^T)V_h, \ (2)$$
$$\text{FFN}(H) = \text{gelu}(HW_1)W_2, \ (3)$$

where $W_h^Q, W_h^K, W_h^V, W_1, W_2$ are parameter matrices; $\text{Self-Att}_h(X)$ computes a single attention head; $H$, the hidden state, is given by projecting the concatenation of all heads; $\text{gelu}$ denotes the GELU activation function (Hendrycks and Gimpel, 2016). For simplicity, we omit the scaling factor in self-attention and the bias terms.

Connections Between Self-Attention and FFN

Comparing Equation (2) and Equation (3), we notice that the formula of $\text{FFN}(\cdot)$ is quite similar to $\text{Self-Att}(\cdot)$, except the activation function $\text{gelu}$ in FFN and $\text{softmax}$ in self-attention. Thus, similar to the query-key-value mechanism in self-attention, it is reasonable to regard the input of the FFN as a query vector, and two linear layers of the FFN as keys and values, respectively. Similar observations are also described in (Geva et al., 2020).
3 Identifying Knowledge Neurons

Similar to (Geva et al., 2020), we view FFNs in Transformer as key-value memories as illustrated in Figure 2. We hypothesize that factual knowledge is stored in FFN memories and expressed by knowledge neurons. In this section, we propose a knowledge attribution method and a refining strategy to identify these knowledge neurons.

3.1 Knowledge Assessing Task

We employ the fill-in-the-blank cloze task to assess whether a pretrained model knows a fact. Following Petroni et al. (2019), each relational fact is in the form of a triplet ⟨h, r, t⟩, where h is the head entity, t is the tail entity, and r is the relation between them. Given a fact, pretrained models answer the cloze query x that expresses the fact but leaves the tail entity as a blank. For example, given the fact ⟨Ireland, capital, Dublin⟩, a possible query is “The capital of Ireland is __”. We also call the query a knowledge-expressing prompt. Petroni et al. (2019) describe that a model knows a fact if it can predict the correct answer. In this paper, rather than just examining the model outputs, we identify the specific knowledge neurons that express factual knowledge.

3.2 Knowledge Attribution

Inspired by Hao et al. (2021), we propose a knowledge attribution method based on integrated gradients (Sundararajan et al., 2017). Our method can evaluate the contribution of each neuron to knowledge predictions. In this paper, we examine FFN intermediate neurons for the masked token, where the answer is predicted.

Given an input prompt x, we first define the model output $P_x(\hat{w}_i^{(l)})$ as the probability of the correct answer predicted by a pretrained model:

$$P_x(\hat{w}_i^{(l)}) = p(g^* | x, w_i^{(l)} = \hat{w}_i^{(l)}),$$  \hspace{1cm} (4)

where $g^*$ denotes the correct answer; $w_i^{(l)}$ denotes the i-th intermediate neuron in the l-th FFN; $\hat{w}_i^{(l)}$ is a given constant that $w_i^{(l)}$ is assigned to.

In order to calculate the attribution score of a neuron $\text{Attr}(w_i^{(l)})$, we gradually change $w_i^{(l)}$ from 0 to its original value $\bar{w}_i^{(l)}$ calculated by the pretrained model, and meanwhile integrate the gradients:

$$\text{Attr}(w_i^{(l)}) = \bar{w}_i^{(l)} \int_0^1 \frac{\partial P_x(\alpha \bar{w}_i^{(l)})}{\partial w_i^{(l)}(\alpha)} \, d\alpha,$$  \hspace{1cm} (5)

where $\frac{\partial P_x(\alpha \bar{w}_i^{(l)})}{\partial w_i^{(l)}(\alpha)}$ calculates the gradient of the model output with regard to $w_i^{(l)}$. Intuitively, as $\alpha$ changes from 0 to 1, by integrating the gradients, $\text{Attr}(w_i^{(l)})$ accumulates the output probability change caused by the change of $w_i^{(l)}$. If the neuron has a great influence on the expression of a fact, the gradient will be salient, which in turn has large integration values. Therefore, the attribution score can measure the contribution of the neuron $w_i^{(l)}$ to the factual expressions.

Directly calculating continuous integrals is intractable. We instead use Riemann approximation $\text{Attr}(w_i^{(l)}) = \frac{\bar{w}_i^{(l)}}{m} \sum_{k=1}^{m} \frac{\partial P_x(\frac{k}{m} \bar{w}_i^{(l)})}{\partial w_i^{(l)}},$ where $m = 20$ is the number of approximation steps. With the attribution algorithm, we can identify a coarse set of knowledge neurons whose attribution scores are greater than a threshold $t$.

3.3 Knowledge Neuron Refining

In order to identify knowledge neurons more accurately, we further propose a refining strategy. Besides “true-positive” knowledge neurons that express factual knowledge, the coarse set of knowledge neurons may contain “false-positive” knowledge neurons that express other information (e.g., syntactic or lexical information). The refining strategy aims to filter out these “false-positive” neurons.

For different prompts corresponding to the same fact, we hypothesize that they share the same set of “true-positive” knowledge neurons, since they express the same factual knowledge. Meanwhile, we hypothesize that they do not share the “false-positive” knowledge neurons as long as the prompts are diverse enough. Therefore, given multiple diverse prompts, we can refine the coarse set of knowledge neurons by retaining only neurons that are widely shared among these prompts.

Specifically, given a relational fact, the complete process to identify its knowledge neurons is described as follows: (1) produce $n$ diverse prompts; (2) for each prompt, calculate the knowledge attribution scores of neurons; (3) for each prompt, retain the neurons with attribution scores greater than the attribution threshold $t$, obtaining the coarse set of knowledge neurons; (4) considering all the coarse sets together, retain the knowledge neurons shared by more than $p\%$ prompts.
4 Experiments

4.1 Experimental Settings

We conduct experiments for BERT-base-cased (Devlin et al., 2019), one of the most widely-used pretrained models. It contains 12 Transformer blocks, where the hidden size is 768 and the FFN inner hidden size is 3,072. Notice that our method is not limited to BERT and can be easily generalized to other pretrained models. For each prompt, we set the attribution threshold $t$ to 0.2 times the maximum attribution score. For each relation, we initialize the refining threshold $p\%$ (Section 3.3) as 0.7. Then, we increase or decrease it by 0.05 at a time until the average number of knowledge neurons lies in $[2, 5]$. We run our experiments on NVIDIA Tesla V100 GPUs. On average, it costs 13.3 seconds to identify knowledge neurons for a relational fact with 9 prompts.

4.2 Dataset

We examine knowledge neurons through the fill-in-the-blank cloze task based on the PARAREL dataset (Elazar et al., 2021). PARAREL is curated by experts, containing various prompt templates for 38 relations from the T-REx dataset (ElSahar et al., 2018). We show some example templates in Table 1. For each relational fact, we fill in the head entity in prompt templates and leave the tail entity as a blank to predict. In order to guarantee the template diversity, we filter out relations with fewer than 4 prompt templates and finally keep 34 relations, where each relation has 8.63 different prompt templates on average. These prompt templates produce 253,448 knowledge-expressing prompts in total for 27,738 relational facts.

4.3 Attribution Baseline

Our baseline method takes the neuron activation value as the attribution score, i.e., $\text{Attr}_{\text{base}}(w_l) = w_l^N$, which measures how sensitive a neuron is to the input. After computing attribution scores, we follow the same pipeline to obtain the refined knowledge neurons. For a fair comparison, we employ the same method to choose the hyperparameters $t$ and $p\%$ for the baseline to ensure the average number of knowledge neurons for each relation lies in $[2, 5]$.

The method based on neuron activation is a reasonable baseline. It is motivated by FFNs’s analogy with the self-attention mechanism (as described in Section 2), because self-attention scores are usually used as a strong attribution baseline (Kovaleva et al., 2019; Voita et al., 2019; Hao et al., 2021).

4.4 Statistics of Knowledge Neurons

Figure 3 presents the layer distribution of knowledge neurons identified by our method in each Transformer layer.

Table 2 shows statistics of knowledge neurons. $\cap$ denotes the intersection of knowledge neurons of fact pairs. “rel.” is the shorthand of relation. Our method identifies more exclusive knowledge neurons.
Their same order of magnitude guarantees the fairness of the subsequent comparisons in the paper.

We also compute the knowledge neuron intersection of different relational facts. Table 2 shows the average number of pair-wise knowledge neuron intersections. For our proposed method, (1) fact pairs with the same relation (intra-relation fact pairs) share 1.23 knowledge neurons on average; (2) fact pairs with different relations (inter-relation fact pairs) share almost no knowledge neurons. In contrast, for the baseline, (3) most identified neurons are shared by intra-relation fact pairs; (4) even a substantial portion of neurons are common for inter-relation fact pairs. The difference in knowledge neuron intersections suggests that our method can identify more exclusive knowledge neurons.

### 4.5 Knowledge Neurons Affect Knowledge Expression

We investigate how much knowledge neurons can affect knowledge expression in Figure 4 and Figure 5. Given a relational fact, we manipulate its knowledge neurons in two ways: (1) suppressing knowledge neurons by setting their activations to 0; (2) amplifying knowledge neurons by doubling their activations. Then, for each relation, we plot the average change ratio of the probability for the correct answer, corresponding to the manipulation.

For comparison, we also plot the results of manipulating baseline-identified knowledge neurons.

Figure 4 shows that suppressing knowledge neurons identified by our knowledge attribution method leads to a consistent decrease (29.03% on average) in the correct probability. By contrast, for baseline-identified neurons, the suppressing operation has a negligible influence (1.47% decrease on average) on the correct probability. Notably, for the relation P178 (developer), the correct probability abnormally increases by using the baseline.

As shown in Figure 5, we have similar observations for amplifying the knowledge neurons identified by our knowledge attribution. We see a consistent increase (31.17% on average) in the correct probability. By contrast, the baseline even decreases the average correct probability by 1.27%.

In summary, the knowledge neurons identified by our knowledge attribution method tend to notably affect knowledge expression. Notice that the above assessment is affected by the distribution of knowledge neurons. For example, if the knowledge neurons for a relation are distributed more widely, we need to manipulate more top-k neurons for better control. We use the above experiments as a proof of concept while leaving precise control for future work.
Table 3: Example relational facts along with their knowledge neurons, their top-2 and bottom-2 activating prompts, and the corresponding neuron activation. \(w_i^{(l)}\) denotes the \(i\)-th intermediate neuron at the \(l\)-th FFN. We fill the blank in each prompt with the correct answer for better readability. Owing to the page width, we show only key parts for overlong prompts. The top-2 activating prompts express exactly the relation, but the bottom-2 do not.

<table>
<thead>
<tr>
<th>Relational Facts</th>
<th>Neurons</th>
<th>Top-2 and Bottom-2 Activating Prompts (Average Activation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>({\text{Ireland, capital, Dublin}})</td>
<td>(w_{2141}^{(9)}, w_{1122}^{(10)})</td>
<td>Top: Our trip ... in \textbf{Dublin}, the capital and largest city of \textbf{Ireland} ... (6.36)</td>
</tr>
<tr>
<td>({\text{Cao_Yunding, place_of_birth, Shanghai}})</td>
<td>(w_{720}^{(10)}, w_{1885}^{(10)}, w_{2876}^{(11)})</td>
<td>Top: \textbf{Cao Yunding} was born in \textbf{Shanghai} in November 1989. (3.58) Full name: \textbf{Cao Yunding}. \textbf{Place of birth: Shanghai}, \textbf{China} ... (2.73)</td>
</tr>
<tr>
<td>({\text{Kuwait, continent, Asia}})</td>
<td>(w_{1461}^{(6)}, w_{880}^{(9)}, w_{1169}^{(10)})</td>
<td>Top: Kuwait is thus one of the smallest countries in \textbf{Asia} ... (6.63) Kuwait is a country in \textbf{Western Asia} ... (6.27)</td>
</tr>
</tbody>
</table>

Table 4: Average activation of knowledge neurons for three types of prompts. The activation of neurons identified by our method can distinguish the knowledge-expressing prompts (T1) clearly.

<table>
<thead>
<tr>
<th>Prompt Types</th>
<th>Ours</th>
<th>Baseline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Containing head and tail (T1)</td>
<td>0.485</td>
<td>2.472</td>
</tr>
<tr>
<td>Containing only head (T2)</td>
<td>0.019</td>
<td>2.312</td>
</tr>
<tr>
<td>Randomly sampled (T3)</td>
<td>-0.018</td>
<td>2.244</td>
</tr>
</tbody>
</table>

4.6 Knowledge Neurons are Activated by Knowledge-Expressing Prompts

In order to study what prompts can activate knowledge neurons, we compare the average activation of knowledge neurons for different types of prompts.

BINGREL Dataset We build a new dataset BINGREL by crawling the Bing search engine to collect new prompts, for a more extensive comparison beyond the PARAREL dataset. For each of the 27,738 facts in PARAREL, we crawl two types of texts: (1) up to ten texts containing both the head and the tail entities (210,217 texts crawled in total); (2) up to ten texts containing only the head entity without restricting tail entities (266,020 texts crawled in total). Following the distant supervision assumption (Mintz et al., 2009), the first type of texts tends to express the whole relational fact, while the second type does not. We mask tail entities for the first type of texts to obtain knowledge-expressing prompts (T1). In order to conduct a controlled experiment, we mask random words for the second type of texts, forming a control group (T2). Moreover, we employ randomly sampled prompts as another control group (T3).

Results As shown in Table 4, for our method, the identified knowledge neurons are more significantly activated by knowledge-expressing prompts \((T_1 = 0.485)\), compared with the control groups \((T_2 = 0.019, T_3 = -0.018)\). By contrast, for the baseline, the activation of identified neurons cannot distinguish three types of prompts. In addition, since our comparison is based on the web-crawled BINGREL dataset, we validate the generalization of knowledge neurons to open-domain texts that are unseen in PARAREL.

Example Prompts In Table 3, we present example prompts that activate knowledge neurons the most and the least, respectively. Given a fact, we first identify its knowledge neurons with our knowledge attribution method. Then, we calculate the average activation of knowledge neurons for each crawled prompt that contains both the head and the tail entities in BINGREL. Finally, we demonstrate two prompts with the highest average activation values and two with the lowest (denoted as top-2 and bottom-2 activating prompts, respectively).

As shown in Table 3, the top-2 activating prompts express exactly the corresponding relational fact. In contrast, despite containing the same head and tail entities, the bottom-2 activating prompts do not express the correct relation. For example, although the bottom-2 activating prompts for \(\{\text{Ireland, capital, Dublin}\}\) express
We use the case studies as a proof of concept while we present two preliminary studies to demonstrate (i.e., the second linear layer of FFNs; see Figure 2): $\text{FFN}_i^{(\text{val})} = \text{FFN}_i^{(\text{val})} - \lambda_1 t + \lambda_2 t'$, where $\text{FFN}_i^{(\text{val})}$ denotes the value slot corresponding to the $i$-th knowledge neuron; $t$ and $t'$ are the word embeddings of $t$ and $t'$, respectively; $\lambda_1$ and $\lambda_2$ are set to 1 and 8 in our experiments.

### 5 Case Studies

We present two preliminary studies to demonstrate the potential applications of knowledge neurons. We use the case studies as a proof of concept while leaving precise fact editing for future work.

#### 5.1 Updating Facts

By leveraging knowledge neurons in pretrained models, we try to update a learned relational fact from $(h, r, t)$ to $(h, r, t')$.

**Methods** First, we identify the knowledge neurons of $(h, r, t)$. Then, we retain the knowledge neurons that are shared by less than 10% of intra-relation facts, to reduce the influence on other facts with the same relation. Finally, we directly modify the corresponding value slots in $\text{FFN}_i^{(\text{val})}$ (i.e., the second linear layer of FFNs; see Figure 2): $\text{FFN}_i^{(\text{val})} = \text{FFN}_i^{(\text{val})} - \lambda_1 t + \lambda_2 t'$, where $\text{FFN}_i^{(\text{val})}$ denotes the value slot corresponding to the $i$-th knowledge neuron; $t$ and $t'$ are the word embeddings of $t$ and $t'$, respectively; $\lambda_1$ and $\lambda_2$ are set to 1 and 8 in our experiments.

### Table 5: Case studies of erasing relations. The influence on knowledge expression is measured by the perplexity change. The knowledge erasing operation significantly affects the erased relation, and has just a moderate influence on the expression of other knowledge.

<table>
<thead>
<tr>
<th>Erased Relations</th>
<th>Perplexity (Erased Relation)</th>
<th>Perplexity (Other Relations)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Before Erasing</td>
<td>After Erasing</td>
</tr>
<tr>
<td>P19 (place_of_birth)</td>
<td>1450.0</td>
<td>2996.0 (+106.6%)</td>
</tr>
<tr>
<td>P27 (country_of_citizenship)</td>
<td>28.0</td>
<td>38.3 (+36.7%)</td>
</tr>
<tr>
<td>P106 (occupation)</td>
<td>2279.0</td>
<td>5202.0 (+128.2%)</td>
</tr>
<tr>
<td>P937 (work_location)</td>
<td>58.0</td>
<td>140.0 (+141.2%)</td>
</tr>
</tbody>
</table>

**Evaluation Metrics** We report two metrics to evaluate the fact updating: (1) change rate, the ratio that the original prediction $t$ is modified to another; (2) success rate, the ratio that $t'$ becomes the top prediction. In addition, we measure the influence on other knowledge by the following two metrics: (1) Δintra-relation PPL, the increase of perplexity on the prompts with the same relation $r$; (2) Δinter-relation PPL, the increase of perplexity on the prompts with different relations.

### Table 6: Case studies of updating facts. ↑ means the higher the better, and ↓ means the lower the better. “rel.” is the shorthand of relation. Keeping a moderate personal information like “Dublin is a city in Ireland”, they do not reflect the capital relation. The examples support again that knowledge neurons are activated by corresponding knowledge-expressing prompts.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Knowledge Neurons</th>
<th>Random Neurons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change rate↑</td>
<td>48.5%</td>
<td>4.7%</td>
</tr>
<tr>
<td>Success rate↑</td>
<td>34.4%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Δintra-rel. PPL↓</td>
<td>8.4</td>
<td>10.1</td>
</tr>
<tr>
<td>Δinter-rel. PPL↓</td>
<td>7.2</td>
<td>4.3</td>
</tr>
</tbody>
</table>

**Setup** We conduct experiments on PARAREL. For each relation, we randomly sample ten facts learned by the pretrained model. For each fact $(h, r, t)$, we randomly choose a different entity $t'$ with the same type as $t$ (e.g., both $t$ and $t'$ belong to city), and then update $t'$ as the target entity. We only manipulate about four top knowledge neurons as in Section 4.4. For reference purposes, we also perform the same update process on the same number of random neurons.

**Results** As shown in Table 6, the surgery of knowledge neurons achieves a nontrivial success rate for updating facts, while random neurons are insufficient. Moreover, we find that such manipulation has little negative influence on other knowledge predictions. It is promising that we can change very few (i.e., about four in the above experiments) neurons to affect certain facts in pretrained Transformers. We can further improve the success rate by including more top knowledge neurons in the update process.
Methods  Given a relation \( r \), we first identify knowledge neurons for all relational facts with \( r \). Then, we retain 20 knowledge neurons that appear most frequently among these facts. Finally, we set the value slots in FFN\(^{\text{val}}\) (see Figure 2) corresponding to these knowledge neurons to 0, i.e., zero vectors.

Results  As shown in Table 5, we report model perplexity before and after knowledge erasing. With the erasing operation, the perplexity of the removed knowledge increases as expected. Moreover, the model perplexity of other relations remains similar. We argue that knowledge neurons provide a promising way to erase undesired knowledge with minimal efforts.

6 Related Work

Probing Knowledge in Pretrained Models

Many pieces of previous work aim to measure knowledge stored in pretrained models. Petroni et al. (2019) propose to retrieve knowledge in pretrained models (such as BERT) using cloze queries. Their experiments show that BERT has a strong ability to recall factual knowledge without any fine-tuning. Jiang et al. (2020b) improve the cloze queries with mining-based and paraphrasing-based methods. Roberts et al. (2020) propose the closed-book question answering to measure how much knowledge a pretrained model has stored in its parameters. Elazar et al. (2021) measure and improve the consistency of pretrained models with respect to factual knowledge prediction. Rather than examining only the model outputs, we provide an open-the-black-box analysis for the knowledge neurons in pretrained Transformers.

Attribution Methods  In order to open the black boxes of deep learning models, attribution methods aim to attribute the model output to input features using different measures. The product of the gradients (of the output with respect to input features) and feature values is a reasonable baseline (Bahrens et al., 2010; Simonyan et al., 2014). Besides, a set of attribution methods (Shrikumar et al., 2017; Binder et al., 2016; Zeiler and Fergus, 2014; Springenberg et al., 2015) back-propagate the final output to input features. However, as stated by Sundararajan et al. (2017), none of these methods can simultaneously satisfy sensitivity and implementation invariance, two fundamental axioms. Taking the axioms as guidance, Sundararajan et al. (2017) propose the integrated gradient method. Our knowledge attribution method is built upon integrated gradients.

Analysis of Transformer  As one of the most popular and effective NLP architectures, Transformer (Vaswani et al., 2017) has attracted extensive studies. Most previous work focuses on the self-attention module (Voita et al., 2019; Clark et al., 2019; Vig and Belinkov, 2019; Hao et al., 2021). Recently, Wu et al. (2019) and Dong et al. (2021) have pointed out that the feed-forward network module also matters to Transformer. Geva et al. (2020) attempt to connect feed-forward networks with key-value memories by qualitative analysis. In this paper, we identify and analyze knowledge neurons in feed-forward networks for given factual knowledge. Moreover, we present how to leverage knowledge neurons to explicitly edit factual knowledge stored in pretrained Transformers.

7 Conclusion and Future Directions

We propose an attribution method to identify knowledge neurons that express factual knowledge in pretrained Transformers. We find that suppressing or amplifying the activation of knowledge neurons can accordingly affect the strength of knowledge expression. Moreover, quantitative and qualitative analysis on open-domain texts shows that knowledge neurons tend to be activated by the corresponding knowledge-expressing prompts. In addition, we present two preliminary case studies that attempt to utilize knowledge neurons to update or erase knowledge in pretrained Transformers.

Despite the effectiveness of identifying knowledge neurons, our current studies still have limitations. First, we examine knowledge neurons based on the fill-in-the-blank cloze task, while knowledge can be expressed in a more implicit way. It is an open question whether Transformer can utilize stored knowledge in a generalized way, such as for reasoning. The interactions between knowledge neurons also remain under explored. Second, we focus on factual knowledge for ease of evaluation, even though our method is also applicable for other types of knowledge. Third, we use the single-word blank in cloze queries for simplicity, which requires multi-word extensions (Jiang et al., 2020a). Besides, an interesting future direction is to figure out how knowledge neurons work in multilingual pretrained Transformers (Conneau and Lample, 2019; Conneau et al., 2020; Chi et al., 2021).
References


Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. 2021. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. *CoRR*, abs/2103.03404.


