Enhancing Semantic Consistency of Large Language Models through Model Editing: An Interpretability-Oriented Approach

Anonymous ACL submission

Abstract

A Large Language Model (LLM) tends to generate inconsistent and sometimes contradictory outputs when presented with a prompt that has equivalent semantics but is expressed differently from the original prompt. To achieve semantic consistency of an LLM, one of the key approaches is to finetune the model with prompt-output pairs with semantically equivalent meanings. Despite its effectiveness, a data-driven finetuning method incurs substantial computation costs in data preparation and model optimization. In this regime, an LLM is treated as a "black box", restricting our ability to gain deeper insights into its internal mechanism. In this paper, we are motivated to enhance the semantic consistency of LLMs through a more interpretable method (i.e., model editing) to this end. We first identify the model components (i.e., attention heads) that have a key impact on the semantic consistency of an LLM. We subsequently inject biases into the output of these model components along the semantic-consistency activation direction. It is noteworthy that these modifications are cost-effective, without reliance on mass manipulations of the original model parameters. Through comprehensive experiments on the constructed NLU and open-source NLG datasets, our method demonstrates significant improvements in the semantic consistency and task performance of LLMs. Additionally, our method exhibits promising generalization capabilities by performing well on tasks beyond the primary tasks.

1 Introduction

The field of Natural Language Processing (NLP) is experiencing a paradigm shift with the advent of Large Language Models (LLMs). These models have demonstrated remarkable capabilities in various tasks such as sentiment classification (Wang et al., 2023), machine translation (Hendy et al., 2023), and summarization (Pu et al., 2023). However, an LLM tends to generate inconsistent and sometimes contradictory outputs when presented with a prompt that has equivalent semantics but is expressed differently from the original prompt. Such behavior is referred to as the issue of "semantic consistency" (Gan and Mori, 2023; Rabinovich et al., 2023; Raj et al., 2022), largely limiting the application of LLMs to real-world scenarios. For specific instances of inconsistency and consistency, please refer to Figure 1.

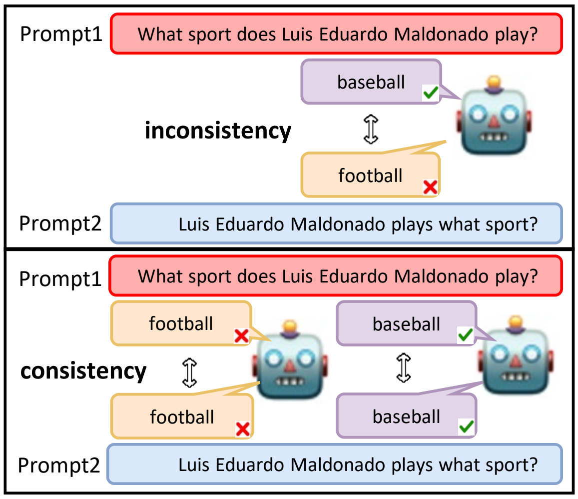


Figure 1: Inconsistency arises when prompts sharing equivalent semantics produce different outcomes, while consistency is achieved when their outputs remain consistently identical, irrespective of their accuracy.

Current mainstream solutions involve prompt engineering or data-driven methods to handle the problem of semantic consistency. For example, Raj et al. (2023) proposed a prompt strategy called 'Ask-to-Choose' (A2C) to improve the semantic consistency of LLMs, but this method requires carefully designed prompts. Applying a data-driven supervised fine-tuning method (SFT) (Ouyang et al., 2022) to finetune an LLM with prompt-output pairs with semantically equivalent meanings is another effective approach. Despite their effectiveness, these methods incur substantial computation costs in data preparation and model optimization. Furthermore, these methods treat an LLM as a "black box", restricting our ability to gain deeper insights into its underlying causes of the semantic consistency problem.

To address the limitations of previous methods to enhance the semantic consistency of LLMs, we propose a method based on model editing that can locate the internal model components (i.e., attention heads) responsible for generating semantic inconsistency. We subsequently inject biases into the outputs of these model components along semantic consistency activation directions. This strategy aims to shift the outputs of the key model components toward a direction resilient to variations of synonymous prompts.

In order to comprehensively evaluate our proposed method under varying prompts, we have constructed relevant NLU-task datasets in addition to utilizing existing evaluation datasets for NLG-task. We leveraged the paraphrasing capability of GPT-4[1] to construct the RobustSST2, RobustMRPC, and RobustBOOLQ datasets. These datasets cover a wide range of tasks, including the sentiment classification dataset SST2 (Socher et al., 2013), the text similarity dataset MRPC (Dolan and Brockett, 2005), and the question-answering dataset BOOLQ (Clark et al., 2019).

Our method has shown significant enhancements in both semantic consistency and task performance on publicly available NLG datasets and our constructed NLU datasets. Furthermore, our method also achieved positive results in out-of-domain experiments, demonstrating a solid generalization capability. In summary, our contributions are two-fold:

- To the best of our knowledge, we are the first to use a model editing approach to address the issue of prompt semantic inconsistency. Through this interpretability-oriented method, we can precisely diagnose the internal components contributing to semantic consistency. By directly injecting biases into the model, our method avoids mass-manipulating model parameters, resulting in a significant saving in GPU hour (around 18 times faster) in a typical task compared to a traditional SFT approach.
- We have curated three datasets, designed to address the absence of NLU semantic consistency evaluation benchmark. The datasets will be released to the community to foster research along this line.

[1]https://platform.openai.com/docs/api-reference/chat

2 Related Work

Semantic Consistency. The study of semantic consistency originated from investigations into Masked Language Models (MLMs) like BERT and Roberta. Elazar et al. (2021) revealed significant semantic inconsistency in the factual information extracted from these MLMs when subjected to paraphrasing. Building on this, Fierro and Søgaard (2022) extended the examination of semantic consistency to a multilingual context, finding that inconsistency issues are not confined to English but are prevalent across various other languages. Despite the significant shift in the Natural Language Processing (NLP) paradigm instigated by Large Language Models (LLMs) (Brown et al., 2020), the issue of semantic inconsistency remains (Gan and Mori, 2023). Rabinovich et al. (2023) developed a benchmark dataset of high-quality paraphrases specifically for factual questions, serving as a testbed for evaluating semantic consistency in a QA context. Existing methods mainly address this issue through prompt engineering and data-driven SFT. For example, Raj et al. (2023) proposed an Ask-to-Choose (A2C) prompting method that can enhance both accuracy and semantic consistency in LLMs. Zhou et al. used an unsupervised finetuning method. They took advantage of the fact that multiple prompts can be used to specify a single task and proposed to regularize prompt consistency, encouraging consistent predictions across this diverse set of prompts. Compared to previous methods, we use a model editing method to modify the output of specific model components in an LLM. This method is both transparent and computationally lightweight.

Model Editing. The goal of model editing is to modify specific knowledge or control model behaviors without affecting the model's performance on other tasks (Yao et al., 2023). There are mainly three types of editing methods: external memory-based methods, constrained fine-tuning methods, and locate-then-edit methods.

Among them, (1) External memory-based methods use new parameters to update knowledge or change model behavior. An example is SERAC (Mitchell et al., 2022), which uses edit memory to

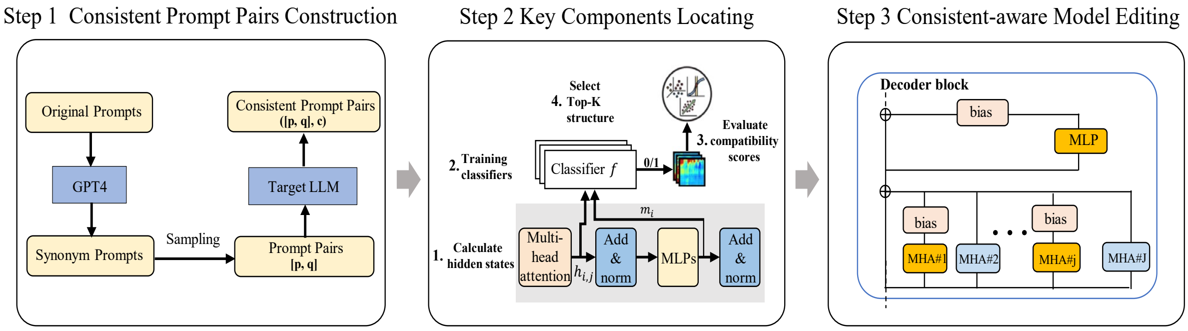


Figure 2: The flowchart of our method. Our method has three main steps: (1) We first construct the prompt pairs $[p, q]$ with consistency evaluation label c. (2) Based on these pairs, we perform key-components locating, which selects the top-K (accuracy) components by training and evaluating classifiers based on these components' output hidden states and related consistency evaluation labels. If a classifier has high accuracy, the component and LLM will behave very similarly (compatible), which suggests that the component is highly likely to be responsible for the inconsistency errors, as mentioned previously. (3) For the selected top-K components, we add biases to the hidden states of these components, which will shift the original activations of these components toward more semantically consistent directions.

store updated knowledge and a classifier to route between the edit memory and the pre-trained model. (2) Constrained fine-tuning methods typically involve specific fine-tuning restrictions to regulate parameter updates, thus maintaining the model's performance on unedited knowledge. For example, the method proposed by (Zhu et al., 2020) updates all model parameters but restricts the norm of parameters before and after updating to preserve old knowledge. (3) Locate-then-edit methods try to first identify relevant model weights or representations that store knowledge or steer model behavior, and then edit these weights or representations to achieve desirable outputs. Meng et al. (2022) used causal analysis to find that factual knowledge is mainly stored in the intermediate MLP layer weights and then used rank-one editing to modify model weights related to factual knowledge. Li et al. (2023) demonstrated that by identifying specific attention heads and editing their activations, the likelihood of the model producing truthful output can be significantly enhanced. We adopt the "locate-then-edit" paradigm, with the main motivation being that we not only want to improve the semantic consistency of the model but also want to analyze which components in an LLM are related to this consistency.

3 Preliminary

LLM representation. The currently prevalent LLM structure adopts the decoder-only paradigm. According to Elhage et al. (2021), this type of LLM mainly consists of three parts: Token embedding, a sequence of decoder blocks, and token unembedding. Among them, token embedding is the process of mapping a token index to an embedding vector, while token unembedding is the reverse operation that maps the embedding back to the probability space of tokens, and then samples to obtain the index of the next token. The vast majority of parameters in LLM are composed of stacked decoder blocks, with each decoder block consisting of the components of multi-head attention and MLP, which can be represented by:

$$a_i = x_i + \sum_{j=1,\dots,J} h_{i,j} \tag{1}$$

$$x_{i+1} = a_i + m_i, \tag{2}$$

where x_i is the i-th decoder layer hidden states. $h_{i,j}$ is the hidden output of the j-th attention head in the i-th layer. a_i is the residual output after multi-head attention. m_i is the i-th MLP layer output. x_{i+1} is the hidden output of the i-th decoder block and also the input of the $i+1$-th decoder block.

4 Methodology

We use GPT-4 to construct prompt pairs that have the same semantics, and the target LLM outputs of these prompt pairs should ideally be consistent. However, when we use the target LLM to predict these prompt pairs, we obtain both consistent and inconsistent results. These inconsistent results are errors made by the target LLM. To locate the sources of these errors, we assume that if the model components (i.e. attention heads) behave similarly to the target LLM, then these components are actually the causes of the semantic consistency problems in the target LLM. On the other hand, those components that have large behavioral differences from the target LLM indicate that they are less relevant to the semantic consistency problems of the prompt pairs. Based on this assumption, we use the linear probing technique (Alain and Bengio, 2016) to identify the relevant components.

Next, we add semantic consistency biases to the identified components to correct their erroneous behavior. These biases are obtained by calculating the difference between the mass mean of the consistency samples and the mass mean of all samples on the corresponding components. These biases will shift the original activations of these components toward more semantically consistent directions.

As shown in Figure 2, our method mainly consists of three steps, which are consistent prompt pairs construction, key components locating, and consistent-aware editing respectively. We will provide detailed explanations of these steps in the following sections.

4.1 Consistent Prompt Pairs Construction

We need to construct consistent prompt pairs for locating and editing an LLM. Specifically, we first construct a prompt pair set $\mathcal{D}$, whose element is represented as $[p, q]$. Here, p represents the input prompt, and q is a synonym or rephrased version of p, which can be generated using existing large-scale models like GPT-4.

Based on $\mathcal{D}$, we need the consistency evaluation label c from the target LLM for key components locating and editing. So we augment the consistency evaluation label c to $[p, q]$ forming $([p, q], c)$. In the case of NLU tasks, we determine consistency labels based on whether the predicted results are the same. For NLG tasks, we can utilize GPT-4 to assess the consistency. Subsequently, we add c to each prompt pair in $\mathcal{D}$, obtaining the set $\mathcal{D}'$

4.2 Key Components Locating

With the constructed prompt pairs, we use linear probing (Alain and Bengio, 2016) to identify which components have similar behavior to the LLM that determine the prompts' semantic consistency. Specifically, we divide the dataset $\mathcal{D}'$ into probe set $\mathcal{D}'_{probe}$ and locate set $\mathcal{D}'_{locate}$ following a 4:1 ratio.

For each component, either an attention head or an MLP in any layer, we train a classifier that takes the concatenated hidden states as input and uses the consistency label c as the ground truth label. These hidden states are the output hidden states of the component with respect to p and q. The training data for this classifier comes from $\mathcal{D}'_{probe}$, and the testing data for this classifier comes from $\mathcal{D}'_{locate}$.

If the classifier achieves a high score on the locate set $\mathcal{D}'_{locate}$, it implies that the component and the overall LLM behave very similarly. On the other hand, a low score indicates that this component is less important for semantic consistency problems. We locate the top K components by ordering the classification accuracy.

More specifically, given a sample $([p, q], c)$, the linear classifier training feature for candidate MLP and attention head are $f(m_i, p, q)$ and $f(h_{i,j}, p, q)$, respectively.

$$f(m_i, p, q) = [m_i^{p^{last}}; m_i^{q^{last}}], \tag{3}$$

$$f(h_{i,j}, p, q) = [h_{i,j}^{p^{last}};\ h_{i,j}^{q^{last}})], \tag{4}$$

where p^{last} and q^{last} indicates the last token of p and q, and $m_i^{p^{last}}$ is the hidden output of the MLP layer in the i-th decoder block and $h_{i,j}^{p^{last}}$ is the hidden output of the j-th attention head in the i-th decoder block, all correspond to the last token. The reason why we only use the last token of p and q is that for a decoder-only architecture, the last token has visibility over all preceding tokens. Therefore, the hidden states corresponding to the last token can be considered a summary representation of the entire prompt. In this manner, we can construct training sets $\mathcal{S}(m_i)$ and $\mathcal{S}(h_{i,j})$ for training linear classifiers for m_i and $h_{i,j}$, respectively.

$$\mathcal{S}(m_i) = \{f(m_i, p, q), c\}_{([p,q],c)\in\mathcal{D}'_{probe}} \tag{5}$$

$$\mathcal{S}(h_{i,j}) = \{f(h_{i,j}, p, q), c\}_{([p,q],c)\in\mathcal{D}'_{probe}}, \tag{6}$$

where $\mathcal{S}(m_i)$ and $\mathcal{S}(h_{i,j})$ are mapped from $\mathcal{D}'_{probe}$.

We train linear classifiers with $\mathcal{S}(m_i)$ or $\mathcal{S}(h_{i,j})$, and then evaluate these classifiers on $\mathcal{D}'_{locate}$. The top K components in LLM with the highest classification accuracy are used for model editing, as these components strongly affect the prompts' semantic consistency.

4.3 Consistent-aware Model Editing

Inspired by the work from (Li et al., 2023; Jorgensen et al., 2023), we make specific adjustments to the hidden states of the top-K components, aligning their hidden states toward greater semantic consistency.

Specifically, we add biases to these components, and the biases are obtained by calculating the difference between the mass mean of the consistency samples and the mass mean of all samples on the corresponding components. All of these samples are from $\mathcal{D}'_{probe}$. Formally, the biases for the candidate MLP and attention head are calculated by:

$$b(m_i)=\sum_{p,c=1}\frac{m_i^{p^{last}}}{N}-\sum_{p}\frac{m_i^{p^{last}}}{M},$$
$$b(h_{i,j})=\sum_{p,c=1}\frac{h_{i,j}^{p^{last}}}{N}-\sum_{p}\frac{h_{i,j}^{p^{last}}}{M}, \tag{7}$$

where N is the number of the prompts in $\mathcal{D}'_{probe}$ with $c = 1$, and M is the number of all the instances in $\mathcal{D}'_{probe}$. After that, these biases are added to the hidden states of the selected Top-K components, obtaining $\hat{m}_i$ and $\hat{h}_{i,j}$ for the K selected components.

$$\hat{m}_i = m_i + \alpha \cdot b(m_i) \tag{8}$$
$$\hat{h}_{i,j} = h_{i,j} + \alpha \cdot b(h_{i,j}). \tag{9}$$

Here, α is the hyperparameter that adjusts the strength of the activations shift.

5 NLU Benchmark Construction

Currently, there are some NLG benchmarks related to the semantic consistency of LLM (Rabinovich et al., 2023). However, there is a relative scarcity of NLU benchmarks specifically designed for semantic consistency research. To address this gap, we propose a benchmark dataset for evaluating semantic consistency in NLU tasks. This benchmark comprises RobustSST2, RobustMRPC, and RobustBOOLQ, which are derived from the sentiment classification dataset SST2 (Socher et al., 2013), the text similarity dataset MRPC (Dolan and Brockett, 2005), and the yes/no question-answering dataset BOOLQ (Clark et al., 2019), respectively. Our primary objective is to assess the semantic consistency of LLMs under synonymous task instructions for these datasets.

More specifically, we first generate 30 synonymous task instructions for each task dataset. For example, we feed the following prompt (bold font) to GPT-4 to generate the synonymous task instructions used for RobustSST2.

Rephrase the following sentence in 30 ways, while retaining the same meaning.

```
Measure the polarity of this sentence and
respond with either 'positive' or 'negative',
give me one word.
```

Then, we slice the generated task instructions according to an 8:2 ratio, meaning the training set uses 24 instructions, while the test set uses 6 instructions.

For the training set, we constructed 24 synonymous prompts for each training instance, *i.e.*, $\text{prompt}_i = [\text{instruction}_i, \text{instance}_{train}]_{i=1}^{24}$, and each prompt_i has a label answer_i, which is consistent across these 24 prompts.

Additionally, to create consistent prompt pairs for model editing, we generated C_{24}^2 prompt pairs by iterating through all possible combinations of these 24 prompts for each training instance. Then, we utilized the target LLM to assess whether the predictions generated for each prompt pair were consistent and obtain the relevant consistency evaluation label c. Lastly, we sampled 250 instances from both the $c = 0$ and $c = 1$ categories, yielding a total of 500 instances used for model editing.

The instance in the test set is different from the instance in the training set. Each instance in the test set includes a constructed prompt along with its corresponding answer $[\text{prompt}_{test}, \text{answer}_{test}]$. In specific, we first use the left 6 task instruction to construct relevant prompts *i.e.*, $\{\text{prompt}_i = [\text{instruction}_i, \text{instance}_{test}]\}_{i=25}^{30}$. Subsequently, We perform sample selection for these 6 prompts to construct a test instance. The selection rule is that if these 6 prompts yield the same result, we randomly choose 1 prompt to construct the sample. However, if they predict N different outcomes, we select 1 prompt from each of the N distinct results from N samples. By employing this approach, we can select hard negative samples while retaining examples that the LLM could originally predict correctly, thereby enhancing the diversity of our test data.

6 Experiments

6.1 Datasets

To verify the effectiveness of the model editing method for addressing the issue of prompt semantic consistency, we conducted tests on both NLU and

NLG tasks. For NLU evaluation, we utilized the specially constructed RobustMRPC, RobustSST2, and RobustBOOLQ datasets. For NLG evaluation, we selected the sport and capital categories from the PopQA question-answering dataset as described by (Rabinovich et al., 2023). Detailed data statistics are shown in Table 1.

Task Category	Datasets	Number of test cases
NLU	RobustMRPC	408
NLU	RobustSST2	872
NLU	RobustBOOLQ	1000
NLG	PopQA_sport	3829
NLG	PopQA_capital	4515

Table 1: Data statistics of Evaluation Datasets.

6.2 Evaluation Metrics

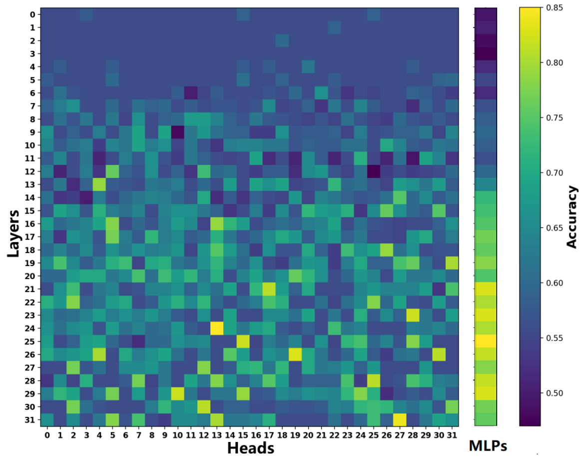
For NLU tasks, we evaluate its task performance by testing the overall accuracy of classification results across different instruction templates. To assess semantic consistency, we measure the standard deviation of the accuracy across these various instruction templates. For the NLG task, accuracy and mean pairwise cosine similarity metrics introduced by (Rabinovich et al., 2023) are employed to evaluate the model's task performance and its semantic consistency respectively. It is worth noting that lower standard deviation and higher mean pairwise cosine similarity are both indicative of better semantic consistency.

6.3 Key Components Locating Result

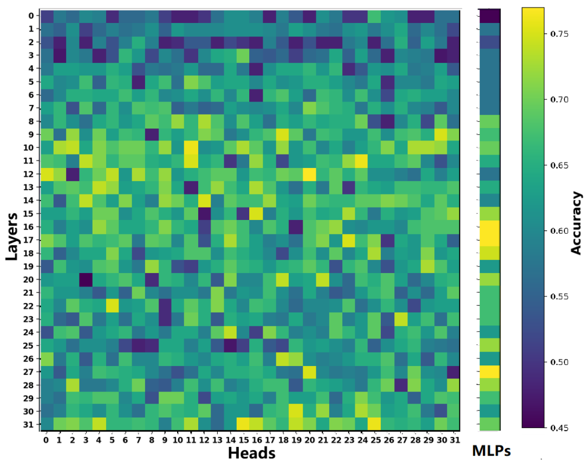
We utilize the LLama2-7B chat-version model (Touvron et al., 2023) as the target LLM to analyze the impact of candidate model components, such as attention heads and MLPs, on LLM's semantic consistency problem. Next, we visualize the locating results of these components, with brighter squares (i.e. yellow squares) highlighting areas of high locating accuracy, indicative of a strong correlation with semantic consistency.

As the visualization result shown in Figure 3, we find that there exists a notable concentration of yellow squares between layers 11 and 32, suggesting that attention heads and MLPs in the mid to final LLM's decoder blocks are highly relevant to semantic consistency.

Furthermore, our findings suggest that model components in the initial layers of transformer blocks exert negligible influence on semantic consistency. Their locating accuracy for synonymous samples hovers around 50%, equivalent to random chance, indicating these samples are treated as identical by these components. This distinction underscores the nuanced role of the model components across different layers in influencing LLM's semantic consistency.



(a) Visualization Result on RobustSST2.



(b) Visualization Result on PopQA_capital.

Figure 3: The visualization experiments on the RobustSST2 (NLU) and PopQA_capital (NLG) dataset. The horizontal axis represents the attention heads and the MLP in certain layer, while the vertical axis indicates the layer number. The column on the right shows the locating accuracy of attention heads or the MLPs. Brighter Squares indicate high locating accuracy.

6.4 Model Editing Experimental Result

Our comprehensive analysis, as presented in Table 2, and Table 3, demonstrates that our editing method can significantly enhance both semantic consistency and task performance across a variety of NLU and NLG tasks. Specifically, we observed notable reductions in the standard deviation for semantic consistency assessments on the RobustM-

Method	RobustMRPC	RobustSST2	RobustBOOLQ
LLama2-7B	$67.15_{\pm 5.36}$	$85.66_{\pm 4.88}$	$46.40_{\pm 10.55}$
+Editing	$68.62_{\pm 4.47}$	$89.90_{\pm 4.54}$	$57.50_{\pm 5.10}$

Table 2: Main experiment result is on NLU datasets. The notation $67.15_{\pm 5.36}$ indicates an average test set accuracy of 67.15 with a standard deviation of ± 5.36.

Method	PopQA_sport	PopQA_capital
LLama2-7B	$50.83_{/0.79}$	$73.33_{/0.73}$
+Editing	$53.20_{/0.80}$	$74.36_{/0.77}$

Table 3: Main experiment on NLG Tasks. The notation $50.83_{/0.79}$ indicates an average test set accuracy of 50.83 with a mean pairwise cosine similarity of 0.79.

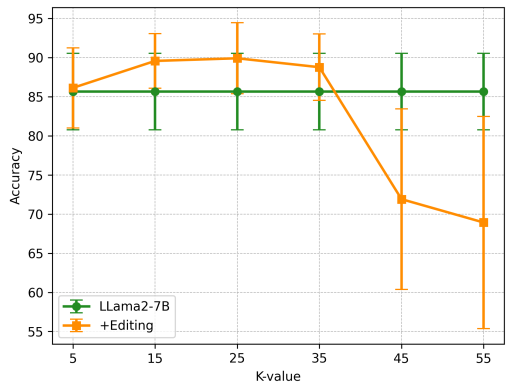


Figure 4: The performance of different K-values.

RPC, RobustSST2, and RobustBOOLQ datasets, with decreases of 0.89, 0.34, and 5.45, respectively. Moreover, the accuracy of model performance experienced substantial improvements, showing increases of 1.47%, 4.24%, and 11.1% across these datasets, respectively. For NLG tasks, we noted improvements in semantic consistency score by 1.0% and 4.0%, respectively, while the accuracy in these tasks rose by 2.37% and 1.03%, respectively.

The findings from these experiments clearly support the conclusion that adjustments to the outputs of the top-K model components can significantly enhance the model's semantic consistency and task performance. Importantly, these advancements are achieved without the need for altering the model's underlying parameters.

6.5 Ablation Study

6.5.1 The Influence of Hyperparameter Top-K

We investigate the impact of the K-value on the experimental setting, selecting the RobustSST2 dataset for our analysis, with the $k \in \{5, 15, 25, 35, 45, 55\}$. As demonstrated in Figure 4, it is observed that the edited model achieves the highest accuracy when the K-value is equal to 25. Conversely, when the K-value is equal to or greater than 35, a decline in model accuracy is noted. These experimental findings underscore the critical importance of selecting an appropriate number of editing heads. Excessive model editing can result in its collapse.

6.5.2 The Influence of the Model Components Selecting Strategy and Editing Direction

To validate the effectiveness of our located model components and editing directions, we carried out two ablation studies. The first one is to randomly pick a number of model components equivalent to our editing method, with the goal of evaluating the effectiveness of our components selection strategy based on locating accuracy. The second study involved altering the editing directions to random directions based on a normal distribution.

Method	RobustMRPC
LLama2-7B	$67.15_{\pm 5.36}$
+Editing	$68.62_{\pm 4.47}$
w/ random components	$61.51_{\pm 10.86}$
w/ random direction	$64.46_{\pm 6.20}$

Table 4: Ablation studies for the influence of model components selecting strategy and effectiveness of the editing direction. "random components" indicates the strategy of randomly selecting an equivalent number of components as our method employs. The "random direction" is the approach of randomly selecting editing directions.

Table 4 demonstrates that on the RobustMRPC dataset, both semantic consistency and the task performance of the model suffer when random model components or random direction editing are applied. Compared to the unedited LLama2-7B chat-version model, semantic consistency experiences a decline of 5.5 and 0.84 points, while accuracy drops by 5.64% and 2.69%, respectively. These results highlight the critical role of specific model components and editing direction in enhancing model semantic consistency and task performance.

6.5.3 Out-of-domain Experiment Result

We evaluate the performance of the edited LLama2-7B chat-version model on out-of-domain datasets. Specifically, after editing the model on the MRPC dataset, we test its performance on four OOD

Model	AG News	IMDB
LLama2-7B	70.00	88.60
+Editing	70.20	89.40

Table 5: Evaluation of OOD performance on AG News and IMDB datasets using a subset of 500 instances from each.

Model	CNN/Daily Mail	XSum
LLama2-7B	21.36	14.28
+Editing	21.14	14.45

Table 6: Experiment results on OOD performance with 500 instances from CNN/Daily Mail and XSum.

datasets: AG News for news categorization (Zhang et al., 2015), IMDB for movie reviews sentiment classification (Maas et al., 2011), and both CNN/Daily Mail (See et al., 2017) and XSum (Narayan et al., 2018) for news summarization, drawing a sample of 500 instances from each for evaluation. For AG News and IMDB, accuracy serves as the evaluation metric, while for CNN/Daily Mail and XSum, we apply the ROUGE-L metric (Lin, 2004) for assessment. According to Table 5 and Table 6, the results indicate that the model's performance remains consistent across most datasets, with a slight increase on the IMDB dataset. This suggests that the editing method not only achieves significant improvements in targeted tasks but also maintains performance across OOD tasks.

6.5.4 Comparison with the SFT Method

To compare our method with the STF approach, we also employ the LLama2-7B-Chat model as the base model. Specifically, we generate training samples with various expressions that have the same semantic meaning as the fine-tuning data. The number of training samples is identical to that of our model editing method (500 for each relevant task).

As the experiment result shown in Tables 7 and 8, while our method enhances both semantic consistency and task performance, the magnitude of improvement is not as pronounced as that achieved by SFT. Notably, SFT outperforms our editing method on RobustMRPC, RobustSST2, RobustBOOLQ, and PopQA_sport. The only exception is the NLG task's PopQA_capital dataset, where our method slightly surpasses SFT (74.36 vs. 70.89). SFT achieves superior performance by precisely adjusting model parameters using backpropagation (BP).

Method	RobustMRPC	RobustSST2	RobustBOOLQ
LLama2-7B	$67.15_{\pm 5.36}$	$85.66_{\pm 4.88}$	$46.40_{\pm 10.55}$
+Editing	$68.62_{\pm 4.47}$	$89.90_{\pm 4.54}$	$57.50_{\pm 5.10}$
+SFT	$80.14_{\pm 2.40}$	$91.39_{\pm 1.94}$	$81.80_{\pm 3.87}$

Table 7: Comparison of performance and consistency between the SFT and our method on NLU datasets.

Method	PopQA_sport	PopQA_capital
LLama2-7B	$50.83_{/0.79}$	$73.33_{/0.73}$
+Editing	$53.20_{/0.80}$	$74.36_{/0.77}$
+SFT	$74.03_{/0.95}$	$70.89_{/0.91}$

Table 8: Comparison of the performance and consistency between the SFT and our method on NLG datasets.

In contrast, our editing method prioritizes model components interpretability, adjusting the output of the key components coarsely. Thus substantial optimization potential remains. From the perspective of computational resource consumption, our method exhibits a significant advantage over SFT, as shown in Table 9. For instance, on the RobustSST2 dataset, SFT requires 2.02 GPU hours, while our method only needs 0.11 GPU hours.

Method	+SFT	+Editing
RobustSST2	2.02	0.11
RobustMRPC	2.80	0.12
RobustBOOLQ	1.68	0.14
PopQA_sport	1.87	0.10
PopQA_captial	1.93	0.10

Table 9: The comparison of the computational cost between SFT and our method in terms of GPU hour.

7 Conclusion

This paper presents the first analysis of the internal mechanism aspects of an LLM that contribute to the problem of semantic inconsistency. We can precisely diagnose the key components that contribute to a model's semantic consistency. Based on this finding, we propose a model editing method that directly injects biases into the model components of an LLM without mass-manipulating model parameters. The proposed method can significantly improve both semantic consistency and the performance of LLMs on the constructed NLU and open-source NLG datasets. Also, our methods exhibit promising generalization capabilities on four OOD task datasets.

Limitations

Our study reveals that semantic consistency is correlated with both attention heads and MLPs in an LLM. However, attention heads tend to have a more predominant influence on an LLM than MLPs with the majority of editing operations focusing on them. Future research will focus on exploring the role of MLPs in the semantic consistency of LLMs.

Despite achieving comparable results on OOD settings, our editing method is not sufficiently validated in terms of other metrics, like locality and portability (Yao et al., 2023). Therefore, more rigorous and effective testing methods are required to evaluate the performance of the proposed method.

We aim to develop an interpretability-oriented approach to enhance the semantic consistency of LLMs. Despite our model editing method being comparably transparent and computationally efficient, it still lags behind an SFT approach in terms of performance. In the future, we plan to extend our research to identify the circuits (Elhage et al., 2021) related to semantic consistency and understand their causal mechanisms. In this way, we can further advance the development of effective techniques that improve the semantic consistency of LLMs while prioritizing interpretability and efficiency.

References

Guillaume Alain and Yoshua Bengio. 2016. Understanding intermediate layers using linear classifier probes. *arXiv preprint arXiv:1610.01644*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. 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 2924–2936.

Bill Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1.

Constanza Fierro and Anders Søgaard. 2022. Factual consistency of multilingual pretrained language models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3046–3052, Dublin, Ireland. Association for Computational Linguistics.

Chengguang Gan and Tatsunori Mori. 2023. Sensitivity and robustness of large language models to prompt in japanese. *arXiv preprint arXiv:2305.08714*.

Amr Hendy, Mohamed Abdelrehim, Amr Sharaf, Vikas Raunak, Mohamed Gabr, Hitokazu Matsushita, Young Jin Kim, Mohamed Afify, and Hany Hassan Awadalla. 2023. How good are gpt models at machine translation? a comprehensive evaluation. *arXiv preprint arXiv:2302.09210*.

Ole Jorgensen, Dylan Cope, Nandi Schoots, and Murray Shanahan. 2023. Improving activation steering in language models with mean-centring. *arXiv preprint arXiv:2312.03813*.

Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Inference-time intervention: Eliciting truthful answers from a language model. *arXiv preprint arXiv:2306.03341*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Xiao Pu, Mingqi Gao, and Xiaojun Wan. 2023. Summarization is (almost) dead. *arXiv preprint arXiv:2309.09558*.

Ella Rabinovich, Samuel Ackerman, Orna Raz, Eitan Farchi, and Ateret Anaby-Tavor. 2023. Predicting question-answering performance of large language models through semantic consistency.

Harsh Raj, Vipul Gupta, Domenic Rosati, and Subhabrata Majumdar. 2023. Semantic consistency for assuring reliability of large language models. *arXiv preprint arXiv:2308.09138*.

Harsh Raj, Domenic Rosati, and Subhabrata Majumdar. 2022. Measuring reliability of large language models through semantic consistency. In *NeurIPS ML Safety Workshop*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Zengzhi Wang, Qiming Xie, Zixiang Ding, Yi Feng, and Rui Xia. 2023. Is chatgpt a good sentiment analyzer? a preliminary study. *arXiv preprint arXiv:2304.04339*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

Chunting Zhou, Junxian He, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2022. Prompt consistency for zero-shot task generalization. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2613–2626, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix X. Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *CoRR*, abs/2012.00363.