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%
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.

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 phrase-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

Model	Task	Mean ± Std
GPT2 (S)	IDM	3 ± 5
	SP	27 ± 9
	HP	27 ± 10
GPT2 (M)	IDM	3 ± 5
	SP	28 ± 10
	HP	26 ± 11
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

positional structure.

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 9, 10, and 11 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

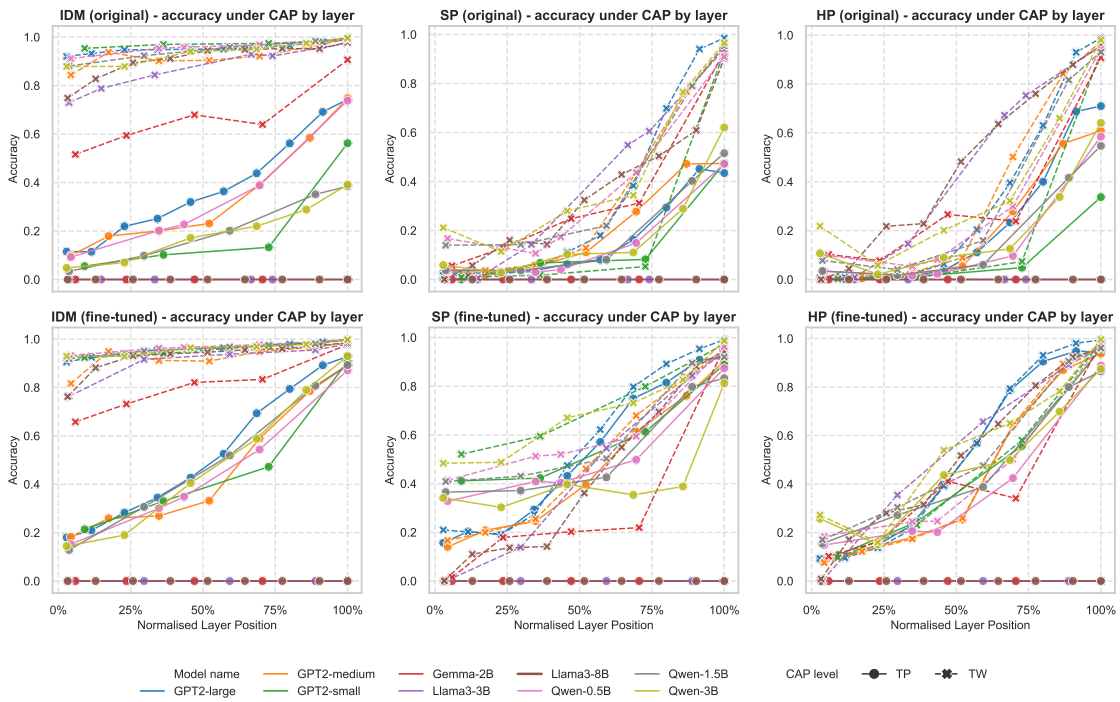

Figure 3: Average grouped accuracy of CAP across different aggregation functions for normalised layer positions (0%-100%) is shown for word-level CAP (TW) and phrasal-level CAP (TP). Sub-figures (a)-(c) illustrate the CAP effect on the original (Org) models, while sub-figures (d)-(f) show its impact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

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.