Interpreting token compositionality in LLMs: A robustness analysis

Anonymous ACL submission

Abstract

Understanding the internal mechanisms of large language models (LLMs) is integral to enhancing their reliability, interpretability, and inference processes. We present Constituent-Aware Pooling (CAP), a methodology designed to analyse how LLMs process compositional linguistic structures. Grounded in principles of compositionality, mechanistic interpretability, and information gain theory, CAP systematically intervenes in model activations through constituent-based pooling at various model levels. Our experiments on inverse definition modelling, hypernym and synonym prediction reveal critical insights into transformers' limitations in handling compositional abstractions. No specific layer integrates tokens into unified semantic representations based on their constituent parts. We observe fragmented information processing, which intensifies with model size, suggesting that larger models struggle more with these interventions and exhibit greater information dispersion. This fragmentation likely stems from transformers' training objectives and architectural design, preventing systematic and cohesive representations. Our findings highlight fundamental limitations in current transformer architectures regarding compositional semantics processing and model interpretability, underscoring the critical need for novel approaches in LLM design to address these challenges.

1 Introduction

The number and diversity of applications for Transformer-based large language models (LLMs) are currently at an accentuated growth, considering their performance at major NLP tasks continues to exceed expectations. Yet, characteristics linked to LLMs critical deficiencies, such as hallucinations and lack of explainability, remain poorly elucidated despite intense research efforts. In particular, there is the matter of linguistic compositionality: how different units of text (morphemes, words, phrases) are combined into units of meaning, and how this can related to models of language interpretation. Uncovering compositional representations in Transformer-based models is a key step in reducing the semantic gap between user intent and model interpretation, explaining model behaviour in different settings and mitigating hallucinations and inconsistent reasoning. To this effect, current work has predominantly focused on mapping input-outputs pairs (Yin et al., 2024), input embeddings (Haslett and Cai, 2024) and layer-wise outputs (Yu and Ettinger, 2020; Modarressi et al., 2023) to expected latent representations by means of semantic similarity or correlation. Such focus reflects two intuitive expectations regarding LLM behaviours:

- LLMs should have an internal representation that surfaces a token/word-level compositionality.
- While substantially distributed across the model's parameters, (phrasal, sentence-level) meaning is expected to be "localisable" in order to be addressed during inference.

Prior work has shown, through different perturbation experiments, that LLM representations and responses are fragile and unreliable across models (Wang et al., 2023; Fodor et al., 2024; Hu et al., 2024). Furthermore, analysis of phrasal representations (Yu and Ettinger, 2020; Carvalho et al., 2022) revealed no significant correlation with phrasal compositional semantics. However, the mechanisms behind such obfuscation of the knowledge addressable by a Transformer model are still open research, which motivates this work.

Aiming to elucidate how token representations, in particular w.r.t. compositionality, affect LM interpretation, this work investigates the sensitivity/robustness of LLMs across local compositional perturbations at different stages of a Transformer

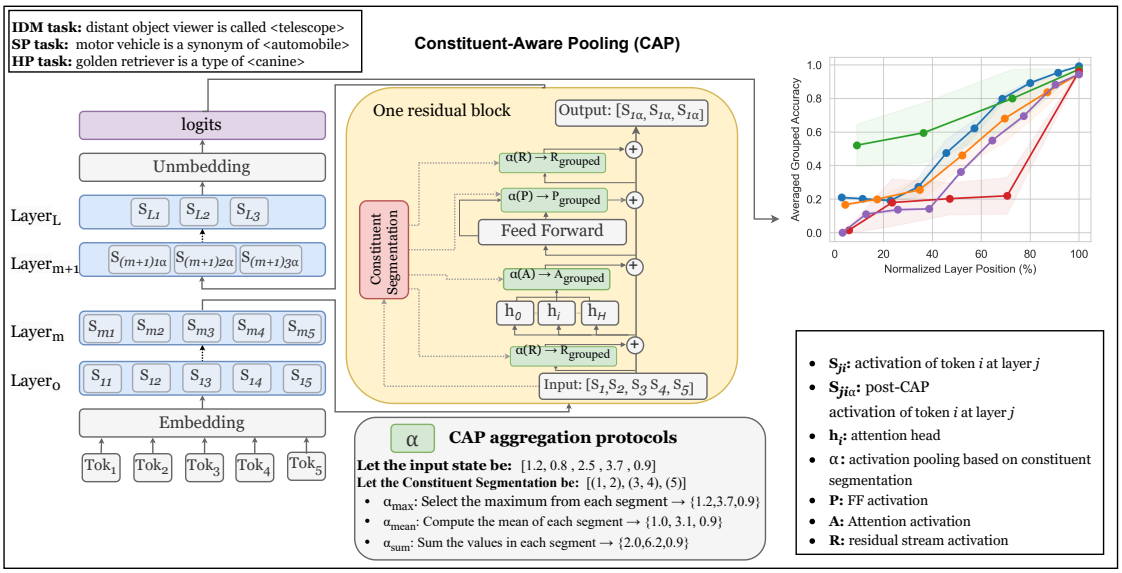

Figure 1: Illustration of the CAP process. Constituent Segmentation creates segments based on token constituents, representing linguistic units like words or phrases. CAP is applied at layer m, pooling activations using aggregation protocols (e.g., max, mean, sum). CAP can be applied to any component, provided dimensionality is maintained. Activations at layer $m+1$ show the model's new-reduced dimensionality. The results graph reflects how CAP is applied at different depths, with accuracy recorded for comparison..

model. In this context, we propose *Constituent-Aware Pooling (CAP)*, a perturbation method for pooling LLM activations that correspond to individual tokens into activations representing more coherent linguistic units, using it to improve the understanding of compositionality mechanisms within transformer-based LMs (for an overview of compositionality and localisation, see Appendix A).

Through this intervention, contrary to expectations of incremental compositional/semantic buildup, we found that LLMs focus heavily on isolated token features rather than hierarchically integrating semantic information across layers, leading to significant performance drops when compositional activation perturbations (CAP) are applied. These fluctuations, especially pronounced in early and middle layers, indicate that aggregation of syntactic and semantic information is distributed across multiple layers rather than localised in any single layer. Surprisingly, larger models are more fragile to compositional perturbations than smaller ones, highlighting the significant difference in the way Transformer-based LMs build meaning representations.

Additionally, we interpret these empirical findings under the light of an information theoretical framework by suggesting that transformer models maximise information gain about the next predicted token by delaying the aggregation of input token representations to later layers, thereby reducing mutual information (redundancy) between tokens at the same layer to optimise prediction, leading to longer aggregation paths across multiple layers.

In summary, the contributions of this work are:

1. A systematic analysis of compositional aggregation (word and phrasal-level) robustness for current LLMs.
2. A theoretical account for the observed results, which also explains the difficulty in locating compositional semantic representations in Transformer models.

Based on the empirical results and theoretical model presented here, we postulate that traditional compositional semantic representations do not exist in any particular stage of a standard Transformer model. This is independent of model size, supervision type or inference task, but linked to the number of hidden layers. Our findings point toward the use of specialised architectures and/or training objectives in order to elicit such representations. The supporting datasets and software are available at *anonymous url*.

2 Tokenisation and compositionality in LLMs

Intuitively, aggregating the representations of tokens that compose a single meaning unit (e.g., averaging the embeddings of 'm', 'amm' and 'al' to form a single token embedding) and then to larger phrasal units (e.g. adjectival and noun compositions), would have a relatively small impact on model inference, since they have a strong dependence on each other in a given context and thus share significant information. However, it has been shown that LLMs are highly sensitive to token placement(Yin et al., 2024; Hu et al., 2024) and that their internal representations have no significant correlation with phrasal composition semantics (Yu and Ettinger, 2020; Carvalho et al., 2022).

The observed disconnection between LLM internal representations and linguistic knowledge regarding compositionality raises practical and theoretical questions towards the robustness of such models to perturbations strictly tied to compositional semantics (Appendix A). Such questions are specially relevant in solving semantic gaps between input prompts and expected responses, as well as localising linguistic knowledge and improving explainability. One way in which they can be addressed is by systematically assessing the impact of said perturbations on model inference performance, at each model layer. We elaborate the methodology to achieve this goal in the following section.

3 Assessing compositional aggregation robustness

In order to accurately measure the effects of compositional aggregation at different layers of abstraction within the model, the inference objective should be a task that is both: 1) strictly dependent on the input tokens and their composition, with few possible input variations; 2) contain as few tokens as possible in the output. For this reason, the following tasks were selected (Figure 1):

1. Inverse definition modelling (IDM): predicting a term given its definition.
2. Synonym prediction (SP): producing a synonym for a given word.
3. Hypernym prediction (HP): generating a more general term for a given word.

The reader is referred to Appendix B.1 for a full formal definition of the tasks and their associated input formats.

Regarding a method to deliver compositional perturbations, we propose **Constituent-Aware Pooling (CAP)**, an approach for pooling LLM activations that correspond to individual tokens into activations representing cohesive linguistic units. CAP operates at two levels: (i) word-level: Grouping tokens that form a single word, and (ii) phrase-level: Grouping tokens that form a single phrase.

For the word-level, CAP reverse-maps each model's tokeniser to reconstruct complete words and determine their ranges. At the phrase-level, it employs a syntactic parser (e.g., Benepar parser (Kitaev et al., 2019; Kitaev and Klein, 2018)) to match tokens to corresponding phrasal constituents and establish their ranges. Aggregations are done progressively from the first to the last layers, using three different pooling modes: *Max* aggregation, which takes the maximum activation within the segment; *Mean* aggregation, which averages the activations across the segment; and *Sum* aggregation, which sums the activations across the segment, capturing cumulative information loss. These aggregation types (max, mean, and sum) offer different perspectives on how information is processed and retained, with Max aggregation highlighting critical features, while Mean and Sum provide broader views of overall performance sensitivity and cumulative loss across tokens.

Formally, we focus on a mathematical conceptualisation of transformers[1] introduced by (Elhage et al., 2021). In Transformer architectures, computation is organised into sequential layers or residual blocks that interact via a shared data pathway (residual stream). Each layer reads input from the residual stream, processes it through its components, and writes the output back to the residual stream. The primary components within these layers are attention heads and feed-forward (FF) neural networks. Attention heads are responsible for transferring information between tokens through the self-attention mechanism, allowing each token to attend to others in the sequence. Feed-forward layers apply non-linear transformations independently to each token representation, enhancing the model's expressive capacity. The residual stream serves to store and propagate information across layers, enabling the integration of new outputs with existing representations while

[1] In this paper, the term 'transformer models' specifically refers to autoregressive transformers.

preserving original input information through residual connections. The structure of transformers provides a basis for processing activations within these model components, enabling the systematic evaluation of compositionality at various layers.

Given a model with L layers and an input sequence of length K, let (s_i, e_i) represent the set of multiple syntactic units ranges, where each (s_i, e_i) corresponds to a word, a phrase or other linguistic constituents. We apply CAP over these unit ranges to key model components, including attention heads, residual streams, and FF layers, using a pooling function α. CAP results in reducing the dimension corresponding to input tokens from K to $K - \Sigma_i(e_i - s_i)$. Below, we formalise how CAP is applied for different components. For simplicity, we demonstrate the operation over one unit, but this can be extended to any number of units. It can also be applied to any component, provided that the dimensional requirements for information flow, as described in (Elhage et al., 2021), are respected.

Attention Heads. Let $A \in \mathbb{R}^{B \times H \times K \times K}$ be the attention activation, where B is the batch size, H is the number of attention heads, and K is the sequence length, we apply CAP over the token range (s_i, e_i) for a syntactic constituent as follows:

$$\begin{aligned} A_{\text{grouped}}[b, h, i, j] &= \alpha_{t=s_i}^{e_i} \alpha_{t'=s_j}^{e_j} A[b, h, t, t'] \\ A &= A_{\text{grouped}} \end{aligned} \tag{1}$$

where α represents the pooling function applied over the specified token ranges (s_i, e_i) and (s_j, e_j), smoothing attention scores across tokens within the same syntactic constituent.

Feed-Forward (FF) Layers. Let $P \in \mathbb{R}^{B \times K \times H}$ denote the FF activations, where B is the batch size, K is the sequence length, and H is the FF dimension. CAP is applied the over the constituent range (s_i, e_i) as follows:

$$\begin{aligned} P_{\text{grouped}}[b, i] &= \alpha_{t=s_i}^{e_i} P[b, t] \\ P &= P_{\text{grouped}}[b, i] \end{aligned} \tag{2}$$

Residual Stream. Let $R \in \mathbb{R}^{B \times K \times H}$ represent the residual stream activations, where B is the batch size, K is the sequence length, and H is the hidden dimension, we apply CAP over the range (s_i, e_i) as follows:

$$\begin{aligned} R_{\text{grouped}}[b, i] &= \alpha_{t=s_i}^{e_i} R[b, t] \\ R &= R_{\text{grouped}} \end{aligned} \tag{3}$$

Here, α serves as the aggregation function, pooling the activations within the token range (s_i, e_i) into a single representation.

These direct interventions within the model are then evaluated with regard to a downstream accuracy effect by measuring an accuracy difference, $\Delta A = A_o - A_c$, where A_c is post-CAP accuracy. As interventions can be localised within the A, P, R components, layer-wise, CAP allows for a granular characterisation of the compositional behaviour across the model structure. In broader terms, it is assumed that a more localised compositional behaviour across layers would imply in a lower Δ.

4 Empirical analysis

4.1 Experimental setup & datasets

Metrics. To evaluate model performance, we report original accuracy (A_o) and accuracy drop after grouping ($\Delta A = A_o - A_c$, where A_c is post-CAP accuracy). These metrics effectively capture the models' performance and the impact of CAP. ΔA directly quantifies the grouping effect, enabling clear cross-model and cross-task comparisons. An expanded reporting on the results is in Appendix C.2.

Datasets Experiments were conducted on three WordNet-derived datasets: definitions, hypernyms, and synonyms (Fellbaum, 1998). These correspond to the three tasks: IDM, HP, and SP. Initially, models were tested on these constructed datasets using the correctly predicted samples as the baseline for further analysis (see Appendix B.2 for datasets details).

LLMs and evaluated dimensions. The methodology was tested across various decoder-only transformer models (Vaswani, 2017). Our main focus was on GPT-2 (small: 124M, medium: 355M, large: 774M parameters) (Radford et al., 2019), Gemma (2B parameters) (Team et al., 2024), and Llama-3 (8B parameters) (Dubey et al., 2024). The models differ in tokenisation methods: GPT-2 models use byte-level BPE tokenisation, Llama-3 employs byte-level BPE but with a large 128,000-token vocabulary, while Gemma-2B utilises a SentencePiece-based (Kudo and Richardson, 2018) tokeniser. All models were tested with and without fine-tuning on downstream tasks. Fine-tuning was conducted for three epochs with a learning rate of 5e-5. The models were chosen to represent a range

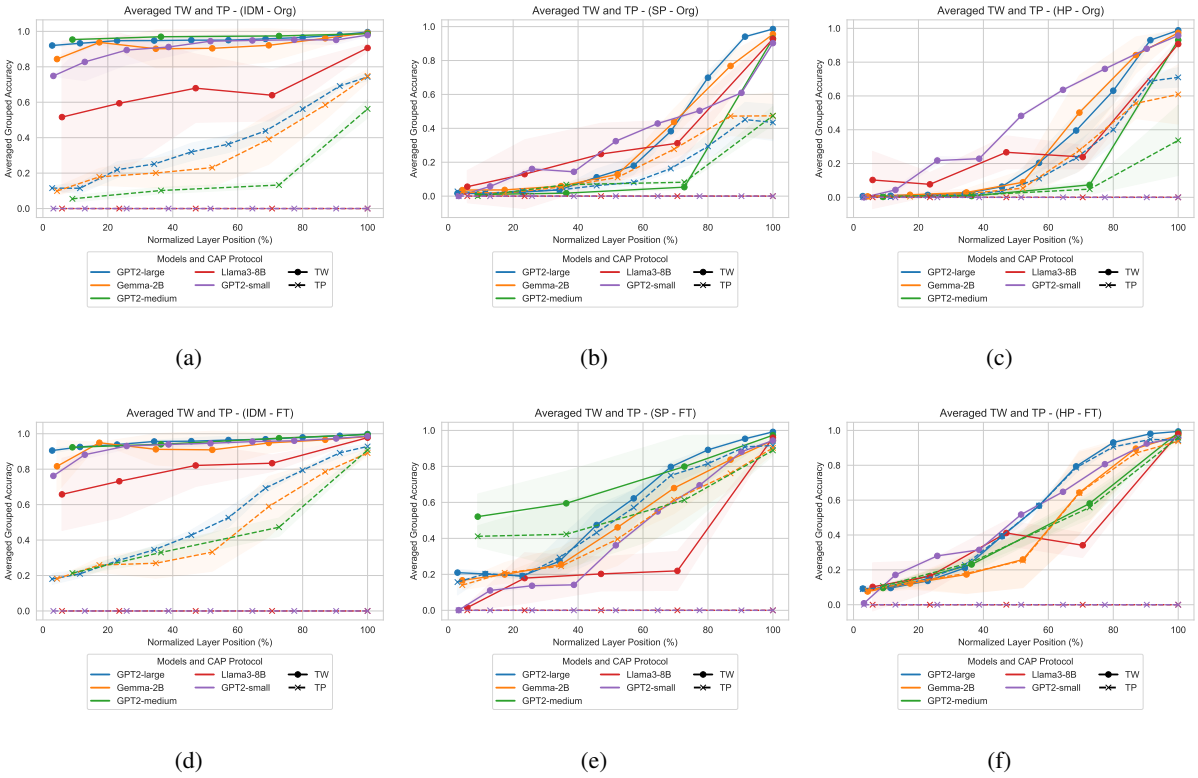

Figure 2: Average grouped accuracy of CAP across different aggregation functions for normalised layer positions (0%-100%) is shown for word-level CAP (TW) and phrasal-level CAP (TP). The shaded areas represent the variance in model performance, with larger shaded regions indicating greater variability across layers. Sub-figures (a)-(c) illustrate the CAP effect on the original (Org) models, while sub-figures (d)-(f) show its impact on the fine-tuned (FT) models. Fine-tuning consistently improves performance, particularly in the middle to late layers (25%-100%), while early layers (0%-25%) show more variability and lower accuracy across models.

Model	Layer Position	Original			Fine-tuned		
		Max	Mean	Sum	Max	Mean	Sum
GPT2-large	1%	8.06%	9.15%	6.70%	10.61%	10.01%	7.83%
	25%	5.19%	4.94%	5.63%	6.25%	5.77%	6.32%
	75%	5.28%	2.62%	2.39%	3.66%	1.62%	0.88%
	100%	0.84%	0.12%	0.19%	0.22%	0.16%	0.16%
Gemma-2B	1%	97.91%	23.51%	23.75%	57.58%	22.70%	21.99%
	25%	86.32%	16.20%	19.27%	50.45%	14.08%	15.57%
	75%	52.38%	31.03%	24.74%	21.77%	14.99%	12.80%
	100%	6.87%	10.61%	10.61%	2.21%	2.05%	2.05%
Llama3-8B	1%	25.49%	24.99%	24.94%	24.44%	23.42%	23.48%
	25%	20.02%	5.87%	5.74%	8.81%	6.03%	5.92%
	75%	7.31%	3.40%	3.54%	5.16%	3.47%	3.29%
	100%	2.80%	1.77%	1.77%	1.55%	1.33%	1.33%

Table 1: IDM accuracy drop Δ in the word-level CAP, highlighting best and worst values in both original and fine-tuned models. The layer numbers were normalised to layer positions as percentages of the total layers, which allows comparing equivalent relative depths across models, such as 25% or 75% of the total layers, rather than using absolute layer numbers. This method ensures fair comparisons between models, even with different architectures.

of sizes, architectures, and tokenisation strategies (see Appendix B.3 for further details on the models and fine-tuning parameters).

Experimental setup. All experiments were conducted using 2x NVIDIA RTX A6000 and 2x NVIDIA RTX A100 GPUs, with the experimental framework being developed in Python 3.11.5. We used the Transformers (v4.44.2) and PyTorch (v2.4.1) libraries, along with Transformer-lens (v2.6.0), to train and evaluate models and for probing. Benepar (v0.2.0) was used for sentence parsing, and statistical analysis was supported by Scikit-

learn (v1.5.2).

4.2 Results and discussion

Compositional inference in LLMs is not a purely incremental process. Contrary to expectations of a smooth and steady layer-wise performance improvement, we observe fluctuations when CAP is applied across layers (see Figure 2). Under perturbations, a more significant drop in performance occurs in the early and middle layers, followed by a sharp improvement (Figure 2 (a), (c), (e), and (f)), suggesting that these layers struggle to process the CAPed activations, particularly the linguistic features captured in earlier layers. *Instead of progressively and hierarchically building up semantic information from individual tokens to more complex phrases and words, the models appear to focus heavily on isolated token features.* The results indicate that attention is distributed over the input tokens and model layers in a non-systematic and decentralised manner that is highly context-dependent and shows minimal reliance on the sequential or positional relationships of constituents. This phenomenon is particularly evident in the sharp decline in performance in the SP and HP tasks, where contextual information is limited and while applying CAP on a phrasal level, effectively abstracting the contextual information. We argue that this behaviour results from the model's training objective, which seeks to maximise information gain from each layer towards the predicted tokens, at the cost of reducing mutual information between tokens in a single layer. This means that *aggregation, including syntactic, is performed across multiple layers and thus is not localisable from any single given layer*. An information theoretical analysis elaborates this reasoning in Section 5. Our findings highlight the sensitivity of token representations to compositional structures, suggesting that the performance fluctuations *can be attributed to information loss incurred as a function of token mutual information across layers.*

Larger models are more fragile to compositional perturbations. Larger models are more sensitive to early-layer interventions than smaller models. For instance, while applying word-level CAP, GPT2-large maintains an accuracy close to 90% in both the original and fine-tuned cases for the IDM task, while Llama3-8B shows an early-layer accuracy of around 75% in both cases (Figure 2 and Table 1). Gemma-2B experiences greater variability, with a sharp accuracy drop of 85%-90% in the early layers across most tasks and settings, highlighting its sensitivity to perturbations. In contrast, smaller models like GPT2-small and GPT2-medium perform more steadily, with early layers' accuracy dropping by around 10-20%. While larger models typically outperform smaller ones in a range of tasks (see Appendix C.1 for original models' performance without intervention), *they exhibit a greater reliance on the identification of intrinsic features in the early layers.* We also observe that the performance of larger models, Gemma-2B and Llama3-8B, is completely broken when applying phrasal-level CAP. This finding suggests a stronger dependency on layer-wise information gain in larger models, where they attempt to separate features in an uncorrelated and highly distinct manner within their parameters. While this ability gives them an advantage in identifying complex and unique feature patterns, it also renders them more vulnerable to changes or noise in the context—*a weakness that threatens the models' robustness and integrity.*

Activation abstraction vs the information loss. For the word-level CAP, Table 1 shows the accuracy drop over a sample of models for the IDM task and the different aggregation functions, presented in a relative manner as to layers positions (see Appendix C.2 for the rest of models and tasks results). The results indicate significant disparities among the aggregation functions' performance, most notably with the Max aggregation. Larger models, such as Gemma-2B and Llama3-8B, were the most affected by this aggregation type. *This finding supports our argument that these models tend to distribute information in a fragmented manner, lacking the integration of compositional (lexical and semantic) information across tokens and contiguous layers.* Thus, they tend to exhibit high sensitivity to any perturbation in their activations. The Mean aggregation provides a more balanced method, consistently resulting in more moderate accuracy drops. However, even with this balanced aggregation, the drop in performance suggests that *there is no identifiable consistent and robust mechanism for composing token meanings.* This issue is more prominent in the token-phrases experiments, as shown in Figure 2. Interestingly, the *Sum aggregation performed better in fine-tuned models*, while the **Mean aggregation was slightly more effective in the original models**. Nevertheless, the differences between them were relatively small. The Sum aggregation reflects the cumulative effect of

aggregating tokens into larger segments, reinforcing our earlier conclusion. Instead of progressively building semantic information across layers, *the models exhibit a cumulative loss of information,* particularly when interventions occur in the early layers.

Fine-tuning expedites the recovery process. As seen in Figure 2 (d-f) and Table 1, there is a positive correlation between the models' ability to maintain performance and fine-tuning (FT). The biggest improvements are observed in the 75%-100% layer positions. SP demonstrated the best gain from FT, which we attribute to the task's high specificity and the minimal activation reduction under CAP. Max aggregation shows the greatest improvement with FT, likely due to fine-tuning amplifying dominant activations and helping retain key information. For example, in Gemma-2B, the drop decreased from 97.91% to 57.65% in the 1% layer, and in Llama3-8B, from 14.98% to around 7% in the 75% layer. Mean aggregation also benefited from FT, with the accuracy drop in Gemma-2 B's 75% layer reduced from 31.03% to 15.00%. However, except for Gemma-2B, Sum aggregation saw only slight improvements. For smaller models like GPT2-large, results showed either the opposite behaviour or minimal improvement, which we believe is due to overfitting. While fine-tuning speeds up recovery, it does not fully mitigate performance reduction, indicating that current Transformer architectures still struggle to form compositional semantic representations from individual tokens.

5 Information Gain & Token Mutual Information

The empirical findings can be explained by looking at the autoregressive next-token objective of a transformer model from an information theoretical standpoint: examining the relationship between each generated token Y to the input token representations $R_l(X)$ of each layer l, in terms of Information Gain $IG_{Y,R_l(X)}$, and the aggregation of a pair of input token representations $R_l(X_i), R_l(X_j)$ in terms of their Mutual Information $I(R_l(X_i), R_l(X_j))$.

$IG_{Y,R_l(X)}$ quantifies the amount of information gained about the predicted token Y from the observation of the $R_l(X)$, for which the expectation is the mutual information $I(Y, R_l(X))$ of Y and $R_l(X)$, which is equivalent to the reduction in entropy of Y achieved by learning the state of $R_l(X)$: $IG_{Y,R_l(X)}(Y, r) = H(Y) - H(Y|r)$.

During training, $R_l(X)$ will be adjusted in a way that reduces the uncertainty about Y, meaning it will promote the maximisation of $IG_{Y,R_l(X)}$ for any given layer l, which can be expressed as:

$$IG_{Y,X} = max(\sum_l IG_{Y,R_l(X)}) \tag{4}$$

where $IG_{Y,X}$ represents the information gain of Y w.r.t. input token X.

When looking at two input tokens X_i, X_j, the higher the mutual information $I(R_l(X_i), R_l(X_j))$ is, the lower the impact that aggregating $R_l(X_i)$ and $R_l(X_j)$ would have over $IG_{Y,X}$, as those variables share more of the same information. Intuitively, that would apply to linguistic composition, e.g., tokens that form a word and thus have a stronger dependence when observed together.

However, as the model's ability to predict Y is contingent on the accumulated information of all layers, and Equation 4 is independent of layer order, there is an intrinsic incentive to delay the aggregation of information (to later layers), as

$$IG_{R_{l_p}(X),R_{l_q}(X)} < IG_{R_{l_p}(X),R_{l_r}(X)}, \quad \forall p < q < r \tag{5}$$

where p, q and r are layer indices, i.e., later layers have more information about the inputs than earlier ones. This can be explained in that optimising Equation 4 can be achieved by retaining at each $R_{l_p}(X)$ only the necessary information to maximise $\sum_{i,j} IG_{R_{l_q}(X_i),MHA(R_{l_p}(X_j))}$, where $MHA(R_{l_p}(X_j))$ is the multi-head attention weighted representation. Such an objective implies minimising the mutual information $I(R_{l_p}(X_i), R_{l_p}(X_j))$, i.e., reducing redundancy across tokens from the same layer. Therefore, token dependencies will tend to be modelled by aggregation paths spanning multiple layers, with more layers allowing for more complex and longer paths. This is in line with the findings of Mechanistic Interpretability studies (Elhage et al., 2021; Conmy et al., 2023). Equation 5 also implies that the earlier an aggregation is done, the larger the impact it will have on $IG_{Y,X}$, which explains the empirical results.

The effects of $I(R_l(X_i), R_l(X_j))$ on LLMs are further compounded by the tokenisation objective (e.g., BPE, WordPiece), which *minimises* $I(X_i, X_j)$, i.e., token redundancy, as a means of reducing the vocabulary size, leading to longer aggregation paths.

6 Related work

General compositionality Compositionality is a core concept across multiple disciplines that refers to understanding complex systems through the meanings of their simpler components. It appears in philosophy, cognitive science, linguistics, and machine learning (Tull et al., 2024). In machine learning (ML) and artificial intelligence (AI), compositionality enables their models to generalise to, reason about, and interpret complex meanings from basic, known components. In cognitive science, the Language of Thought Hypothesis (LOTH) (Fodor, 1975) hypothesises that mental processes are compositional, constructing complex thoughts from basic elements (Rescorla, 2024). Recent ML studies explore how transformers and neural networks build their inferences, dissecting potential compositionality. (Dai et al., 2022) show that feedforward layers in transformers act as key-value memories, where the models combine information to create complex hashing-like predictions. MEMIT (Meng et al., 2023) and PMET (Li et al., 2024) reveal how controlled inferences can be built by manipulating models' parts. These studies suggest transformers may employ compositional processing, systematically building complex representations from simpler components.

Linguistic compositionality. In linguistics, compositionality is often linked to the principle of compositionality, which states that the meaning of a complex expression is derived from "the meanings of its constituent parts, the grammar used to combine them, and the syntactic structure as a whole"(Montague and Thomason, 1975). (Yu and Ettinger, 2020) found that transformers mainly encode individual word content rather than true phrase-level meaning, with model performance often tied to word overlap rather than compositional understanding. (Carvalho et al., 2022) probed transformer models' representation of adjectival modifier phenomena in adjective-noun phrases. Their tests revealed that models partially capture meaning intersective composition, which does not generalise across other adjective types. DecompX, a method introduced in (Modarressi et al., 2023), aims to explain transformer models by decomposing token representations and tracing their aggregation through all layers. It provides a more faithful explanation of model predictions by linking individual token dynamics to final decisions without the need for vector aggregation across layers, which offers compositional behaviour analysis on a token-prediction basis in transformer models. (Haslett and Cai, 2024), examine how LLMs segment words into multiple tokens and the extent of semantic information encoded within these tokens. Their findings show that models from Multilingual BERT and GPT-2 to GPT-4o often fail to segment words meaningfully, particularly in non-Latin script languages, where morphemes are not well-represented.

This work complements and expands the aforementioned studies' findings to characterise the previously observed phenomena and better understand the disconnection between model representations and compositional semantics reported in all of them.

7 Conclusion

This work presents a systematic analysis of the robustness of Transformer-based LLMs to perturbations grounded in compositionality. Motivated by prior studies highlighting a significant and unexpected gap between linguistic compositionality and LLM representations, we aimed to characterise the impact of compositional aggregation at each inference step and provide an information-theoretical explanation for the observed results. Our findings indicate a clear pattern of information structuring across contemporary LLMs, where token dependencies tend to be modelled by aggregation paths spanning multiple layers, and the capability to learn complex token structures is gained at the cost of higher sensitivity to perturbations at the inputs and earlier layers. Furthermore, based on the relation between the information gain from input representation to predicted token and the mutual information between token representations in the model, we postulate that compositional semantic representations do not exist in any particular stage of a standard Transformer model for a general input. These findings point to future work on compositional-aware models towards the use of specialised architectures and/or training objectives to elicit such representations.

Limitations

A number of limitations need to be acknowledged regarding our paper. First, the dataset used, WordNet, may not fully represent the diversity of language usage across all relevant domains. Secondly, the employed transformer models are not exempted from possible biases or limitations in their train-

ing data or modelling. Third, our findings rely on the Benepar parsing model, which is liable to inaccuracies as far as linguistic analysis is concerned. Thus, the accuracy of our conclusions is limited to the parsing model performance. Furthermore, the applicability of our results to other languages has not been tested. In future research, these shortcomings should be addressed, and it might also be possible to combine languages and parsing models to further validate and extend our findings.

Ethical Statement

The proposed framework aims to have a positive impact on improving the critical understanding of the mechanisms involved in language interpretation in transformers. A more complete understanding of these mechanisms requires coordination with other interpretability methods.

References

Danilo S Carvalho, Edoardo Manino, Julia Rozanova, Lucas Cordeiro, and André Freitas. 2022. Montague semantics and modifier consistency measurement in neural language models. *arXiv preprint arXiv:2212.04310*.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352.

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.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

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(1):12.

Christiane Fellbaum. 1998. Wordnet: An electronic lexical database. *MIT Press google schola*, 2:678–686.

JA Fodor. 1975. The language of thought.

James Fodor, Simon De Deyne, and Shinsuke Suzuki. 2024. Compositionality and Sentence Meaning: Comparing Semantic Parsing and Transformers on a Challenging Sentence Similarity Dataset. *Computational Linguistics*, pages 1–52.

Gottlob Frege. 1892. Über sinn und bedeutung [on sense and reference]. *Zeitschrift für Philosophie Und Philosophische Kritik*, 100:25–50.

David A Haslett and Zhenguang G Cai. 2024. How much semantic information is available in large language model tokens?

Zhibo Hu, Chen Wang, Yanfeng Shu, Hye-Young Paik, and Liming Zhu. 2024. Prompt perturbation in retrieval-augmented generation based large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1119–1130.

Ray Jackendoff. 1997. *The Architecture of the Language Faculty*. MIT Press.

Jerrold J. Katz and Paul M. Postal. 1963. Semantic interpretation of idioms and sentences containing them. *Quarterly Progress Report*, 70:275–282.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2024. Pmet: Precise model editing in a transformer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18564–18572.

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

Ali Modarressi, Mohsen Fayyaz, Ehsan Aghazadeh, Yadollah Yaghoobzadeh, and Mohammad Taher Pilehvar. 2023. Decompx: Explaining transformers decisions by propagating token decomposition. In *Pro-*

ceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2649–2664.

Richard Montague. 1970a. English as a formal language. In *Linguaggi nella società e nella tecnica*, pages 189–223. Edizioni di Comunità.

Richard Montague. 1970b. Universal grammar. *Theoria*, 36(3):373–398.

Richard Montague and Richmond H Thomason. 1975. Formal philosophy. selected papers of richard montague. *Erkenntnis*, 9(2).

Barbara H. Partee. 1984. Compositionality. In Fred Landman and Frank Veltman, editors, *Varieties of Formal Semantics*, pages 281–312. Foris Publications.

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.

Michael Rescorla. 2024. The Language of Thought Hypothesis. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*, Summer 2024 edition. Metaphysics Research Lab, Stanford University.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Sean Tull, Robin Lorenz, Stephen Clark, Ilyas Khan, and Bob Coecke. 2024. Towards compositional interpretability for xai. *arXiv preprint arXiv:2406.17583*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Haoyu Wang, Guozheng Ma, Cong Yu, Ning Gui, Linrui Zhang, Zhiqi Huang, Suwei Ma, Yongzhe Chang, Sen Zhang, Li Shen, et al. 2023. Are large language models really robust to word-level perturbations? In *Socially Responsible Language Modelling Research Workshop*.

Yongjing Yin, Lian Fu, Yafu Li, and Yue Zhang. 2024. On compositional generalization of transformer-based neural machine translation. *Information Fusion*, 111:102491.

Lang Yu and Allyson Ettinger. 2020. Assessing phrasal representation and composition in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4896–4907.

A Compositionality and Localisation

The concept of linguistic compositionality has evolved from its origins in Frege's work (Frege, 1892), which started conceptualising the notion that the meaning of a complex expression is determined by its constituent parts and their syntactic arrangement. This principle was formalized by Montague (Montague, 1970b,a), who applied mathematical rigour to natural language semantics, thereby reinforcing the compositional approach within formal semantics. Linguistic phenomena such as idioms, context-dependence, and metaphor, which seemed to violate compositionality, prompted debates on its universality (Katz and Postal, 1963; Jackendoff, 1997), with theoretical accounts evolving to integrate these phenomena, leading to a more nuanced understanding that balances strict compositional rules with allowances for non-compositional elements (Partee, 1984).

While the syntactic-logical connection entailed by formal models is not assumed to be induced by neural language models, there is a common assumption that those models should entail a syntactic compositionality function, which allows for a systematic model for meaning composition, i.e., that the syntactic structure of a complex expression s is significantly determined by the syntactic properties of its constituent parts and the rules used to combine them. Formally, for any sentence s, its syntactic properties can be defined as a function f of the syntactic properties of its immediate constituents $s_1, s_2, \ldots, s_n$ and the syntactic operations applied:

$$\text{Syntax}(s) = f\left(\text{Syntax}(s_1), \text{Syntax}(s_2), \ldots, \text{Syntax}(s_n), \text{Rules}\right) \tag{6}$$

Within the context of distributed representations, a meaning representation can be factored into its syntactic and content (term embedding) components. A compositional distributional semantic model merges syntactic compositionality with distributional semantics by representing token meanings as vectors (token embeddings) in a continuous semantic space and combining them according to syntactic structure. Formally, each token t is associated with a vector $\mathbf{v}_t \in \mathbb{R}^n$ that captures its semantic content based on distributional information.

For a complex syntactic expression s composed of constituents $s_1, s_2, \ldots, s_n$, the semantic representation $\mathbf{v}_s$ is computed using a compositional function f that integrates both the vectors of the constituents and the syntactic operations applied:

$$\mathbf{v}_s = f\left(\mathbf{v}_{s_1}, \mathbf{v}_{s_2}, \ldots, \mathbf{v}_{s_n}, \text{Syntactic structure}\right) \tag{7}$$

This function f is designed to reflect syntactic compositionality by structurally combining the embeddings of the constituents according to the syntactic rules governing their combination.

In the context of a specific transformer-based LM model implementing an interpretation function of an input s, the question which is central to this work is whether the contiguous composition of tokens is reflected within the structure of the transformer-based LMs and its constituent parts, layers $l_0 ... l_n$, multi-head attention, feed-forward layers and residual connections, i.e. whether the representations $\mathbf{h}_i^{(k)}$ at each layer l_k explicitly encode the composition of contiguous tokens t_i, t_{i+1}, and how the model's components contribute to this encoding.

B Elaborations on Experimental Setup

B.1 Downstream Task Definitions

The tasks selected for this study are designed to evaluate the effects of compositional aggregation, focusing on tasks that are strictly dependent on input tokens and their compositional semantics while minimising variability. Each task produces a single-token output, and predictions are considered correct if they exactly match the target token. The following are the formal definitions for each task.

Inverse Definition Modelling (IDM): The *Inverse Definition Modelling (IDM)* task involves predicting a term T based on a given natural language definition D. Let $D = \{d_1, d_2, \ldots, d_n\}$ represent the sequence of tokens constituting the definition. The goal is to generate the corresponding term T, where:

$$T = \arg\max_{t \in \mathcal{V}} P(t \mid D) \tag{8}$$

Here, $\mathcal{V}$ is the vocabulary of possible terms, and t is a candidate term. A prediction is correct if the term T exactly matches the target term. The task prompt used for IDM was structured as follows:

```
"<definition> is called a"
```

For example, given the definition "A domesticated carnivorous mammal that typically has a long snout, an acute sense of smell, non-retractile claws, and a barking or howling voice," the task would require the model to predict the term "dog."

Synonym Prediction (SP): The *Synonym Prediction (SP)* task requires the model to generate a synonym S for a given word W. Let $W \in \mathcal{V}$ represent the input word. The task is to predict a synonym S, such that:

$$S = \arg\max_{s \in \mathcal{V}} P(s \mid W) \tag{9}$$

where s is a candidate synonym from the vocabulary $\mathcal{V}$. The prediction is considered correct if S exactly matches the target synonym. The task prompt used for SP was structured as follows:

```
"<word> is a synonym of"
```

For instance, given the input word "happy," the task would ask the model to predict the synonym "joyful."

Hypernym Prediction (HP): The *Hypernym Prediction (HP)* task involves predicting a more general term, or hypernym, H for a given word W. Let $W \in \mathcal{V}$ represent the input word. The objective is to predict a hypernym H, such that:

$$H = \arg\max_{h \in \mathcal{V}} P(h \mid W) \tag{10}$$

where h is a candidate hypernym. The prediction is correct if H exactly matches the intended hypernym. The task prompt used for HP was structured as follows:

```
"<word> is a type of"
```

For example, given the word "cat," the task would ask the model to predict the hypernym "animal."

These tasks focus on generating precise, single-token predictions, allowing for a rigorous evaluation of the model's ability to capture and process compositional semantics.

B.2 Dataset Descriptions and Preprocessing

The training and test datasets are constructed by extracting definitions, hypernyms, and synonyms for each synset from WordNet (Fellbaum, 1998), whose usage is unencumbered by licensing restrictions. WordNet is a lexical database of the English language, containing over 117,000 synsets of nouns, verbs, adjectives, and adverbs. Each synset represents a unique concept and is annotated with part of speech, definition, hypernyms, synonyms, and other semantic relationships. Focused

Model	Task	Original Test Set	Fine-tuned Test Set
GPT2 (S,M,L)	IDM	11,948	8,651
	SP	7,753	5,578
	HP	25,364	18,273
Gemma-2B	IDM	24,831	17,859
	SP	16,014	11,533
	HP	44,687	32,209
Llama3-8B	IDM	14,991	10,828
	SP	9,360	6,723
	HP	31,962	23,070

Table 2: Test set sizes for each model and task (IDM: Inverse Dictionary Modelling, SP: Synonym Prediction, HP: Hypernym Prediction) derived from WordNet.

Model	Params	Layers	D_{model}	Heads	Act.	MLP Dim
GPT2-small	85M	12	768	12	GELU	3072
GPT2-medium	302M	24	1024	16	GELU	4096
GPT2-large	708M	36	1280	20	GELU	5120
Gemma-2B	2B	32	4096	16	GELU	8192
LLama3-8B	7.8B	32	4096	32	SiLU	14336

Table 3: Model properties across architectures. Params: number of parameters, Layers: number of layers, D_{model}: size of word embeddings and hidden states, Heads: number of attention heads, Act.: Activation function, MLP Dim: dimensionality of the FF layers.

on general-purpose vocabulary and does not target specific demographic groups or domains. Definitions were cleaned using typical preprocessing techniques, such as removing special characters, punctuation, and extra spaces, and removing parenthesised content when necessary. The dataset was initially split 80-20, with 20% used for training. The remaining 80% was then split 90-10, with 10% for validation and 90% for testing. The test dataset was filtered to retain only single-token predictions matching each model's tokenisation. Table 2 shows the test dataset sizes used for each task and model, including inverse dictionary modelling (IDM), synonym prediction (SP), and hypernym prediction (HP).

Model	Original			Fine-tuned		
	IDM	SP	HP	IDM	SP	HP
GPT2-small	7.10%	2.59%	17.04%	13.52%	8.18%	26.59%
GPT2-medium	10.70%	4.27%	16.77%	16.34%	11.65%	28.75%
GPT2-large	11.33%	5.93%	13.90%	17.80%	11.78%	27.66%
Gemma-2B	16.76%	6.38%	10.16%	9.57%	10.75%	23.31%
Llama3-8B	25.17%	10.80%	15.30%	18.28%	10.75%	24.14%

Table 4: Baseline performance of various models on three tasks: (inverse dictionary modelling) IDM, synonym prediction (SP), and hypernym prediction (HP). The values represent the accuracy of each model's original and fine-tuned versions.

B.3 Model Specifications and Fine-tuning Parameters

The results in Table 3 provide a comparative overview of various Transformer models used in this study. We used GPT-2 models (released under the Modified MIT License), Gemma-2B (released under the Gemma Terms of Use), and Llama3-8B (released under the Meta Llama 3 Community License). All models were used for research purposes, specifically for language modeling and text generation in English, aligning with their intended usage. The models differ in their number of parameters, layers, heads, and feedforward (FF) dimensions. The number of parameters ranges from 85M for GPT2-small to 7.8B for LLama3-8B. The activation functions and FF dimensions also highlight variations in the internal processing architecture, influencing the models' performance across different tasks. In addition to these architectural differences, the models were fine-tuned using a consistent set of hyperparameters. The fine-tuning process spanned over 3 training epochs with a batch size of 16. The learning rate was set to 5e-5, while a weight decay of 0.01 was applied to prevent overfitting. Training logs were generated every 200 steps, with model checkpoints saved every 1000 steps, but limited to retaining only one checkpoint to manage storage efficiently. The evaluation strategy during fine-tuning was set to evaluate at the end of each epoch, and similarly, the model was saved at the end of each epoch as well.

C Detailed Performance Evaluation and Results

C.1 Baseline Performance

Table 4 summarises the baseline performance of the models used in this paper on the three tasks. The results include the accuracy of each model's original and fine-tuned versions on the test set described in Table 2. Fine-tuning generally improves performance, particularly in the larger models such as Gemma-2B and Llama3-8B, which show notable increases in accuracy in most tasks, except the IDM task.

C.2 Comprehensive CAP Results for All Models and Tasks

Table 5 and Table 6 present the reduction in accuracy when applying word-level and phrasal CAP, respectively, to several models—GPT2-small, GPT2-medium, GPT2-large, Gemma-2B, and Llama3-

8B—across three tasks: Inverse Dictionary Modelling (IDM), Synonym Prediction (SP), and Hypernym Prediction (HP). The results of phrasal-level CAP for Gemma-2B and Llama3-8B are not reported due to the severe degradation in model performance under these conditions, rendering the outputs effectively unusable.

Let A_o represent the original accuracy and A_c represent the accuracy after applying CAP. The reported drop in accuracy, ΔA, is calculated as:

$$\Delta A = A_o - A_c \tag{11}$$

This ΔA value is expressed in percentage points. For example, $\Delta A = 40$ indicates that the model's accuracy has decreased by 40 percentage points from its original performance, which could represent a change from $A_o = 100\%$ to $A_c = 60\%$, or any other pair of accuracies with a 40 percentage point difference.

The tables report ΔA for different layer positions (1%, 25%, 75%, and 100%) in both Original and Fine-tuned settings, using three CAP protocols: Max, Mean, and Sum. This representation allows for a direct comparison of CAP's impact across different models and tasks, independent of their baseline performance levels.

Model	Layer Position	Original			Fine-tuned		
		Max	Mean	Sum	Max	Mean	Sum
IDM (Inverse Dictionary Modelling)							
GPT2-small	1%	4.76%	4.44%	4.69%	8.04%	7.72%	7.22%
	25%	3.09%	2.74%	3.26%	5.87%	5.85%	6.24%
	75%	2.64%	2.36%	2.74%	2.72%	2.47%	2.35%
	100%	1.43%	1.24%	1.24%	0.46%	0.39%	0.39%
GPT2-medium	1%	16.75%	16.36%	13.77%	24.51%	12.70%	7.44%
	25%	6.73%	5.692%	6.22%	5.04%	4.84%	5.36%
	75%	18.61%	2.13%	2.89%	11.79%	2.09%	1.72%
	100%	1.58%	0.41%	0.41%	2.27%	1.29%	1.29%
GPT2-large	1%	8.06%	9.15%	6.70%	10.61%	10.01%	7.83%
	25%	5.19%	4.94%	5.63%	6.25%	5.77%	6.32%
	75%	5.28%	2.62%	2.39%	3.66%	1.62%	0.88%
	100%	0.84%	0.12%	0.19%	0.22%	0.16%	0.16%
Gemma-2B	1%	97.91%	23.51%	23.75%	57.58%	22.70%	21.99%
	25%	86.32%	16.20%	19.27%	50.45%	14.08%	15.57%
	75%	52.38%	31.03%	24.74%	21.77%	14.99%	12.80%
	100%	6.87%	10.61%	10.61%	2.21%	2.05%	2.05%
Llama3-8B	1%	25.49%	24.99%	24.94%	24.44%	23.42%	23.48%
	25%	20.02%	5.87%	5.74%	8.81%	6.03%	5.92%
	75%	7.31%	3.40%	3.54%	5.16%	3.47%	3.29%
	100%	2.80%	1.77%	1.77%	1.55%	1.33%	1.33%
SP (Synonym Prediction)							
GPT2-small	1%	99.04%	99.04%	99.04%	59.68%	49.40%	34.68%
	25%	98.56%	98.56%	97.60%	61.09%	30.85%	29.64%
	75%	96.15%	94.23%	93.75%	40.12%	9.68%	10.48%
	100%	6.73%	7.21%	7.21%	3.23%	2.42%	2.42%
GPT2-medium	1%	96.43%	96.43%	96.43%	83.35%	82.50%	84.06%
	25%	96.13%	96.43%	96.43%	79.22%	80.22%	80.79%
	75%	63.93%	48.30%	56.63%	48.36%	23.23%	24.53%
	100%	6.68%	3.41%	3.41%	6.55%	5.12%	5.12%
GPT2-large	1%	98.49%	98.49%	98.06%	78.61%	78.33%	80.17%
	25%	97.63%	97.63%	97.63%	80.93%	81.78%	79.89%
	75%	34.27%	27.59%	28.52%	11.91%	10.02%	10.49%
	100%	1.29%	1.51%	1.51%	1.22%	39.12%	0.61%
Gemma-2B	1%	99.99%	99.80%	83.47%	99.93%	99.15%	96.38%
	25%	99.99%	97.46%	63.68%	90.20%	90.24%	65.82%
	75%	84.63%	60.66%	61.15%	89.87%	75.68%	68.65%
	100%	4.30%	8.69%	8.69%	2.98%	4.57%	4.57%
Llama3-8B	1%	99.99%	99.90%	99.90%	99.99%	99.88%	99.88%
	25%	85.55%	83.50%	82.81%	87.63%	85.75%	85.63%
	75%	53.35%	50.55%	49.77%	31.29%	30.29%	29.91%
	100%	9.28%	9.96%	9.96%	5.20%	5.82%	5.82%
HP (Hypernym Prediction)							
GPT2-small	1%	99.75%	99.75%	99.75%	91.19%	91.08%	88.20%
	25%	99.47%	99.29%	98.94%	81.35%	76.76%	72.63%
	75%	95.40%	91.16%	91.32%	48.75%	38.54%	38.40%
	100%	8.12%	6.39%	6.39%	1.35%	1.38%	1.28%
GPT2-medium	1%	99.42%	99.40%	99.44%	93.42%	92.17%	91.69%
	25%	99.11%	98.55%	97.85%	91.64%	86.11%	85.76%
	75%	74.83%	33.22%	41.52%	3.86%	2.23%	2.33%
	100%	4.42%	1.79%	1.79%	3.86%	2.23%	2.32%
GPT2-large	1%	99.27%	99.32%	99.20%	91.49%	90.90%	89.80%
	25%	98.81%	98.75%	98.10%	87.30%	87.54%	84.16%
	75%	45.17%	29.85%	35.66%	7.61%	6.89%	6.22%
	100%	2.14%	0.45%	0.90%	0.69%	0.50%	0.56%
Gemma-2B	1%	99.99%	98.97%	70.22%	99.88%	95.39%	74.03%
	25%	99.98%	90.58%	86.35%	90.98%	73.78%	86.01%
	75%	68.14%	80.06%	80.20%	58.56%	72.57%	66.56%
	100%	5.89%	10.99%	10.99%	1.58%	2.12%	2.12%
Llama3-8B	1%	99.99%	99.99%	99.14%	99.99%	99.10%	99.14%
	25%	80.85%	76.97%	76.81%	72.67%	71.86%	71.40%
	75%	24.43%	24.39%	23.11%	19.65%	19.71%	18.77%
	100%	3.83%	4.49%	4.49%	4.63%	4.04%	4.20%

Table 5: Performance drop (in percentage points) for GPT2-small, GPT2-medium, GPT2-large, Gemma-2B, and Llama3-8B models after applying word-level CAP across three tasks: Inverse Dictionary Modelling (IDM), Synonym Prediction (SP), and Hypernym Prediction (HP). Results are reported for different layer positions (1%, 25%, 75%, and 100%) in both Original and Fine-tuned settings, using three CAP protocols: Max, Mean, and Sum.

Model	Layer Position	Original			Fine-tuned		
		Max	Mean	Sum	Max	Mean	Sum
IDM (Inverse Dictionary Modelling)							
GPT2-small	1%	93.00%	93.94%	96.56%	77.912%	77.73%	80.28%
	25%	90.20%	87.85%	91.41%	65.73%	62.95%	72.31%
	75%	87.81%	78.66%	84.90%	55.74%	46.81%	55.73%
	100%	48.10%	45.10%	38.04%	11.11%	8.45%	8.11%
GPT2-medium	1%	87.96%	89.87%	92.52%	81.12%	82.37%	81.83%
	25%	77.06%	82.71%	86.54%	69.53%	75.19%	77.55%
	75%	76.35%	48.76%	57.68%	60.60%	29.52%	33.12%
	100%	29.23%	23.12%	23.21%	13.03%	9.75%	9.94%
GPT2-large	1%	87.06%	89.91%	88.44%	81.14%	85.35%	79.46%
	25%	73.54%	78.18%	82.48%	69.39%	73.85%	71.90%
	75%	49.02%	42.06%	40.38%	20.59%	19.78%	21.45%
	100%	28.14%	24.22%	24.78%	6.46%	6.67%	8.44%
SP (Synonym Prediction)							
GPT2-small	1%	99.99%	99.99%	99.99%	64.90%	58.47%	53.22%
	25%	92.97%	93.36%	93.36%	61.27%	37.19%	74.69%
	75%	92.58%	90.63%	92.19%	43.35%	20.57%	52.22%
	100%	58.46%	47.92%	51.43%	13.27%	7.57%	12.45%
GPT2-medium	1%	97.55%	95.11%	99.99%	88.92%	84.23%	84.80%
	25%	97.55%	99.73%	97.55%	75.00%	76.85%	85.65%
	75%	71.20%	68.21%	77.45%	47.72%	22.16%	45.88%
	100%	66.30%	39.40%	52.17%	12.93%	6.68%	9.52%
GPT2-large	1%	96.67%	98.33%	96.67%	92.55%	80.76%	79.58%
	25%	96.67%	96.44%	97.90%	79.44%	80.48%	82.86%
	75%	78.83%	66.72%	66.32%	18.63%	15.80%	21.00%
	100%	67.10%	45.83%	56.68%	9.69%	7.15%	8.33%
HP (Hypernym Prediction)							
GPT2-small	1%	99.40%	99.26%	47.24%	89.31%	89.86%	88.76%
	25%	99.31%	98.12%	46.38%	77.72%	73.12%	76.08%
	75%	95.63%	91.78%	45.57%	47.73%	336.59%	48.32%
	100%	65.62%	45.84%	34.80%	4.80%	3.64%	4.00%
GPT2-medium	1%	99.77%	99.56%	99.950%	92.67%	90.40%	92.54%
	25%	99.92%	99.35%	99.47%	90.38%	84.29%	86.84%
	75%	77.77%	58.17%	80.58%	63.00%	21.55%	23.32%
	100%	59.28%	27.47%	30.54%	8.46%	5.10%	5.10%
GPT2-large	1%	99.77%	99.71%	99.76%	91.63%	92.56%	88.92%
	25%	99.82%	98.72%	98.82%	85.31%	85.35%	84.58%
	75%	66.58%	49.79%	63.56%	9.87%	8.79%	9.73%
	100%	35.57%	24.79%	26.69%	6.99%	5.05%	4.82%

Table 6: Performance drop (in percentage points) for GPT2-small, GPT2-medium, and GPT2-large models after applying phrasal-level CAP across three tasks: Inverse Dictionary Modelling (IDM), Synonym Prediction (SP), and Hypernym Prediction (HP). Results are reported for different layer positions (1%, 25%, 75%, and 100%) in both Original and Fine-tuned settings, using three CAP protocols: Max, Mean, and Sum. Results for Gemma-2B and Llama3-8B are omitted due to severe performance degradation under phrasal-level CAP.