Where Do LLMs Compose Meaning? A Layerwise Analysis of Compositional Robustness

Anonymous ACL submission

Abstract

Understanding how large language models (LLMs) process compositional linguistic structures is integral to enhancing their reliability and interpretability. We present Constituent-Aware Pooling (CAP), a methodology grounded in compositionality, mechanistic interpretability, and information theory that intervenes in model activations by pooling token representations into linguistic constituents at various layers. Experiments across eight models (124M-8B parameters) on inverse definition modelling, hypernym and synonym prediction reveal that semantic composition is not localised to specific layers but distributed across network depth. Performance degrades substantially under constituent-based pooling, particularly in early and middle layers, with larger models showing greater sensitivity. We propose an information-theoretic interpretation: transformers' training objectives incentivise deferred integration to maximise token-level throughput, resulting in fragmented rather than localised composition. These findings highlight fundamental architectural and training constraints requiring specialised approaches to encourage robust compositional processing.

1 Introduction

Large language models (LLMs) based on Transformer architectures have rapidly expanded in scope and capability, demonstrating strong performance across a wide range of NLP tasks. However, critical limitations remain, including hallucinations, limited interpretability and a lack of semantic transparency. One open challenge concerns *linguistic compositionality*: how models combine smaller units of text (e.g., morphemes, words, phrases) into coherent meaning structures, and how this process is reflected in internal representations.

Understanding how and where compositional structure is encoded in LLMs is essential for bridging the gap between user intent and model behaviour. Prior work has explored this by aligning model inputs and outputs (Yin et al., 2024), examining embedding spaces (Haslett, 2024), or analysing layer-wise activations (Yu and Ettinger, 2020; Modarressi et al., 2023).

These approaches are often grounded in two key assumptions: (1) that LLMs internally represent compositional structure at the token or word level, and (2) that this information should be at least partially localisable at specific layers during inference.

However, growing evidence suggests that these assumptions may not hold robustly. LLMs are often brittle under perturbations (Wang et al., 2023; Fodor et al., 2024; Hu et al., 2024), and phrase-level representations may diverge from expected semantics (Carvalho et al., 2025). Nevertheless, the underlying causes of this fragility, particularly at the level of internal activations, remain poorly understood.

To investigate this, we propose *Constituent-Aware Pooling (CAP)*, a structured perturbation method that groups token-level activations into larger linguistic constituents (e.g., words or phrases) at arbitrary layers. CAP enables systematic probing of whether, and where, semantic meaning is robustly composed within the model. By applying CAP at varying depths, we assess the fragility of internal representations to compositional perturbations and examine whether, and how, semantic abstraction is distributed across layers.[1].

Our empirical results challenge the notion of strictly hierarchical semantic buildup. Rather than gradually constructing meaning across layers, LLMs often maintain a token-level focus well into intermediate layers. Even semantically coherent pooling via CAP leads to notable performance drops, particularly when applied in early layers. Surprisingly, larger models are more sensitive to

[1]Supporting code and datasets are available at a public repository < anonymised url >

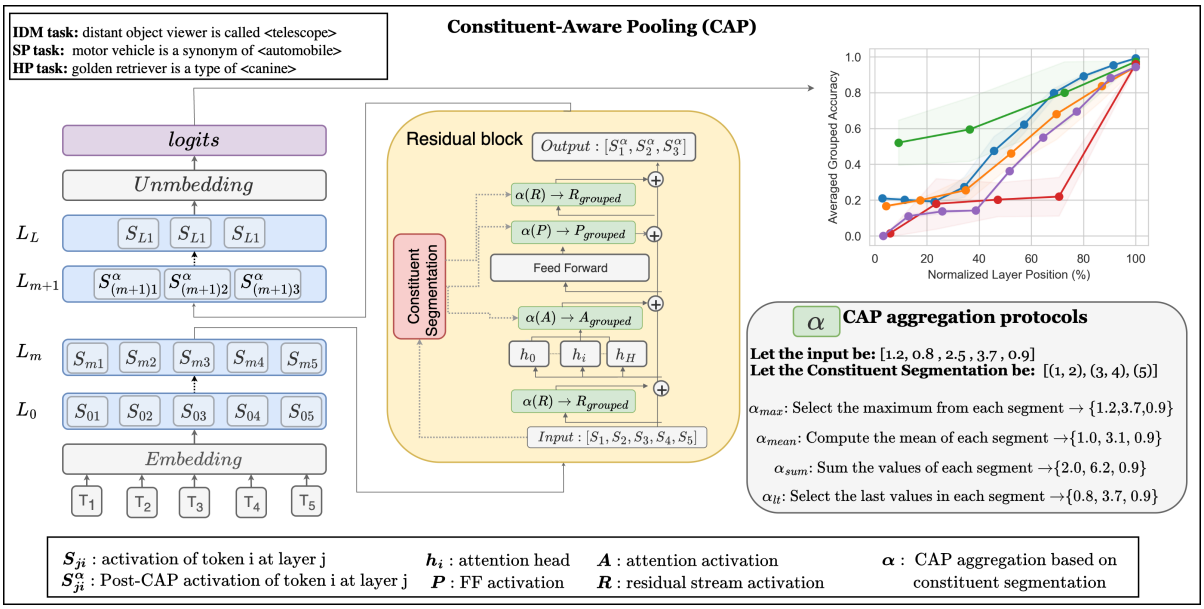

Figure 1: Illustration of the CAP process. Constituent segmentation identifies linguistic units (e.g., words or phrases), and CAP pools their activations at layer m using aggregation (e.g., max, mean, sum, last token). This operation reduces sequence length, and the modified activations are propagated to layer $m+1$. The results graph shows task accuracy under CAP at different depths.

perturbations than smaller ones, suggesting increasing representational fragility with scale.

We interpret these results through an information-theoretic lens, proposing that Transformers may delay integration to maximise token-level information throughput. This yields distributed, rather than localised, composition across layers, resulting in longer dependency paths and reduced mutual redundancy at each layer.

Our findings suggest that compositional semantics are not reliably localised to any fixed layer of standard Transformer models. This pattern holds across model scales, tasks, and supervision types, appearing to be a function of architectural depth. Recovering explicit compositional structure may require specialised training objectives or architectural modifications that encourage more robust hierarchical abstraction.

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., 2025).

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 especially relevant in solving semantic gaps between input prompts and expected responses, as well as localising linguistic knowledge and improving interpretability. One way to address this is by systematically assessing the impact of these perturbations on model inference performance at each model layer. We elaborate on the methodology to achieve this goal in the following section.

3 Assessing compositional aggregation robustness

To accurately assess the effects of compositional grouping at different layers of abstraction within transformer models, 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) contains as few tokens as

possible in the output. For this reason, the following tasks were selected (Fig. 1):

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

Formal definitions and input formats are in Appendix B.1.

Constituent-Aware Pooling (CAP). To introduce structured compositional perturbations, we propose CAP, a method for pooling (i.e., grouping) LLM activations corresponding to tokens that belong to the same linguistic unit. CAP operates at two levels: (i) *Word-level (TW-CAP)*, where subword token activations are aggregated into full word representations via inverse mapping of the tokeniser; and (ii) *Phrase-level (TP-CAP)*, where token activations are aggregated into phrase-level representations by aligning tokens to syntactic constituents identified by a parser such as Benepar (Kitaev et al., 2019; Kitaev and Klein, 2018). In both cases, CAP defines contiguous activation spans and applies a pooling function over each, producing grouped representations that reflect linguistic structure. Parser evaluation and mapping procedures are described in Appendix D.

Formalising CAP within Transformers. This work builds on the mathematical framework of transformers introduced by (Elhage et al., 2021), in which each layer consists of a residual block that reads from and writes to a residual stream. Each layer includes attention heads and feedforward (FF) networks that process inputs before writing updates to the stream. Attention heads are responsible for transferring information between tokens through the self-attention mechanism, allowing each token to attend to others in the sequence. FF apply non-linear transformations independently to each token representation, enhancing the model's expressive capacity. The residual stream stores and propagates information across layers, enabling the integration of new outputs with existing representations while preserving original input information through residual connections.

Let a Transformer have L layers, a sequence of length K, batch size B, and inner activations X, with tensor shapes varying by model component as follows:

- Attention layers output: $X \in \mathbb{R}^{B\times K\times H_m}$, where H_m is the hidden dimension after projection.
- FF: $X \in \mathbb{R}^{B\times K\times H_f}$, where H_f is the feed-forward dimension.
- Residual stream: $X \in \mathbb{R}^{B\times K\times H_h}$, where H_h is the hidden dimension.

CAP Pooling Protocols. CAP can be applied at arbitrary layers using four aggregation functions α over the activation ranges:

- **Max**: selects the maximum activation in a segment, emphasising dominant features;
- **Mean**: computes the average activation, reflecting shared contributions;
- **Sum**: aggregates the total signal, preserving cumulative interactions;
- **Last Token** (lt): selects the final token's activation, mimicking autoregressive emphasis.

Formally, let $\mathcal{S} = \{(s_1, e_1), \ldots, (s_n, e_n)\}$ be the set of token spans corresponding to syntactic units (e.g., words or phrases). CAP replaces activations within each span $[s_i, e_i]$ with a pooled vector $\alpha([s_i, e_i])$, reducing sequence length from K to G, where

$$G = K - \Sigma_{i=1}^{n}(e_i - s_i) \tag{1}$$

Then the CAP functions are defined as:

$$\textbf{Sum:} \quad \alpha([s_i, e_i]) = \sum_{t=s_i}^{e_i} X[t] \tag{2}$$

$$\textbf{Mean:} \quad \alpha([s_i, e_i]) = \frac{1}{e_i - s_i + 1} \sum_{t=s_i}^{e_i} X[t] \tag{3}$$

$$\textbf{Max:} \quad \alpha([s_i, e_i]) = \max_{t\in[s_i,e_i]} X[t] \tag{4}$$

$$\textbf{Last Token } (lt)\textbf{:} \quad \alpha([s_i, e_i]) = X[e_i] \tag{5}$$

Post-CAP, the activation shape changes accordingly:

- For attention layers output, $X \in \mathbb{R}^{B\times K\times H_m}$ becomes $X \in \mathbb{R}^{B\times G\times H_m}$.

- For FF, $X \in \mathbb{R}^{B \times K \times H_f}$ becomes $X \in \mathbb{R}^{B \times G \times H_f}$.

- For residual stream:, $X \in \mathbb{R}^{B \times K \times H_h}$ becomes $X \in \mathbb{R}^{B \times G \times H_h}$.

This process consolidates activations for each syntactic unit, enabling systematic evaluation of compositional robustness across layers. For simplicity, we demonstrate the operation over these components, but this approach can be extended to any transformer's components, provided that the dimensional requirements for information flow, as described in (Elhage et al., 2021), are respected. For example, consider attention layer internal activations of shape $X \in R^{B \times H_a \times K \times K}$, where H_a is the number of attention heads, and K represents the query and key token dimensions. Applying CAP with the **Sum** protocol involves aggregating activations over the query range $[s_i, e_i]$ and the key range $[s_j, e_j]$. The grouped activations are computed as:

$$\alpha_{b,h}([s_i, e_i], [s_j, e_j]) = \sum_{t=s_i}^{e_i} \sum_{t'=s_j}^{e_j} X[b, h, t, t'].$$

After CAP, the grouped activations have shape $X \in \mathbb{R}^{B \times H_a \times G \times G}$, where G is the number of grouped syntactic units. This ensures that query–key interactions are consolidated into cohesive units, aligning activations with higher-level linguistic structures. Analysis of CAP's removal ratio ($K \rightarrow G$) is in Appendix C, and effects on positional encodings are detailed in Appendix B.4.

Evaluation. We evaluate CAP only on examples correctly answered by the model pre-intervention, ensuring that any change reflects the effect of compositional perturbation. For each model–task pair, we report:

- **Original Accuracy** (A_o), which represents model performance prior to CAP and serves as a baseline reference.

- **Grouped Accuracy** (A_c): performance after CAP, reported separately for each pooling protocol and also averaged providing a concise robustness summary.

- **Accuracy Drop** ($\Delta A = A_o - A_c$), which quantifies the performance degradation caused by compositional perturbation.

Smaller ΔA indicates greater robustness. Together, these metrics show how models encode and integrate constituents across layers, where higher A_c and lower ΔA reflect stronger resilience to compositional restructuring.

4 Empirical analysis

4.1 Experimental setup & datasets

Datasets and metrics. The CAP effect is evaluated using three WordNet-derived datasets—definitions, hypernyms, and synonyms—corresponding to the IDM, HP, and SP tasks (Fellbaum, 1998). Evaluation is restricted to test examples that the original models answered correctly (A_o), ensuring that any change reflects the effect of CAP rather than model error. Post-CAP accuracy (A_c) is reported for this subset, and the accuracy drop ($\Delta A = A_o - A_c$) is calculated for each pooling protocol. Appendix B.2 gives dataset details, and Appendix E.3 reports comprehensive results.

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), Gemma1 (2B parameters) (Team et al., 2024), Llama (3B, and 8B parameters) (Dubey et al., 2024), and Qwen (0.5B, 1.5B, and 3B parameters) (Yang et al., 2024). These models use different tokenisation approaches: byte-level BPE (GPT-2, Qwen), expanded BPE with 128K vocabulary (Llama3), and SentencePiece (Gemma). Models were tested before and after task-specific fine-tuning (3 epochs, learning rate 5e-5). This selection spans diverse architectures, sizes, and tokenisation strategies. Appendix B.3 provides model details 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).

Model	Layer Position	Original				Fine-tuned			
		Max ↓	Mean ↓	Sum ↓	*lt* ↓	Max ↓	Mean ↓	Sum ↓	*lt* ↓
GPT2-large	1%	8.06%	9.15%	6.70%	**23.10%**	10.61%	10.01%	7.83%	**19.77%**
	25%	5.19%	4.94%	5.63%	**19.43%**	6.25%	5.77%	6.32%	**16.85%**
	75%	5.28%	2.62%	2.39%	**6.31%**	3.66%	1.62%	0.88%	**5.59%**
	100%	**0.84%**	0.12%	0.19%	0.75%	**0.22%**	0.16%	0.16%	0.13%
Gemma-2B	1%	97.91%	23.51%	23.75%	**98.46%**	57.58%	22.70%	21.99%	**83.15%**
	25%	86.32%	16.20%	19.27%	**88.50%**	50.45%	14.08%	15.57%	**50.48%**
	75%	52.38%	31.03%	24.74%	**62.65%**	21.77%	14.99%	12.80%	**28.58%**
	100%	6.87%	**10.61%**	**10.61%**	8.27%	2.21%	2.05%	2.05%	**4.58%**
Qwen-3B	1%	12.63%	12.27%	11.44%	**14.94%**	7.85%	6.71%	6.48%	**11.42%**
	25%	**18.61%**	8.59%	9.11%	17.89%	10.66%	4.75%	5.82%	**11.59%**
	75%	**7.23%**	4.00%	3.79%	5.30%	**3.65%**	2.83%	1.85%	2.68%
	100%	0.39%	0.4%	0.4%	0.54%	**0.31%**	0.17%	0.2%	**0.33%**
Llama3-8B	1%	**25.49%**	24.99%	24.94%	**99.97%**	24.44%	23.42%	23.48%	**98.21%**
	25%	20.02%	5.87%	5.74%	**94.57%**	8.81%	6.03%	5.92%	**91.33%**
	75%	7.31%	3.40%	3.54%	**60.40%**	5.16%	3.47%	3.29%	**30.50%**
	100%	2.80%	1.77%	1.77%	**8.41%**	1.55%	1.33%	1.33%	**4.73%**

Table 1: IDM accuracy drop Δ in the TW-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.

4.2 Results and discussion

Compositional inference in LLMs is not a purely incremental process. Contrary to the expectation of a steady layer-wise improvement, we observe sharp fluctuations in performance when CAP is applied across layers. Accuracy drops notably in early and middle layers before partially recovering, and then declines again (Fig. 2a–c, e–f), suggesting that these layers struggle to process pooled linguistic features, particularly those formed in earlier layers. *Rather than progressively building semantic information from tokens to phrases, the models appear to rely on token-level features or combine tokens in ways that do not align with natural linguistic constituency. These results indicate that compositional abstraction is not constructed in a smooth, incremental fashion, but rather is distributed across multiple layers.*

An important distinction arises between TW-CAP, which groups tokens according to model-specific tokenisation, and TP-CAP, which applies externally parsed syntactic structures. While TP-CAP introduces richer constituent information, it may not align with the model's internal segmentation or syntactic reasoning. This misalignment is not a limitation of CAP, but rather a diagnostic signal: if LLMs encoded localised human-like syntax, TP-based grouping should be minimally disruptive. The observed sensitivity under TP-CAP, therefore, suggests that LLMs do not reliably internalise hierarchical syntactic structures, but instead prioritise token-level features over constituent-based abstraction.

The results also show that attention and information flow are distributed across tokens and layers in a highly context-dependent manner, rather than following sequential or constituent-based order. This effect is most pronounced in the SP and HP tasks, where TP-CAP results in sharp performance drops due to limited contextual redundancy. We interpret this as a consequence of the training objective: transformers maximise predictive information at each step, which encourages token-level throughput but reduces mutual information between tokens within a single layer. As a result, *aggregation, including syntactic grouping, appears to be distributed across multiple layers rather than localised to any single one*. These findings highlight that compositional structures are highly sensitive to token representation dynamics across layers, and that performance fluctuations under CAP can be understood as information loss linked to reduced redundancy between tokens. An information-theoretical analysis elaborates this reasoning in Section 5.

Larger models are more fragile to compositional perturbations. The IDM task highlights this fragility in larger models, as they rely on finer

feature extraction. For instance, original Qwen's smaller variants are more robust than larger ones (at 25% depth, Qwen-1.5B shows a 10.03% drop vs. 13.55% for Qwen-3B), and Llama3-3B is more vulnerable than Llama3-8B. Despite having similar reduction ratios to Llama models (Appendix C), Gemma-2B shows sharper declines (e.g., at 1% depth with Max pooling: 97.91% vs. 25.49% for Llama3-8B). This is likely due to Gemma's larger vocabulary, which produces finer-grained tokenisation and amplifies early-layer sensitivity. Interestingly, last-token pooling narrows these differences: Gemma-2B and Llama3-8B perform similarly (98.46% vs. 99.97% at 1%), suggesting that although the two architectures may distribute information differently across token representations, both show limited robustness to compositional aggregation in early layers, where reliance on positional cues dominates.

Fine-grained token knowledge benefits standard tasks; it appears to increase susceptibility to compositional perturbations. The superior performance of Llama3-8B over its 3B variant can be attributed to its enhanced capacity for maintaining feature relationships across layers while preserving key compositional information. While larger models excel in standard tasks (Appendix E.1), *they exhibit a greater reliance on the identification of intrinsic features in the early layers.* We find that TP-CAP substantially impacts Gemma-2B and Llama models, suggesting a heavy dependence on layer-wise information gain, where they separate features in an uncorrelated and highly distinct manner. While this aids in identifying complex feature patterns, it also makes them more vulnerable to contextual noise, *a weakness that threatens their robustness and integrity*. Notably, Qwen models outperform Llama and Gemma despite similar parameter counts, likely due to byte-level BPE tokenisation and multilingual training, which enhance compositional stability, whereas Llama's expanded BPE and Gemma's SentencePiece prioritise efficiency over phrase retention, increasing vulnerability to CAP interventions.

Activation abstraction vs the information loss. Table 1 shows strong variation in aggregation performance across models for the IDM task (full results in Appendix E.3). The last-token aggregation shows the most dramatic impact, followed by the max aggregation, signifying that reliance on the final token as a semantic proxy may be overstated: while effective for specific entity types (e.g. named entities or numbers), it fails for general natural language compositional tasks. In IDM, where semantic integration is critical, last-token pooling produces drops exceeding 90% in early layers across models, fundamentally challenging the widespread assumption in current literature that final token positions serve as reliable semantic aggregators in autoregressive models. The Max aggregation also shows substantial impact across models, *supporting our argument that information is fragmented across tokens and layers rather than integrated into stable compositional units.* Mean aggregation produces more moderate declines (e.g. 4.44% in GPT-2-small to 31.03% in Gemma-2B at deeper layers), but still indicates *absence of consistent compositional mechanisms.* By contrast, Sum aggregation consistently outperforms other methods, particularly in original models. It 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 cumulative information loss, particularly when interventions occur in early layers.*

Fine-tuning enhances recovery capabilities across models. Fig. 2 (d–f) shows that fine-tuning improves performance across all model families, with the strongest gains in later layers (75–100%). SP tasks showed maximum benefit, attributed to high task specificity and minimal activation reduction under CAP. Max aggregation displays the largest improvements, implying enhanced retention of key information post-fine-tuning. For instance, Gemma-2B's accuracy drop decreased from 97.91% to 57.65% in the 1% layer, while Qwen-3B improved from 7.23% to 3.65% in the 75% layer. Mean aggregation also shows substantial gains in smaller models (e.g. Gemma-2B's drop at 75% reduced from 31.03% to 15.00%). The Qwen family improves consistently across all protocols, while GPT-2-large shows minimal gains, possibly due to overfitting. Notably, larger models, such as Llama3-8B, exhibit limited improvement in IDM tasks, suggesting that standard fine-tuning objectives may not directly enhance compositional robustness. Overall, fine-tuning enhances resilience under CAP but does not resolve the deeper challenge of forming stable compositional representations, pointing to an architectural rather than training limitation.

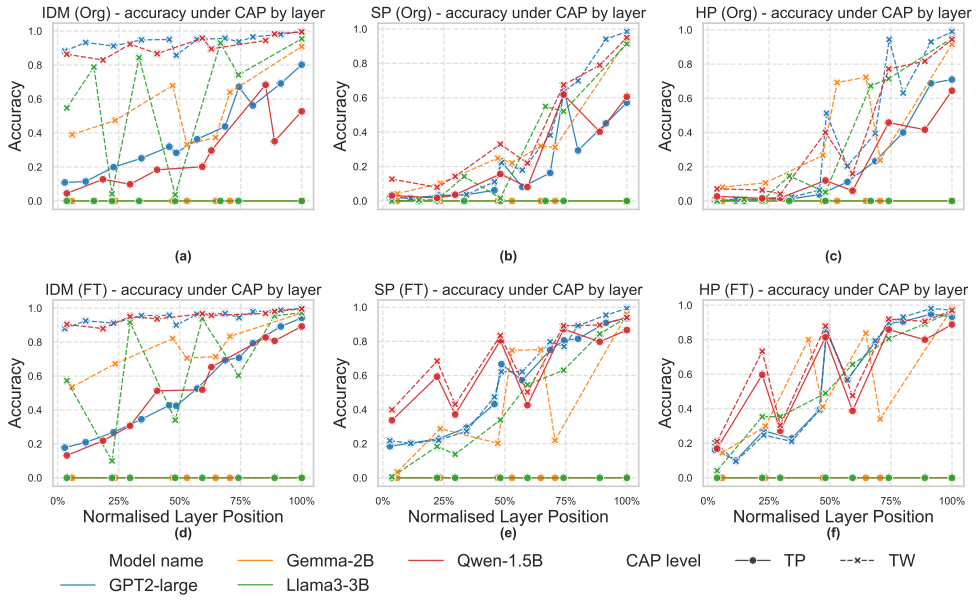

Figure 2: Average grouped accuracy of CAP across different aggregation functions for normalised layer positions (0%-100%) is shown for word-level CAP (TW) and phrasal-level CAP (TP). Sub-figures (a)-(c) illustrate the CAP effect on the original (Org) models, while sub-figures (d)-(f) show its impact 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.

5 Information Gain & Token Mutual Information

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

Predictive information. $\mathrm{IG}_{Y,R_l(X)}$ quantifies the amount of information gained about the predicted token Y from observing $R_l(X)$. Its expectation is the mutual information $I(Y; R_l(X))$ between Y and $R_l(X)$, which is equivalent to the reduction in entropy of Y achieved by learning the state of $R_l(X)$:

$$\mathrm{IG}_{Y,R_l(X)} = H(Y) - H(Y \mid R_l(X)) = I(Y; R_l(X)). \tag{6}$$

During training, the model minimises $H(Y \mid R_L(X))$ at the final layer L, equivalently maximising $I(Y; R_L(X))$. Earlier layers are not directly optimised to maximise predictive information, but instead propagate token-level features forward until they can be integrated at deeper layers.

Mutual information across layers. Consider the sequence of representations $R_{l_p}(X) \to R_{l_q}(X) \to R_{l_r}(X)$ for $p < q < r$. By the data processing inequality,

$$I(R_{l_p}(X); R_{l_r}(X)) \leq I(R_{l_p}(X); R_{l_q}(X)). \tag{7}$$

This inequality reflects that later representations cannot contain more mutual information with an earlier one than an intermediate representation does. In practice, information is progressively re-encoded: redundancy across tokens is reduced while features relevant for predicting Y are preserved and concentrated.

Implications for compositionality. If two token representations $R_l(X_i)$ and $R_l(X_j)$ share high mutual information, then aggregating them has little effect on predictive information $I(Y; R_l(X))$, as their signals are largely redundant. However, common tokenisation schemes (e.g. BPE, WordPiece) explicitly minimise redundancy between adjacent tokens to reduce vocabulary size, i.e. they minimise $I(X_i; X_j)$. This forces compositional dependencies to be reconstructed through aggregation paths that span multiple layers. The model therefore has an intrinsic incentive to *delay integration*: earlier layers prioritise token-level throughput, while later layers gradually compose these signals.

This deferred-integration perspective explains

why CAP interventions in early layers lead to the strongest performance degradation: they enforce aggregation before the model has had the opportunity to distribute and re-integrate token information across depth. As a result, compositional robustness depends on long-range aggregation paths rather than localised integration within any single layer, consistent with mechanistic interpretability findings (Elhage et al., 2021; Conmy et al., 2023).

6 Related work

Compositionality, the idea that complex meaning arises from combining simpler parts, is central to linguistics, cognitive science, and AI (Fodor, 1975; Montague and Thomason, 1975; Tull et al., 2024). In neural models, compositionality is key to generalisation and interpretability, yet remains difficult to diagnose or enforce (Donatelli and Koller, 2023).

Several studies explore how and where compositional structure emerges in Transformers. At the representation level, Carvalho et al. (2025) found weak integration of adjective–noun semantics; Haslett (2024) reported limited morphological segmentation, especially in non-Latin scripts; and Yu and Ettinger (2020) observed that transformers mainly encode individual word content rather than phrasal meaning. DecompX (Modarressi et al., 2023) traced token representations layer by layer, finding only partial semantic merging. Together, these results suggest a brittle and distributed encoding of composition.

Other works examine dynamics across layers. The logit lens (Nostalgebraist, 2020) showed how early layers make rough predictions while deeper layers refine them. Dai et al. (2022) characterised FF layers as key–value memories storing compositionally useful chunks. Petty et al. (2024) observed greater compositional behaviour in deeper models, though gains saturate with depth. Intervention methods, such as MEMIT (Meng et al., 2023) and PMET (Li et al., 2025), demonstrate that structured inferences can be elicited by directly manipulating internal representations.

Recent work highlights reliance on last-token positions: Feucht et al. (2024) showed erasure effects for named entities, and Heinzerling and Inui (2024) found similar patterns for numeric properties. Kamoda et al. (2025) analysed GPT-2's first-layer attention, showing that detokenisation often depends on biases toward adjacent tokens. While these studies focus on specific token types or shallow layers, our method examines compositional pooling broadly across constituents. We find that reliance on final-token information is brittle under structured aggregation, challenging its adequacy as a general-purpose compositional strategy.

Compositional generalisation has also been studied in synthetic or constrained settings (Hupkes et al., 2020; Lake and Baroni, 2018), where controlled grammars obscure naturalistic complexity. Even naturalistic studies often target task outputs rather than internal structure (Kim and Linzen, 2020). Others highlight instability: Dankers et al. (2022) showed that apparent compositionality varies with framing and data distribution, and Dziri et al. (2023) found that LLMs often rely on pattern matching rather than true decomposition in multi-step reasoning.

In contrast, CAP probes compositionality at the level of hidden activations. By merging token representations into word- and phrase-level units at arbitrary layers, CAP tests where and how composition is integrated. Unlike output-based probing, which relies on correlations, CAP intervenes directly in the model's internal state, providing a causal diagnostic of compositionality and exposing the robustness and locality of abstraction across layers.

7 Conclusion

We introduced CAP as a structured intervention for systematically probing how transformer-based LLMs process compositional meaning. CAP pools token activations into linguistic constituents at different depths, directly testing whether semantic composition is localised, robust, or deferred.

Our analysis reveals that compositional meaning is not hierarchically or locally built within individual layers, but instead distributed across depth through long-range aggregation. Perturbations in early layers severely degrade performance, especially in larger models, highlighting the fragility of premature integration. This supports an information-theoretic view in which transformers prioritise token-level throughput and delay integration, preventing stable compositional representations from forming.

CAP provides a principled diagnostic for revealing this weakness, and future work should explore architectural or training objectives that encourage hierarchical abstraction, as well as extensions to other model types and languages.

Limitations

Several limitations are acknowledged in our paper. First, the WordNet dataset may not fully represent language diversity across all domains. Second, the employed transformer models are decoder-based only and could be subject to biases from their training data. Third, our findings depend on the Benepar parsing model, which may introduce particular biases in linguistic analyses. Additionally, while our tasks provide an indirect signal of meaning preservation, incorporating explicit reconstruction tasks in future work could offer complementary insight into how CAP affects the retention of input-level information. Finally, the applicability of our results to other languages has not been tested. Expanding CAP to multilingual settings and testing with alternative parsers or models trained with different positional encodings would further validate the generality of 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. 2025. Montague semantics and modifier consistency measurement in neural language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5515–5529.

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.

Verna Dankers, Elia Bruni, and Dieuwke Hupkes. 2022. The Paradox of the Compositionality of Natural Language: A Neural Machine Translation Case Study. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4154–4175, Dublin, Ireland. Association for Computational Linguistics.

Lucia Donatelli and Alexander Koller. 2023. Compositionality in computational linguistics. *Annual Review of Linguistics*, 9(Volume 9, 2023):463–481.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, and 1 others. 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.

Sheridan Feucht, David Atkinson, Byron Wallace, and David Bau. 2024. Token erasure as a footprint of implicit vocabulary items in llms. *arXiv preprint arXiv:2406.20086*.

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. 2024. How much semantic information is available in large language model tokens? Preprint available on OSF.

Benjamin Heinzerling and Kentaro Inui. 2024. Monotonic representation of numeric properties in language models. *arXiv preprint arXiv:2403.10381*.

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.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2020. Compositionality decomposed: How do neural networks generalise? (extended abstract). In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 5065–5069. International Joint Conferences on Artificial Intelligence Organization. Journal track.

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

Go Kamoda, Benjamin Heinzerling, Tatsuro Inaba, Keito Kudo, Keisuke Sakaguchi, and Kentaro Inui. 2025. Weight-based analysis of detokenization in language models: Understanding the first stage of inference without inference. *arXiv preprint arXiv:2501.15754*.

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

Najoung Kim and Tal Linzen. 2020. COGS: A compositional generalization challenge based on semantic interpretation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9087–9105, Online. Association for Computational Linguistics.

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.

Brenden Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR.

Xiaopeng Li, Shasha Li, Shezheng Song, Jing Yang, Jun Ma, and Jie Yu. 2025. Pmet: precise model editing in a transformer. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

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.

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 *Proceedings 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).

Nostalgebraist. 2020. Interpreting gpt: The logit lens. LessWrong.

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

Jackson Petty, Sjoerd Steenkiste, Ishita Dasgupta, Fei Sha, Dan Garrette, and Tal Linzen. 2024. The Impact of Depth on Compositional Generalization in Transformer Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7239–7252, Mexico City, Mexico. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, and 1 others. 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, and 1 others. 2023. Are large language models really robust to word-level perturbations? In *Socially Responsible Language Modelling Research Workshop*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

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 formalised 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:

$$\text{Syntax}(s) = f\left(\text{Syntax}(s_1), \text{Syntax}(s_2), \ldots, \text{Syntax}(s_n), \text{Rules}\right) \tag{8}$$

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:

$$\mathbf{v}_s = f\left(\mathbf{v}_{s_1}, \mathbf{v}_{s_2}, \ldots, \mathbf{v}_{s_n}, \text{Syntactic structure}\right) \tag{9}$$

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, feedforward 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 *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:

$$T = \arg\max_{t \in \mathcal{V}} P(t \mid D) \tag{10}$$

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 *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:

$$S = \arg\max_{s \in \mathcal{V}} P(s \mid W) \tag{11}$$

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 *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:

$$H = \arg\max_{h \in \mathcal{V}} P(h \mid W) \tag{12}$$

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."

Exact Word Reconstruction (EWR): The EWR task involves recovering an original sentence $S = (w_1, w_2, \ldots, w_n) \in \mathcal{V}^n$, given a corrupted version $C = (c_1, c_2, \ldots, c_m) \in \mathcal{V}^m$, where $\mathcal{V}$ is the vocabulary and typically $m < n$. The corruption may involve masking, deletion, or replacement of one or more spans in S, in our paper we revert to masking. The model receives C as input and is evaluated on its ability to reconstruct the original sequence S, producing an output $\hat{S} \in \mathcal{V}^n$ such that:

$$\hat{S} = \arg\max_{s \in \mathcal{V}^n} P(s \mid C) \tag{13}$$

A prediction is considered correct if and only if $\hat{S} = S$ — that is, the reconstructed output must match the original sentence exactly in both token content and order.

The EWR task provides a stringent evaluation of a model's ability to infer missing linguistic information, maintain structural coherence, and recover semantically faithful output under partial or noisy input conditions.

Exact Sequence Autoencoding (ESA): The ESA task involves reconstructing an original sentence $S = (w_1, w_2, \ldots, w_n) \in \mathcal{V}^n$, where $\mathcal{V}$ is the vocabulary. The model receives the uncorrupted input sequence S and is trained or evaluated to produce an output sequence $\hat{S} \in \mathcal{V}^n$, such that:

$$\hat{S} = \arg\max_{s \in \mathcal{V}^n} P(s \mid S) \tag{14}$$

A prediction is considered correct if and only if $\hat{S} = S$ — that is, the output must exactly match the input sequence in both content and order, with no insertions, deletions, or substitutions.

This task serves as a controlled diagnostic to assess a model's capacity to encode, preserve, and reproduce linguistic structure without any form of input corruption or external noise.

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. It is focused on general-purpose vocabulary and does not target

Model	Task	Original Test Set	Fine-tuned Test Set
GPT2 (S,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 (3B, 8B)	IDM	14,991	10,828
	SP	9,360	6,723
	HP	31,962	23,070
Qwen2.5 (0.5B, 1.5B, 3B)	IDM	14,927	10,780
	SP	9,195	6,598
	HP	31,845	23,000

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	124M	12	768	12	GELU	3072
GPT2-large	708M	36	1280	20	GELU	5120
Gemma-2B	2B	32	4096	16	GELU	8192
LLama3-3B	3.2B	28	3072	24	SiLU	8192
LLama3-8B	7.8B	32	4096	32	SiLU	14336
Qwen2.5-0.5B	391M	24	896	14	SiLU	4864
Qwen2.5-1.5B	1.4B	28	1536	12	SiLU	8960
Qwen2.5-3B	3.0B	36	2048	16	SiLU	11008

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.

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-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%
Llama3-3B	20.51%	8.26%	12.19%	26.42%	13.43%	31.1%
Qwen-0.5B	8.21%	6.10%	12.03%	18.83%	10.94%	28.03%
Qwen-1.5B	12.35%	7.61%	14.64%	30.01%	13.70%	31.31%
Qwen-3B	13.35%	7.53%	14.40%	31.80%	13.66%	31.95%

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

Table 3 provides a comparative overview of various Transformer models used in this study. We used GPT2 models (released under the Modified MIT License), Gemma-2B (released under the Gemma Terms of Use), Llama3 models (released under the Meta Llama 3 Community License), and Qwen models (released under Apache License 2.0). The used models were mainly pre-trained on English data, with Qwen and LLama models providing additional multilingual support, which is English, German, French, Italian, Portuguese, Hindi, Spanish, and Thai for LLama, and more than 10 languages, including Chinese, English, French, Spanish, Portuguese, Russian, Arabic, Japanese, Korean, Vietnamese, Thai, and Indonesian for Qwen. All models were used for research purposes, specifically for language modelling 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 three 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.

B.4 Handling of Sequence Reduction and Positional Encoding in CAP

CAP reduces the number of token-level activations from the original input length K to a shorter grouped sequence length G, by merging activations corresponding to word-level or phrasal-level constituents. This reduction is applied post-token embedding and affects intermediate activations within the transformer, specifically the outputs of residual blocks or their internal components (e.g., attention

or feedforward sublayers). From the point of CAP application onward, the model processes a reduced-length sequence of size G. This operation does not alter the model's input embeddings or positional encodings.

Effect of Positional Encoding Schemes. The impact of this reduction depends on the positional encoding strategy used by the model: (i) **GPT2 models** use *Sinusoidal positional embeddings*, where each position index corresponds to a unique learned embedding. While CAP does not alter these embeddings directly, reducing the sequence length at intermediate layers can introduce misalignment between positional indices and semantic content. This may disrupt downstream attention or feedforward computations that assume consistent positional context; (ii) **LLaMA, Qwen, and Gemma** models use *rotary positional encodings (RoPE)*, which encode position relationally through rotation in embedding space. These relative encodings are more robust to changes in sequence length, and CAP has a milder impact on positional semantics in these models. Nevertheless, changes in sequence structure may still affect how models integrate cross-token context.

Although CAP does not interfere with the model's input or positional embedding layer, it alters the spatial structure of activations mid-forward pass. This may influence how transformers aggregate information across positions, especially in models with absolute position encoding. Nevertheless, we did not observe severe performance degradation in those models compared to others. We acknowledge this as a potential contributing factor to the observed degradation under CAP and consider it an important area for future study.

Namely, Embedding-level analysis represents a promising direction for future exploration. Although this work evaluates a wide range of models with differing positional encoding schemes, we acknowledge the need for more targeted analysis of how CAP interacts with these embeddings. In particular, it would be valuable to quantify the impact of CAP under controlled conditions that isolate embedding effects. For instance, experiments using fixed or masked positional encodings, or applying CAP to models trained from scratch with alternative positional schemes, could help disentangle the influence of compositional pooling from that of positional structure.

Model	Task	Mean ± Std
GPT2 (S)	IDM	3 ± 5
	SP	27 ± 9
	HP	27 ± 10
GPT2 (L)	IDM	3 ± 5
	SP	27 ± 9
	HP	26 ± 11
Gemma-2B	IDM	9 ± 4
	SP	19 ± 9
	HP	30 ± 9
Llama3-3B	IDM	10 ± 5
	SP	23 ± 6
	HP	28 ± 6
Llama3-8B	IDM	10 ± 5
	SP	21 ± 7
	HP	28 ± 9
Qwen 0.5B	IDM	3 ± 5
	SP	9 ± 11
	HP	20 ± 10
Qwen 1.5B	IDM	3 ± 5
	SP	12 ± 10
	HP	19 ± 10
Qwen 3B	IDM	3 ± 5
	SP	12 ± 10
	HP	19 ± 10

Table 5: Reduction percentages.

C Token Reduction Analysis

Table 5 presents an analysis of activation reduction percentages across different LLMs, particularly for the token-to-words case. In this context, the mean represents the average reduction percentages across samples, while the standard deviation indicates the variability of these reductions. While models within a family (e.g., Qwen) share the same tokeniser and vocabulary, the reduction percentages still vary across tasks (e.g., SP vs. HP) because different tasks involve input definitions or prompts with different average sentence lengths and syntactic complexity, which in turn affect how many groupings are formed under CAP. In other words, although the tokeniser is fixed, the number and size of groupable units (e.g., multi-token words or phrases) are input-dependent. The purpose is to assess whether token reduction across models would highly influence the results of CAP.

Token reduction is a factor but not the sole determinant of performance degradation. The results presented in Tables 10, 11, and 12 indicate that while token reduction percentage influences performance degradation, it is not the sole determining factor. Several key observations support this conclusion, which is discussed below.

First, we observe that higher token reduction does not always lead to a greater performance drop. For instance, models such as Gemma-2B and

Llama3-8B exhibit high token reduction percentages (Table 5), yet their performance degradation varies significantly across tasks and layer positions. Also, despite lower token reduction percentages, the models Qwen 0.5B and GPT2-small still show substantial accuracy drops, particularly in early layers in the SP and HP tasks. Second, model size and depth influence degradation, as evident in the larger models (e.g., Llama3-8B, Gemma-2B) exhibiting greater fragility to CAP interventions, particularly in early layers (1% and 25%). Third, as discussed in the paper, layer-specific variability suggests hierarchical processing differences. Early-layer CAP interventions cause severe accuracy drops in large models but have a less pronounced effect in smaller models, suggesting that deeper architectures defer compositional integration to later layers. Further, fine-tuning reduces degradation in later layers (75% and 100%), implying that learned representations in deeper layers mitigate the effects of early perturbations. Finally, architectural differences influence sensitivity. We observe that higher MLP dimensions (e.g., Llama3-8B: 14,336 vs. GPT2-small: 3,072) correlate with greater vulnerability to CAP perturbations, likely due to increased parameter redundancy and disruption of the key-value recall mechanism in MLPs (Meng et al., 2022).

While the token reduction percentage contributes to performance degradation, it is insufficient to fully explain the observed variations. Task nature, model size, layer depth, activation functions, and MLP dimensions collectively influence the robustness of CAP interventions. Larger, deeper models demonstrate greater sensitivity to early perturbations, while fine-tuning helps recover performance in later layers. These findings suggest that effective compositional representations in LLMs are distributed rather than localised, requiring specialised architectures or training objectives to improve robustness.

D Evaluating Parsing Accuracy and Addressing the Impact of Benepar Parser Errors

A key potential bias in our results comes from the reliance on the constituency parser for token-to-phrase experiments. Inaccuracies in parsing may distort the results of CAP. To address this, we report the chosen parser's accuracy by testing it on the Stanford Sentiment Treebank (SST) dataset, a dataset that offers golden labels for parsing. We aim to alleviate concerns about the parser's impact on our findings by showcasing its accuracy on the SST dataset. The parser evaluation was conducted as follows:

Dataset. A subset of 1,000 randomly sampled sentences from the test split of the SST dataset was used for the analysis. The Stanford Sentiment Treebank (SST) provides annotated constituency labels, which serve as the golden labels for comparison with parser outputs. While WordNet definitions offer rich semantic information, they lack annotated golden constituency labels, making direct parser validation infeasible. The use of SST's annotations enables reliable parser evaluation and indirectly supports the validation of the parsing correctness for WordNet definitions, provided they follow standard syntactic structures.

Parser. The Benepar parser was employed for parsing sentences due to its strong performance in constituency parsing tasks. Benepar is widely recognised for its robustness and ability to handle diverse syntactic structures. For this evaluation, the constituency structures generated by Benepar were directly compared against SST's golden annotations to assess its parsing accuracy.

Evaluation metrics. The parser's performance was evaluated using the following metrics: (i) Precision: Proportion of correctly predicted constituents out of all predicted constituents; (ii) Recall: Proportion of correctly predicted constituents out of all ground truth constituents; (iii) F1-Score: Harmonic mean of precision and recall, providing an overall performance measure; and (iv) Accuracy: Percentage of sentences where the predicted constituency structure fully matches the ground truth.

Results robustness. To ensure robustness and consistency, the evaluation was repeated across five different random seeds. This allowed for an assessment of variability in performance across multiple subsets of the dataset. Additionally, constituents were evaluated at hierarchical levels—such as root level, phrase level, and token level—to analyse parsing performance across varying syntactic granularities.

Results. The evaluation yielded the following averaged metrics across five seeds for the default level of parsing (Level 1, the immediate children of the root node):

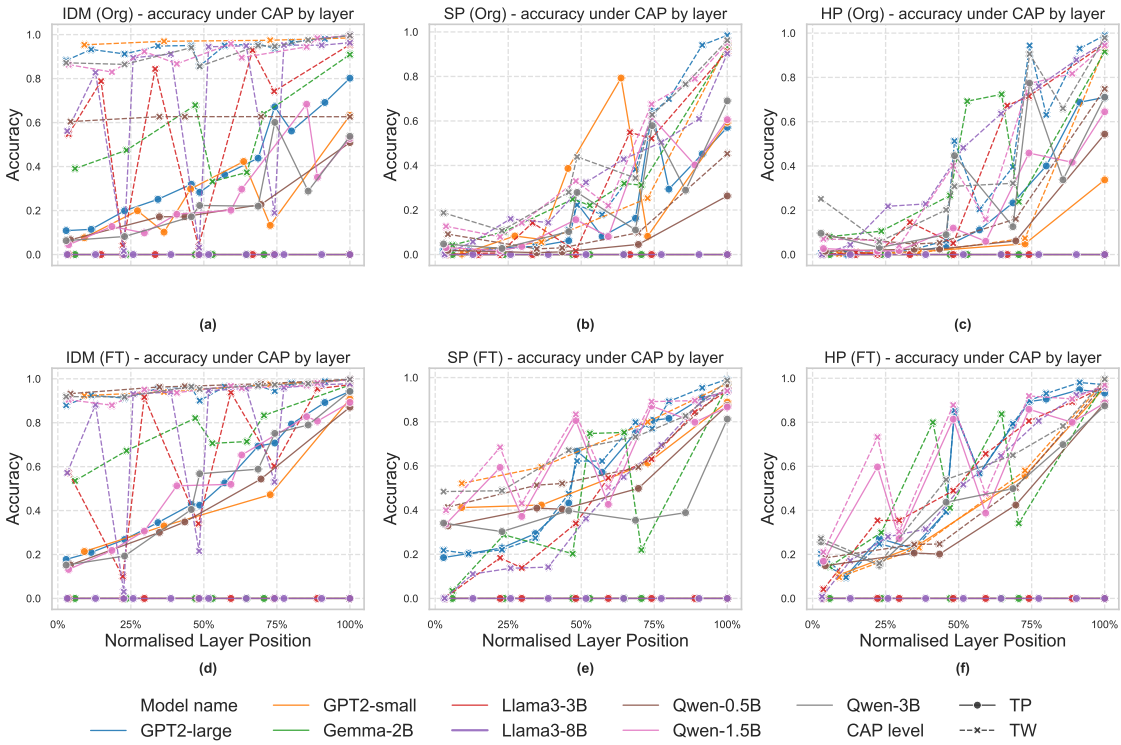

Figure 3: Average grouped accuracy of CAP across different aggregation functions for normalised layer positions (0%-100%) is shown for word-level CAP (TW) and phrasal-level CAP (TP). Sub-figures (a)-(c) illustrate the CAP effect on the original (Org) models, while sub-figures (d)-(f) show its impact 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.

Metric	Mean ± Std
Precision	0.956 ± 0.001
Recall	0.956 ± 0.001
F1-Score	0.956 ± 0.001
Accuracy	0.956 ± 0.001

Table 6: Aggregated evaluation metrics for Level 1 constituents using the Benepar parser, averaged across five seeds.

Interpretation. The results demonstrate consistently high parsing accuracy across all evaluation metrics, with minimal variability (as indicated by the low standard deviation). These findings validate the Benepar parser's reliability for parsing Level 1 constituents, which form the backbone of sentence structure. Consequently, the parser's impact on CAP results is minimal, ensuring robustness and validity of our conclusions. We note that while SST provides gold-standard constituency annotations, WordNet definitions may differ syntactically from SST sentences. However, both consist of standard English declarative sentences, suggesting the parser's high accuracy on SST likely generalises to our dataset.

E Detailed Performance Evaluation and Results

E.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 FT 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.

E.2 Qualitative Analysis of CAP-Induced Prediction Shifts.

Tables 7 and 8 present representative examples of predictions from multiple models across all the tasks, before and after CAP is applied. These examples are drawn from inputs that the model originally predicted correctly, allowing us to isolate the effects of CAP perturbations without confounding them with unrelated model failures. Each example specifies the CAP layer, CAP type (token-to-word

Task / Input Prompt	Model	CAP Layer (Type)	Prediction (No CAP)	Prediction (W/ CAP)	Observation / Interpretation
IDM: *lacking embellishment or ornamentation is called a: "*	Qwen2.5-1.5B	Layer 8 (TW)	*plain*	*ornament*	Prediction shifts from correct to antonymic, likely due to token merging altering polarity.
IDM: *remaining after all deductions is called a: "*	LLaMA3.1-8B	Layer 4 (TW)	*net*	*gain*	Subtle financial distinction lost; CAP causes confusion between output and intermediate step.
IDM: *make an effort or attempt is called a:"*	Gemma-2B	Layer 1 (TP)	*try*	*<h1>*	Invalid token generation suggests breakdown in early compositional encoding.
IDM: *a formal series of statements showing that if one thing is true something else necessarily follows from it is called a:"*	GPT2-L	Layer 24 (TP)	*proof*	*form*	Loss of logical structure leads to a more abstract or vague concept.
SP: *"journal" is a synonym of*	Qwen2.5-1.5B	Layer 18 (TW)	*diary*	*di*	Output truncated, likely due to disruption in longer multi-token word embedding.
SP: *"get" is a synonym of*	Qwen2.5-0.5B	Layer 16 (TW)	*catch*	*break*	Semantic drift under CAP; verb meaning shifts from acquisition to interruption.
HP: *"voice" is a type of*	Gemma1-2B	Layer 16 (TW)	*sound*	*noise*	Precision reduced; CAP causes substitution with a noisier, less neutral concept.
HP: *"guama" is a type of*	LLaMA3.2-3B	Layer 12 (TW)	*tree*	*street*	The output reflects a contextual domain shift, likely due to token-level confusion post-CAP.

Table 7: Representative examples of model predictions with and without CAP applied at various layers. Examples highlight semantic degradation and conceptual drift caused by TW-CAP or TP-CAP applied to original models.

Task / Input Prompt	Model	CAP Layer (Type)	Prediction (No CAP)	Prediction (W/ CAP)	Observation / Interpretation
IDM: *prepare for eating by applying heat is called a: "*	GPT2-S	Layer 4 (TW)	*cook*	*heat*	CAP leads to a shift from action to cause, indicating surface-level generalisation.
IDM: *fail to attend an event or activity is called a: "*	LLaMA3.2-3B	Layer 1 (TW)	*miss*	*catch*	CAP appears to invert the meaning, suggesting confusion in early compositional buildup.
IDM: *general term for any insect or similar creeping or crawling invertebrate is called a:"*	Gemma-2B	Layer 11 (TP)	*bug*	*un*	Invalid token generation suggests breakdown in compositional encoding
IDM: *an institution of higher education created to educate and grant degrees often a part of a university is called a:"*	GPT2-S	Layer 1 (TP)	*college*	*regular*	CAP reduces specificity, misclassifying to a generic adjective.
SP: *"one fourth" is a synonym of*	Gemma1-2B	Layer 10 (TW)	*fourth*	*half*	CAP merges related quantities but loses precision, leading to broader but incorrect substitution.
HP: *"hotel" is a type of*	Qwen2.5-3B	Layer 16 (TW)	*building*	*room*	Shift from category to subcomponent suggests CAP disrupted higher-level abstraction.
HP: *"hexagon" is a type of*	Qwen2.5-3B	Layer 16 (TW)	*polygon*	*plane*	Hierarchical class (shape) replaced by domain (geometry); abstraction misaligned.

Table 8: Representative examples of model predictions with and without CAP applied at various layers. Each example shows the prompt, model, CAP configuration (layer and type), predictions, and qualitative interpretation. All examples applied to fine-tuned (FT) models.

Variant	Layer 25%			Layer 75%		
	Acc.	Δ vs TW	Trend	Acc.	Δ vs TW	Trend
TW-CAP	50.45%	—	structured	21.77%	—	structured
B-CAP	49.98%	-0.5 p.p.	same pattern	21.21%	-0.6 p.p.	same pattern
N-CAP	50.82%	+0.4 p.p.	same pattern	22.03%	+0.3 p.p.	same pattern
R-CAP	29.74%	-20.7 p.p.	uniform drop	13.26%	-8.5 p.p.	monotonic drop

Table 9: Comparison of grouped accuracies for Gemma-2B (fine-tuned) on the inverse-dictionary task. B-CAP and N-CAP closely track TW-CAP, whereas R-CAP collapses sharply, confirming that CAP degradation is not due to cumulative or random effects.

or token-to-phrase), and the model involved. Table 7 focuses on predictions made by original (non-fine-tuned) models, while Table 8 includes outputs from fine-tuned variants. Observed shifts include truncation of multi-token terms (e.g., "diary" → "di"), polarity inversion (e.g., "plain" → "ornament"), loss of abstraction ("polygon" → "plane"), and domain misalignment (e.g., "tree" → "street").

These qualitative differences provide interpretability insights that complement the aggregate metrics reported earlier. They reveal how CAP affects not only performance but the nature of model outputs, especially in terms of semantic generalisation, abstraction shifts, and lexical precision. While we do not observe a uniform trend across layers or model families, TP-CAP consistently induces more severe semantic degradation. This suggests that as model capacity increases, internal representations may become more sensitive to disruptions from externally imposed syntactic structures, potentially due, as argued in the main paper, to a stronger reliance on learned token-level dependencies that diverge from higher-level compositional groupings. This analysis highlights the nature of semantic and lexical shifts induced by CAP, reinforcing the need for future task-specific fine-tuning strategies that improve robustness to structured representation pooling.

E.3 Comprehensive CAP Results for All Models and Tasks

Fig. 3 and Tables 10- 14, and 15 present the reduction in accuracy when applying word-level and phrasal CAP, respectively, across models and the three tasks: IDM, SP, and 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:

$$\Delta A = A_o - A_c \tag{15}$$

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.

E.4 Quantitative Comparison Across Control Variants

Table 9 compares the grouped accuracies of CAP variants for **Gemma-2B** on the inverse-dictionary task (fine-tuned models). The baseline TW-CAP values correspond to those reported in Table 10. All control variants were evaluated under identical pooling and evaluation protocols.

B-CAP (broadcasted) and N-CAP (norm-preserved) produce nearly identical accuracies to TW-CAP across depths, differing by less than 1–2 percentage points (p.p.) at any layer. This indicates that neither sequence-length reduction nor activation-norm scaling explains the degradation. In contrast, R-CAP (random grouping) causes large, monotonic decreases, over 20 p.p. even at early layers—demonstrating that the structured TW-CAP pattern is not due to random perturbation or cumulative noise propagation.

At early layers (25 %), TW-CAP, B-CAP, and N-CAP all retain roughly half of baseline accuracy, while R-CAP falls by over 20 p.p. This divergence in early layers is critical: if degradation were cumulative or driven by sequence shortening, B-CAP

Model	Layer Position	Original				Fine-tuned			
		Max	Mean	Sum	lt	Max	Mean	Sum	lt
IDM (Inverse Dictionary Modelling)									
GPT2-small	1%	4.76%	4.44%	4.69%	23.23%	8.04%	7.72%	7.22%	14.42%
	25%	3.09%	2.74%	3.26%	23.15%	5.87%	5.85%	6.24%	12.47%
	75%	2.64%	2.36%	2.74%	9.29%	2.72%	2.47%	2.35%	6.08%
	100%	1.43%	1.24%	1.24%	2.08%	0.46%	0.39%	0.39%	1.53%
GPT2-large	1%	8.06%	9.15%	6.70%	23.1%	10.61%	10.01%	7.83%	19.77%
	25%	5.19%	4.94%	5.63%	19.43%	6.25%	5.77%	6.32%	16.85%
	75%	5.28%	2.62%	2.39%	6.31%	3.66%	1.62%	0.88%	5.59%
	100%	0.84%	0.12%	0.19%	0.75%	0.22%	0.16%	0.16%	0.13%
Gemma-2B	1%	97.91%	23.51%	23.75%	98.46%	57.58%	22.70%	21.99%	83.15%
	25%	86.32%	16.20%	19.27%	88.50%	50.45%	14.08%	15.57%	50.48%
	75%	52.38%	31.03%	24.74%	62.65%	21.77%	14.99%	12.80%	28.58%
	100%	6.87%	10.61%	10.61%	8.27%	2.21%	2.05%	2.05%	4.58%
Llama3-8B	1%	25.49%	24.99%	24.94%	99.97%	24.44%	23.42%	23.48%	98.21%
	25%	20.02%	5.87%	5.74%	94.57%	8.81%	6.03%	5.92%	91.33%
	75%	7.31%	3.40%	3.54%	60.40%	5.16%	3.47%	3.29%	30.50%
	100%	2.80%	1.77%	1.77%	8.41%	1.55%	1.33%	1.33%	4.73%
Llama3-3B	1%	28.79%	26.36%	25.96%	99.87%	25.54%	22.71%	22.74%	99.77%
	25%	31.73%	8.08%	6.99%	95.84%	13.44%	5.84%	5.8%	90.02%
	75%	12.27%	5.84%	5.22%	79.72%	8.54%	5.03%	5.15%	39.71%
	100%	3.62%	1.99%	1.99%	10.46%	2.37%	1.82%	1.85%	4.81%
Qwen2.5-0.5B	1%	10.12%	8.2%	8.23%	18.96%	7.85%	6.39%	6.00%	14.76%
	25%	5.19%	4.21%	4.45%	14.88%	4.35%	3.29%	3.49%	9.56%
	75%	3.56%	2.82%	3.15%	6.28%	2.39%	2.24%	2.15%	4.8%
	100%	0.98%	0.98%	0.98%	0.94%	0.23%	0.28%	0.33%	0.33%
Qwen2.5-1.5B	1%	14.56%	11.04%	10.22%	18.81%	9.47%	7.36%	7.48%	13.82%
	25%	13.29%	4.45%	5.34%	17.02%	6.83%	3.86%	4.00%	12.03%
	75%	7.03%	2.68%	2.84%	5.56%	4.21%	2.74%	2.79%	3.08%
	100%	0.7%	0.4%	0.4%	0.43%	0.65%	0.23%	0.23%	0.39%
Qwen2.5-3B	1%	12.63%	12.27%	11.44%	14.94%	7.85%	6.71%	6.48%	11.42
	25%	18.61%	8.59%	9.11%	17.89%	10.66%	4.75%	5.82%	11.59%
	75%	7.23%	4.00%	3.79%	5.30	3.65%	2.83%	2.8%	2.68%
	100%	0.39%	0.4%	0.4%	0.54%	0.31%	0.17%	0.2%	0.33%

Table 10: Performance drop (in percentage points) for GPT2 (small, large), Gemma-2B, Llama3 (3B, 8B), and Qwen2.5 (0.5B, 1.5B, 3B) models after applying word-level CAP for the Inverse Dictionary Modelling (IDM) task. 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, Sum and last token (lt).

would have diverged similarly. Instead, the persistence of the TW-CAP pattern across these controls demonstrates that the effect originates from compositional disruption rather than mechanical perturbation.

Model	Layer Position	Original				Fine-tuned			
		Max	Mean	Sum	lt	Max	Mean	Sum	lt
SP (Synonym Prediction)									
GPT2-small	1%	99.04%	99.04%	99.04%	98.56%	59.68%	49.40%	34.68%	74.34%
	25%	98.56%	98.56%	97.60%	83.01%	61.09%	30.85%	29.64%	73.37%
	75%	96.15%	94.23%	93.75%	14.69%	40.12%	9.68%	10.48%	57.85%
	100%	6.73%	7.21%	7.21%	2.78%	3.23%	2.42%	2.42%	11.38%
GPT2-large	1%	98.49%	98.49%	98.06%	98.28%	78.61%	78.33%	80.17%	75.7%
	25%	97.63%	97.63%	97.63%	97.41%	80.93%	81.78%	79.89%	69.27%
	75%	34.27%	27.59%	28.52%	36.42%	11.91%	10.02%	10.49%	23.04%
	100%	1.29%	1.51%	1.51%	1.72%	1.22%	39.12%	0.61%	0.44%
Gemma-2B	1%	99.99%	99.80%	83.47%	99.9 %	99.93%	99.15%	96.38%	90.79%
	25%	99.99%	97.46%	63.68%	97.75%	90.20%	90.24%	65.82%	38.5%
	75%	84.63%	60.66%	61.15%	68.11%	89.87%	75.68%	68.65%	24.85%
	100%	4.30%	8.69%	8.69%	5.57%	2.98%	4.57%	4.57%	4.01%
Llama3-8B	1%	99.99	99.90%	99.90%	100%	99.99%	99.88%	99.88%	98.85%
	25%	85.55%	83.50%	82.81%	99.8%	87.63%	85.75%	85.63%	96.43%
	75%	53.35%	50.55%	49.77%	74.98%	31.29%	30.29%	29.91%	54.19%
	100%	9.28%	9.96%	9.96%	16.52%	5.20%	5.82%	5.82%	3.19%
Llama3-3B	1%	100%	100%	100%	100%	100%	100%	100%	97.56%
	25%	85.81%	86.2%	85.16%	99.31%	88.47%	84.54%	85.48%	81.63%
	75%	40.18%	39.3%	38.91%	73.11%	14.77%	16.48%	15.64%	36.91%
	100%	5.77%	6.16%	6.16%	16.53 %	5.8%	6.12%	6.12%	5.68%
Qwen2.5-0.5B	1%	81.77	88.89%	79.17%	90.97 %	64.24%	58.36%	53.3%	%69.69
	25%	90.8%	91.15%	86.11%	92.01 %	54.51%	54.38%	37.22%	37.29%
	75%	63.72%	66.32%	39.06%	43.92%	48.87%	48.57%	24.29%	18.21%
	100%	8.51%	10.07%	8.51%	10.94%	3.67%	3.8%	3.8%	3%
Qwen2.5-1.5B	1%	89.35%	84.52%	84.23%	91.19%	64.55%	56.79%	56.03%	62.93%
	25%	90.58%	83.48%	83.19%	92.05%	60.45%	55.5%	54.79%	31.57%
	75%	22.06%	22.21%	18.8%	32.48%	10.88%	10.34%	10.02%	10.89%
	100%	6.82%	3.55%	3.55%	6.11%	8.19%	7.87%	7.87%	0.32%
Qwen2.5-3B	1%	81.39%	81.53%	73.58%	88.49 %	55.93%	49.35%	49.57%	58.84%
	25%	93.04%	89.91%	82.81%	91.34%	72.41%	42.78%	38.47%	33.39%
	75%	77.84%	69.6%	49.43%	37.22%	43.24%	22.13%	15.25%	11.03%
	100%	3.98%	3.13%	3.13%	4.83%	1.4%	1.29%	1.29%	1.51%

Table 11: Performance drop (in percentage points) for GPT2 (small, large), Gemma-2B, Llama3 (3B, 8B), and Qwen2.5 (0.5B, 1.5B, 3B) models after applying word-level CAP for the Synonym Prediction (SP) task. 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, Sum and last token (lt).

Model	Layer Position	Original				Fine-tuned			
		Max	Mean	Sum	lt	Max	Mean	Sum	lt
HP (Hypernym Prediction)									
GPT2-small	1%	99.75%	99.75%	99.75%	98.48%	91.19%	91.08%	88.20%	16%
	25%	99.47%	99.29%	98.94%	89.31%	81.35%	76.76%	72.63%	15.67&
	75%	95.40%	91.16%	91.32%	1.87%	48.75%	38.54%	38.40%	7.35&
	100%	8.12%	6.39%	6.39%	0.58%	1.35%	1.38%	1.28%	1.63%
GPT2-large	1%	99.27%	99.32%	99.20%	99.34%	91.49%	90.90%	89.80%	36.77%
	25%	98.81%	98.75%	98.10%	99.31%	87.30%	87.54%	84.16%	32.18&
	75%	45.17%	29.85%	35.66%	5.58%	7.61%	6.89%	6.22%	0.34%
	100%	2.14%	0.45%	0.90%	0.42%	0.69%	0.50%	0.56%	0.06%
Gemma-2B	1%	99.99%	98.97%	70.22%	98.51%	99.88%	95.39%	74.03%	72.53%
	25%	99.98%	90.58%	86.35%	80.83%	90.98%	73.78%	86.01%	29.49%
	75%	68.14%	80.06%	80.20%	27.65%	58.56%	72.57%	66.56%	16.26%
	100%	5.89%	10.99%	10.99%	5.77%	1.58%	2.12%	2.12%	2.12%
Llama3-8B	1%	99.99%	99.99%	99.14%	99.99%	99.99%	99.10%	99.14%	99.98
	25%	80.85%	76.97%	76.81%	91.42%	72.67%	71.86%	71.40%	96.38
	75%	24.43%	24.39%	23.11%	37.32%	19.65%	19.71%	18.77%	18.56
	100%	3.83%	4.49%	4.49%	5.48%	4.63%	4.04%	4.20%	2.37%
Llama3-3B	1%	100%	99.95%	99.95%	99.99%	99.93%	99.86%	99.82%	83.75%
	25%	88.04%	83.87%	84.34%	99.74%	65.53%	63.92%	64.17%	64.58%
	75%	26.06%	24.47%	23.4%	40.03%	11.06%	10.52%	10.79%	19.42%
	100%	4.34%	4.31%	4.31%	6.37%	3.85%	4.08%	3.86%	3.13%
Qwen2.5-0.5B	1%	93.76%	90.95%	85.27%	95.16%	86.33%	80.55%	77.91%	55.52%
	25%	97.12%	97.51%	89.18%	97.72%	74.83%	75.41%	75.77%	18.19%
	75%	76.74%	77.96%	55.39%	11.65%	50.69%	49.71%	48.81%	7.04%
	100%	6.15%	5.56%	5.56%	3.18%	2.48%	2.34%	2.34%	0.23%
Qwen2.5-1.5B	1%	97.14%	90.5%	88.96%	95.23%	88.52%	83.19%	77.21%	66.83%
	25%	98.12%	95.66%	94.04%	93.63%	72.29%	68.18%	68.33%	26.72%
	75%	18.27%	18.72%	17.94%	22.8%	8.94%	9.64%	9.51%	8.04%
	100%	7.13%	6.81%	6.81%	1.54%	3.95%	3.8%	3.8%	0.5%
Qwen2.5-3B	1%	83.26%	82.41%	68.8%	65.07 %	75.13%	72.56%	70.69%	44.64%
	25%	97.36%	96.32%	88.81%	94.36 %	92.69%	79.67%	79.63%	36.53%
	75%	86.56%	71.45%	45.47%	9.43%	40.87%	30.95%	33.04%	5.13%
	100%	2.07%	1.89%	1.89%	2.66 %	0.45%	0.35%	0.41%	0.31%

Table 12: Performance drop (in percentage points) for GPT2 (small, large), Gemma-2B, Llama3 (3B, 8B), and Qwen2.5 (0.5B, 1.5B, 3B) models after applying word-level CAP for the Hypernym Prediction (HP) task. 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, Sum and last token (lt).

Model	Layer Position	Original				Fine-tuned			
		Max	Mean	Sum	lt	Max	Mean	Sum	lt
IDM (Inverse Dictionary Modelling)									
GPT2-small	1%	93.00%	93.94%	96.56%	85.97%	77.912%	77.73%	80.28%	66.78%
	25%	90.20%	87.85%	91.41%	80.08%	65.73%	62.95%	72.31%	60.98%
	75%	87.81%	78.66%	84.90%	57.7%	55.74%	46.81%	55.73%	51.11%
	100%	48.10%	45.10%	38.04%	15.67%	11.11%	8.45%	8.11%	7.04%
GPT2-large	1%	87.06%	89.91%	88.44%	91.08%	81.14%	85.35%	79.46%	82.96%
	25%	73.54%	78.18%	82.48%	86.27%	69.39%	73.85%	71.90%	77.62%
	75%	49.02%	42.06%	40.38%	32.81%	20.59%	19.78%	21.45%	29.24%
	100%	28.14%	24.22%	24.78%	2.09%	6.46%	6.67%	8.44%	1.22%
Qwen2.5-0.5B	1%	93.97%	91.19%	87.15%	88.32%	90.94%	84.44%	78.85%	80.89 %
	25%	84.64%	76.78%	78.00%	84.37%	76.36%	66.24%	67.16%	73.03%
	75%	61.75%	57.95%	63.86%	45.28%	48.86%	41.8%	46.25%	34.05%
	100%	32.29%	26.8%	19.5%	3.66%	13.55%	10.17%	15.08%	4.41%
Qwen2.5-1.5B	1%	98.24%	95.8%	95.82%	92.38%	93.31%	87.33%	80.81%	85.71%
	25%	96.4%	84.72%	89.41%	81.66%	79.52%	63.00%	65.53%	78.14%
	75%	69.68%	64.6%	60.33%	31.63%	19.11%	14.72%	24.01%	17.23%
	100%	68.03%	60.04%	56.6%	4.2%	12.01%	7.46%	12.72%	3.24%
Qwen2.5-3B	1%	96.51%	94.37%	94.64%	89.2%	90.11%	86.02%	80.57%	82.63%
	25%	96.82%	89.89%	92.39%	88.15%	90.24%	76.55%	76.28%	79.73%
	75%	82.27%	74.71%	77.07%	39.9%	47.45%	36.06%	39.95%	24.85%
	100%	62.26%	62.21%	58.12%	2.38%	7.41%	5.52%	8.18%	1.8%

Table 13: Performance drop (in percentage points) for GPT2-small and GPT2-large and Qwen (0.5B-3B) models after applying phrasal-level CAP across the Dictionary Modelling (IDM) task. 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, Sum and last token (lt). Results for Gemma-2B and Llama3 models are omitted due to severe performance degradation under phrasal-level CAP.

Model	Layer Position	Original				Fine-tuned			
		Max	Mean	Sum	lt	Max	Mean	Sum	lt
SP (Synonym Prediction)									
GPT2-small	1%	99.99%	99.99%	99.99%	99.22&	64.90%	58.47%	53.22%	71.91%
	25%	92.97%	93.36%	93.36%	91.67&	61.27%	37.19%	74.69%	70.84%
	75%	92.58%	90.63%	92.19%	20.7&	43.35%	20.57%	52.22%	53.15%
	100%	58.46%	47.92%	51.43%	4.43&	13.27%	7.57%	12.45%	14.07%
GPT2-large	1%	96.67%	98.33%	96.67%	99.79%	92.55%	80.76%	79.58%	73.06%
	25%	96.67%	96.44%	97.90%	96.46%	79.44%	80.48%	82.86%	65.77%
	75%	78.83%	66.72%	66.32%	35.48%	18.63%	15.80%	21.00%	19.43%
	100%	67.10%	45.83%	56.68%	1.88%	9.69%	7.15%	8.33%	0.3%
Qwen2.5-0.5B	1%	99.32%	95.88%	92.87%	98.65%	81.67%	61.89%	57.95%	74.94%
	25%	98.65%	95.91%	96.45%	98.82%	60.19%	58.75%	58.43%	38.9%
	75%	93.21%	84.66%	77.4%	46.14%	56.29%	49.3%	44.94%	20.08%
	100%	68.78%	45.74%	43.92%	16.11%	13.56%	7.47%	16.79%	5.92%
Qwen2.5-1.5B	1%	98.1%	96.33%	94.43%	98.64%	72.33%	58.5%	59.55%	74.75%
	25%	97.55%	96.2%	95.38%	98.23%	63.79%	55.84%	68.93%	40.69%
	75%	75.72%	55.17%	48.41%	38.13%	19.33%	14.48%	26.87%	12.85%
	100%	70.39%	38.68%	36.29%	12.24%	18.73%	10.41%	20.97%	3.34%
Qwen2.5-3B	1%	96.47%	95.52%	90.31%	98.64%	74.05%	67.1%	56.57%	69.92%
	25%	99.32%	98.1%	94.29%	97.28%	94.89%	56.93%	57.38%	43.78
	75%	94.02%	89.46%	83.4%	42.09%	86.43%	64.01%	43.39%	18.71
	100%	47.00%	35.56%	31.32%	9.96%	20.07%	15.19%	21.15%	4.09

Table 14: Performance drop (in percentage points) for GPT2-small and GPT2-large and Qwen (0.5B-3B) models after applying phrasal-level CAP for the Synonym Prediction (SP) task. 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, Sum and last token (lt). Results for Gemma-2B and Llama3 models are omitted due to severe performance degradation under phrasal-level CAP.

Model	Layer Position	Original				Fine-tuned			
		Max	**Mean**	**Sum**	lt	**Max**	**Mean**	**Sum**	lt
HP (Hypernym Prediction)									
GPT2-small	1%	99.40%	99.26%	47.24%	98.41%	89.31%	89.86%	88.76%	18.04%
	25%	99.31%	98.12%	46.38%	80.44%	77.72%	73.12%	76.08%	16.41%
	75%	95.63%	91.78%	45.57%	2.98%	47.73%	36.59%	48.32%	9.47%
	100%	65.62%	45.84%	34.80%	0.85%	4.80%	3.64%	4.00%	2.4%
GPT2-large	1%	99.77%	99.71%	99.76%	99.77%	91.63%	92.56%	88.92%	52.22%
	25%	99.82%	98.72%	98.82%	99.49%	85.31%	85.35%	84.58%	25.8%
	75%	66.58%	49.79%	63.56%	6.72%	9.87%	8.79%	9.73%	0.54%
	100%	35.57%	24.79%	26.69%	2.28%	6.99%	5.05%	4.82%	0.14%
Qwen2.5-0.5B	1%	99.06%	97.77%	92.97%	98.8%	94.46%	81.39%	79.64%	55.33%
	25%	99.85%	98.54%	96.95%	99.82%	75.14%	76.07%	86.94%	24.11%
	75%	94.87%	87.81%	88.37%	93.1%	56.27%	53.09%	63.33%	19.81%
	100%	68.71%	27.91%	27.92%	19.21%	10.6%	7.68%	15.16%	12.08%
Qwen2.5-1.5B	1%	99.81%	97.07%	92.75%	99.55%	90.34%	84.61%	78.76%	78.63%
	25%	99.64%	97.97%	96.98%	98.48%	72.81%	68.48%	77.13%	40.29%
	75%	84.28%	47.63%	43.15%	54.17%	17.12%	14.76%	28.18%	14.09%
	100%	82.22%	26.00%	27.7%	6.19%	13.49%	9.08%	17.98%	4.28%
Qwen2.5-3B	1%	93.95%	91.81%	82.05%	93.77%	77.6%	73.86%	71.41%	48.13%
	25%	99.24%	98.54%	95.97%	93.59&	93.6%	80.32%	80.77%	42.29%
	75%	94.48%	88.91%	78.88%	55.35%	54.32%	38.19%	57.87%	11.07%
	100%	55.28%	27.4%	25.1%	7.96%	15.1%	8.77%	13.77%	3.78%

Table 15: Performance drop (in percentage points) for GPT2-small and GPT2-large and Qwen (0.5B-3B) models after applying phrasal-level CAP across on the Hypernym Prediction (HP) task. 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, Sum and last token (lt). Results for Gemma-2B and Llama3 models are omitted due to severe performance degradation under phrasal-level CAP.