Interpreting token compositionality in LLMs: A robustness analysis

Nura Aljaafari[1†], **Danilo S. Carvalho**[1,3], **André Freitas**[1,2,3]
[1] Department of Computer Science, University of Manchester, United Kingdom
[2] Idiap Research Institute, Switzerland
[3] National Biomarker Centre, CRUK-MI, Univ. of Manchester, United Kingdom
{firstname.lastname}@[postgrad.]†manchester.ac.uk

Abstract

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

1 Introduction

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, poor interpretability, and limited 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), embedding spaces (Haslett, 2024), or layer-wise activations (Yu and Ettinger, 2020; Modarressi et al., 2023) with expected semantic representations. These approaches are grounded in two intuitive 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.

Several studies have revealed that LLMs are often brittle under perturbation (Wang et al., 2023; Fodor et al., 2024; Hu et al., 2024), and that phrase-level representations may fail to align with expected semantics (Carvalho et al., 2025). Despite this, the mechanisms behind such 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 constituent units (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.

Our empirical findings challenge common assumptions of hierarchical semantic buildup. Rather than gradually constructing compositional meaning across layers, LLMs often retain token-level focus well into the middle layers. Applying CAP, even at semantically coherent groupings, results

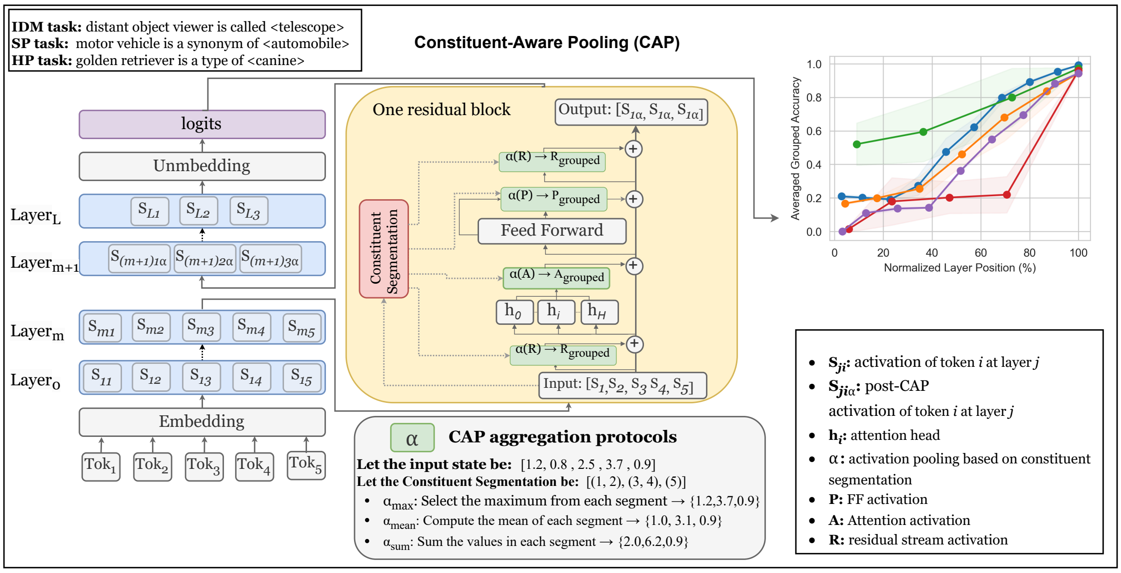
IDM task: distant object viewer is called <telescope>
SP task: motor vehicle is a synonym of <automobile>
HP task: golden retriever is a type of <canine>
Constituent-Aware Pooling (CAP)
logits
Unmbedding
Layer_L
S_{L1} S_{L2} S_{L3}
Layer_{m+1}
$S_{(m+1)1\alpha}$ $S_{(m+1)2\alpha}$ $S_{(m+1)3\alpha}$
Layer_m
S_{m1} S_{m2} S_{m3} S_{m4} S_{m5}
Layer_o
S_{11} S_{12} S_{13} S_{14} S_{15}
Embedding
Tok_1 Tok_2 Tok_3 Tok_4 Tok_5
One residual block
Output: $[S_{1\alpha}, S_{1\alpha}, S_{1\alpha}]$
$\alpha(R) \rightarrow R_{\text{grouped}}$
$\alpha(P) \rightarrow P_{\text{grouped}}$
Feed Forward
$\alpha(A) \rightarrow A_{\text{grouped}}$
h_0 h_i h_H
$\alpha(R) \rightarrow R_{\text{grouped}}$
Input: $[S_1, S_2, S_3\ S_4, S_5]$
Constituent Segmentation
Averaged Grouped Accuracy
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
Normalized Layer Position (%)
α
CAP aggregation protocols
Let the input state be: [1.2, 0.8 , 2.5 , 3.7 , 0.9]
Let the Constituent Segmentation be: [(1, 2), (3, 4), (5)]
α_{max}: Select the maximum from each segment → {1.2,3.7,0.9}
α_{mean}: Compute the mean of each segment → {1.0, 3.1, 0.9}
α_{sum}: Sum the values in each segment → {2.0,6.2,0.9}
S_{ji}: activation of token i at layer j
$S_{ji\alpha}$: post-CAP activation of token i at layer j
h_i: attention head
α: activation pooling based on constituent segmentation
P: FF activation
A: Attention activation
R: residual stream activation

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). 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.

in substantial accuracy degradation, especially in earlier layers. Surprisingly, larger models are more sensitive to such perturbations than smaller ones, suggesting increased representational fragility with scale.

We contextualise these results using an information-theoretic lens, proposing that Transformer models delay aggregation to maximise token-level information throughput. This leads to distributed, rather than localised, composition across layers, resulting in longer dependency paths and reduced mutual redundancy at each layer.

In summary, our contributions are:

- **A systematic analysis** of how current LLMs handle constituent-level composition, evaluated via CAP across layers, models, and tasks.
- **A theoretical explanation** grounded in information theory, suggesting that LLMs optimise for prediction by postponing semantic integration, thus fragmenting compositional meaning across depth.

We conclude that compositional semantics are not reliably localisable within any fixed layer of standard Transformer models. This holds across model scales, supervision types, and inference tasks, and instead appears tied to architectural depth. Our results suggest that recovering explicit compositional structure may require specialised training objectives or architectural constraints. Supporting code and datasets are available at a public repository[1].

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 composi-

[1] < anonymised url>

tional 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 in which they can be addressed is by systematically assessing the impact of said perturbations on model inference performance, at each model layer. We elaborate 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 (Figure 1):

1. Inverse definition modelling (IDM): predicting a term given its definition.

2. Synonym prediction (SP): producing a synonym for a given word.

3. Hypernym prediction (HP): generating a more general term for a given word.

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

Constituent-Aware Pooling (CAP). To introduce compositional perturbations, we propose CAP, a method for pooling (i.e., grouping) LLM activations corresponding to individual tokens into cohesive linguistic units. CAP operates at two levels: (i) word-level: grouping tokens that form a single word, and (ii) phrase-level: grouping tokens that form a single phrase. At the word-level, CAP reverse-maps each model's tokeniser to reconstruct complete words and identify their activation ranges. At the phrase-level, CAP uses a syntactic parser, such as Benepar (Kitaev et al., 2019; Kitaev and Klein, 2018), to align tokens with their corresponding phrasal constituents and define their activation ranges. Further details on the parser evaluation methodology are provided in Appendix D.

CAP Pooling Protocols. CAP is applied progressively across layers using three protocols α: *Max:* selects the maximum activation within a segment, identifying dominant features and their propagation through layers; *Mean:* computes the average activation, providing a balanced representation of all token contributions and their collective impact on model decisions; and *Sum:* sums the activations, capturing cumulative information flow and aggregates effects of token interactions. These protocols offer complementary insights into how models process and integrate information: Max reveals feature prominence patterns, Mean shows distributed representation effects, and Sum reflects accumulated semantic content across segments.

Transformer conceptualisation and the formalisation of CAP. This work builds on the mathematical framework of transformers introduced by (Elhage et al., 2021), where computation is formalised into sequential residual blocks. Each layer reads inputs from the residual stream, processes them through its components (attention heads and feed-forward neural networks (FF)), and writes the outputs back into the residual 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 the transformer model have L layers, input sequence of length K, batch size B, and inner activations X, with with tensor shapes varying by model component as follows:

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

Let $\mathcal{S} = \{(s_1, e_1), \ldots (s_n, e_n)\}$ be the set of syntactic unit ranges (e.g., tokens, words or phrases), where s_i and e_i denote the start and end indices of the i-th range. CAP pools/groups activations within these ranges, reducing the sequence dimension K to a grouped dimension G, where

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

For each syntactic unit, CAP applies the grouping function α over the range $[s_i, e_i]$ in one of three