First Look at Stochastic Parameter Decomposition on SimpleStories

Author: Chandan Sreedhara

Epistemic status:
This is exploratory analysis of Stochastic Parameter Decomposition (SPD) [1] subcomponents on the decomposed ~1.25M parameter model trained on SimpleStories dataset. I can show activation patterns but NOT causal mechanisms (no ablation studies).

Confidence: High on descriptive statistics, Low on mechanistic interpretation.

What I did

- Analyzed the SPD decomposed SimpleStories [2] model across ~25k stories (link to run: at the time I started this experiment this was the latest but by the time I am writing this report, this is already outdated and results could possibly improve by using later runs)
- Wrote a script to index the activations of subcomponents for tokens for easier search and analysis
- Multiple plottings and tables to understand the subcomponent activation patterns
- Total tokens: ~8M tokens
- 3894 unique tokens
- Vocabulary size of the model: 4096

With 50,000 stories sampled from the full SimpleStories dataset (2.3% coverage) and 8M tokens analyzed covering 95% of the vocabulary, we have **>99% confidence in the aggregate layer-wise activation patterns** and **93-97% confidence in the component specialization distributions** (within ±5 percentage points), with highest confidence for medium to high-frequency tokens.

Key limitation: I'm analyzing individual rank-1 subcomponents, not the clustered multi-rank components that SPD claims form "mechanisms. This is only statistical analysis and there is no ablation studies done here.

Core question

How often do sub-components activate? How specialized are the sub-components? Do they activate on specific tokens (monosemantic) or broadly (polysemantic)?

Methodology

To analyze subcomponent activation patterns, I applied the trained SPD model to approximately 100,000 randomly sampled stories from the SimpleStories dataset. Following the SPD framework, each weight matrix in the neural network is decomposed into C rank-one subcomponents of the form $\mathbf{U}_c^l \mathbf{V}_c^{l\top}$, where the number of subcomponents C can exceed the rank of the original matrix to capture computations in superposition.

For each story, I tokenized the text and performed a forward pass through the SPD model. For each subcomponent at each layer, I computed its "inner activation"

$$h_c^l(x) = \sum_j V_{c,j}^l a_j^l(x)$$

where $\mathbf{V}_c^l$ represents the subcomponent's input direction (analogous to reading weights) and $\mathbf{a}^l(x)$ is the activation vector at that layer.

This inner activation quantifies how much information from the current token flows through that particular subcomponent. These inner activations were then passed through small trained MLPs (causal importance functions Γ_c^l) that predict whether each subcomponent is causally necessary for computing the model's output. A subcomponent was considered "active" at a given token if its predicted causal importance exceeded a threshold of 0.01. The activation data was saved incrementally in batches of 500 stories to manage memory constraints.

To enable efficient analysis, I implemented a two-stage indexing pipeline. I randomly sampled 50 batches (approximately 50,000 stories) and built a SQLite database storing activation records as tuples with indexed lookups on layer-component pairs. A token mapping dictionary linked each to its token string.

Main findings

These are the findings I found most interesting to pursue for now and added a stop here to write the report.

Finding 1: Activity of layers

For each weight matrix in the model, I measure how many subcomponents are active vs. dead. This reveals whether the decomposition is over-parameterized or efficiently captures computation. For each weight matrix in the model (q_proj, k_proj, v_proj, o_proj in attention, and gate_proj, up_proj, down_proj in MLPs), how many of the 4000 subcomponents per layer activated at least once across our dataset is counted. "Dead" components never activated; "active" components activated on at least one token. This tells us how much of the model's decomposed capacity is actually being used in each layer. The active components are further divided in rare (1-100) and frequent (>100)

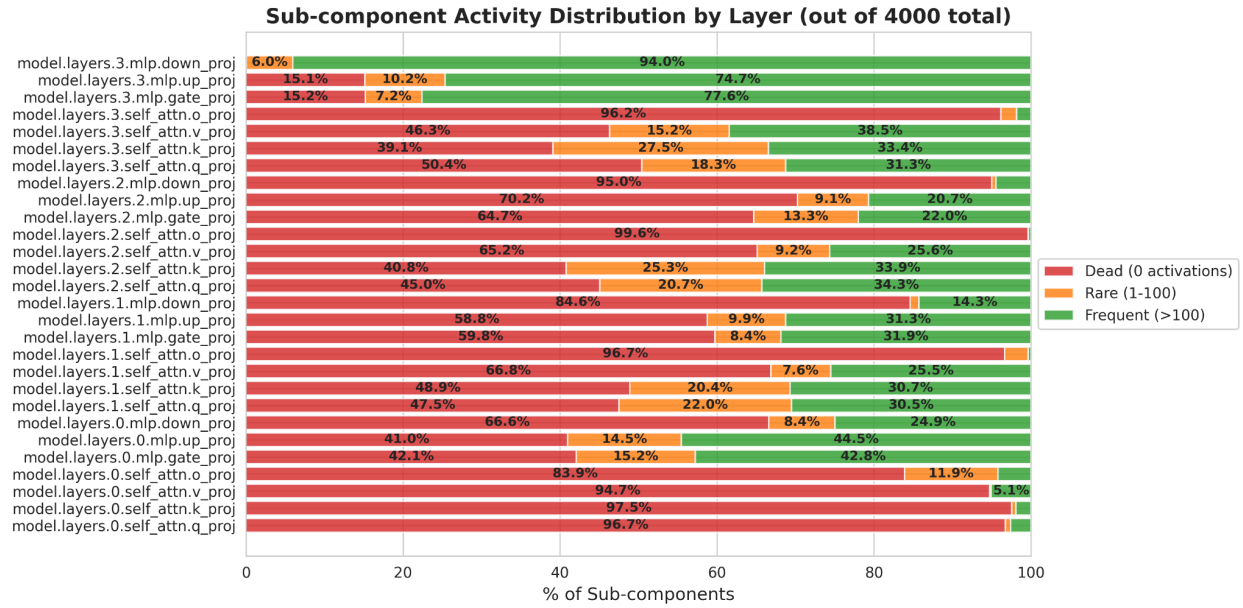

Figure 1: sub-component activity distribution in each layer in percentage. If a subcomponent has zero activation across tokens then it is considered a dead sub-component. Anything between 1-100, it is rare and above 100 is frequent. We are trying to see how active a layer is from sub-component's perspective

Layer	Total sub-components	Active (≥1)	Dead (0)
model.layers.0.self_attn.q_proj	4000	131 (3.3%)	3869 (96.7%)
model.layers.0.self_attn.k_proj	4000	98 (2.5%)	3902 (97.5%)
model.layers.0.self_attn.v_proj	4000	211 (5.3%)	3789 (94.7%)
model.layers.0.self_attn.o_proj	4000	644 (16.1%)	3356 (83.9%)
model.layers.0.mlp.gate_proj	4000	2316 (57.9%)	1684 (42.1%)
model.layers.0.mlp.up_proj	4000	2362 (59.0%)	1638 (41.0%)
model.layers.0.mlp.down_proj	4000	1336 (33.4%)	2664 (66.6%)
model.layers.1.self_attn.q_proj	4000	2099 (52.5%)	1901 (47.5%)
model.layers.1.self_attn.k_proj	4000	2044 (51.1%)	1956 (48.9%)
model.layers.1.self_attn.v_proj	4000	1326 (33.1%)	2674 (66.8%)
model.layers.1.self_attn.o_proj	4000	134 (3.4%)	3866 (96.7%)
model.layers.1.mlp.gate_proj	4000	1610 (40.2%)	2390 (59.8%)
model.layers.1.mlp.up_proj	4000	1649 (41.2%)	2351 (58.8%)
model.layers.1.mlp.down_proj	4000	615 (15.4%)	3385 (84.6%)
model.layers.2.self_attn.q_proj	4000	2198 (55.0%)	1802 (45.0%)
model.layers.2.self_attn.k_proj	4000	2369 (59.2%)	1631 (40.8%)
model.layers.2.self_attn.v_proj	4000	1394 (34.9%)	2606 (65.2%)
model.layers.2.self_attn.o_proj	4000	15 (0.4%)	3985 (99.6%)
model.layers.2.mlp.gate_proj	4000	1413 (35.3%)	2587 (64.7%)
model.layers.2.mlp.up_proj	4000	1190 (29.8%)	2810 (70.2%)
model.layers.2.mlp.down_proj	4000	200 (5.0%)	3800 (95.0%)
model.layers.3.self_attn.q_proj	4000	1984 (49.6%)	2016 (50.4%)
model.layers.3.self_attn.k_proj	4000	2436 (60.9%)	1564 (39.1%)
model.layers.3.self_attn.v_proj	4000	2147 (53.7%)	1853 (46.3%)
model.layers.3.self_attn.o_proj	4000	154 (3.9%)	3846 (96.2%)
model.layers.3.mlp.gate_proj	4000	3393 (84.8%)	607 (15.2%)
model.layers.3.mlp.up_proj	4000	3395 (84.9%)	605 (15.1%)
model.layers.3.mlp.down_proj	4000	3999 (100.0%)	1 (0.0%)

Table 1: Table showing the raw numbers of sub-component activations. This gives a picture of how active a layer is without going to details of rare and frequent activations.

Observed distribution of active vs dead components:

Layer 0 attention shows 96-97% dead subcomponents across q/k/v projections and 84% dead in o_proj, while Layer 0 MLP is more active with 42-66% dead components. Layer 1 attention "wakes up" with q/k/v projections showing 48-67% dead components, but o_proj remains sparse at 96.7% dead. Layer 2 follows a similar pattern - q/k/v have 40-65% dead components while o_proj reaches 99.6% dead. Layer 3 MLP shows a dramatic shift: down_proj is 100% active (0% dead) while gate/up_proj maintain ~15% dead components. Across all layers, attention output projections (o_proj) are consistently sparse (96-99% dead) while query/key/value projections become progressively more active in deeper layers.

In this decomposition run, the loss on layer 3 down_proj was quite high so I have decided to not explore Layer 3 down_proj being 100% active further since this could be the result of faulty training (since this was one of the first training runs).

Key takeaway: Early attention layers are extremely sparse (96%+ dead components), while Layer 3 MLP is fully active. This suggests a progression from sparse lookup-table computation to dense integration.

Possible mechanistic hypothesis (Speculative, requires ablation study to gain more confidence):

The sparse early layers (Layer 0 attention at 96%+ dead) suggest lookup table behavior where only a few specific subcomponents activate for specific tokens, consistent with early feature detection before much context has accumulated in the residual stream. As layers progress, activation increases because the residual stream accumulates more context; by Layer 3, the model needs dense computation (down_proj 100% active) to map from a context-rich residual stream to vocabulary logits.

The o_proj anomaly where attention output projections remain consistently sparse across all layers, suggests the model learned to write very selectively to the residual stream, only updating it when specific attention patterns are detected. This raises questions:

1. Does the sparse o_proj mean attention is doing pattern matching but only occasionally finding matches worth writing about?
2. Why does Layer 1 attention "wake up" (q/k/v more active) but then Layer 2 attention output becomes even more sparse? The extreme sparsity in o_proj combined with

increasing activity in q/k/v suggests attention may be computing many patterns but discarding most of them as irrelevant.

Finding 2: Variation in specialization in active sub-components

In this section, I wanted to explore how specialized the active sub-components are. So for each active subcomponent, I measure how many unique tokens activate it. I categorize components as: "Truly Monosemantic" (1 token), "Highly Specialized" (2-5 tokens), "Specialized" (6-20 tokens), "Moderate" (21-100 tokens), or "Polysemantic" (>100 tokens). The bar charts show the percentage of active components in each category per layer. The tables show specific examples of components, their activation counts, unique token counts, and top-5 most frequently activating tokens.

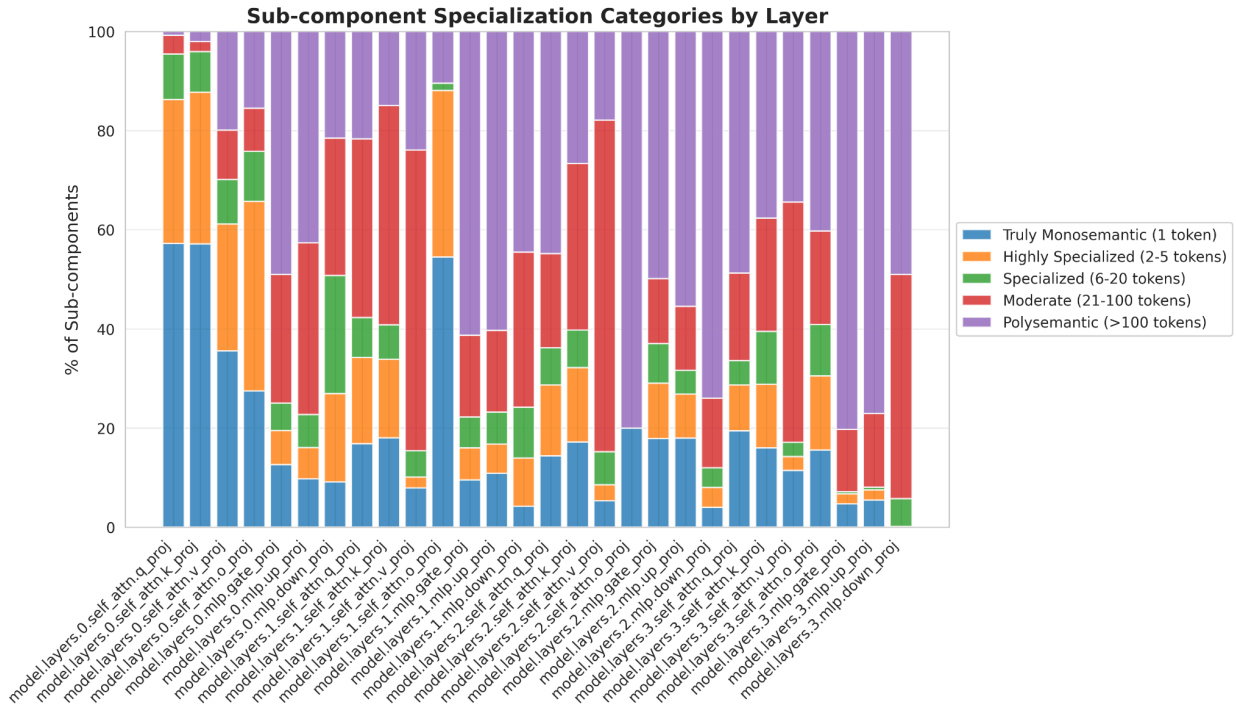

Figure 2: sub-component behaviour in terms of what it activates in each layer. Monosemantic is if it activates for only one token across stories. Polysemantic is if it activates for more than 100 tokens. Note that this is a bit different than typical mech-interp definition of "more than one" for polysemantic, the reason being there are lots of sub-component activity between 1-100.

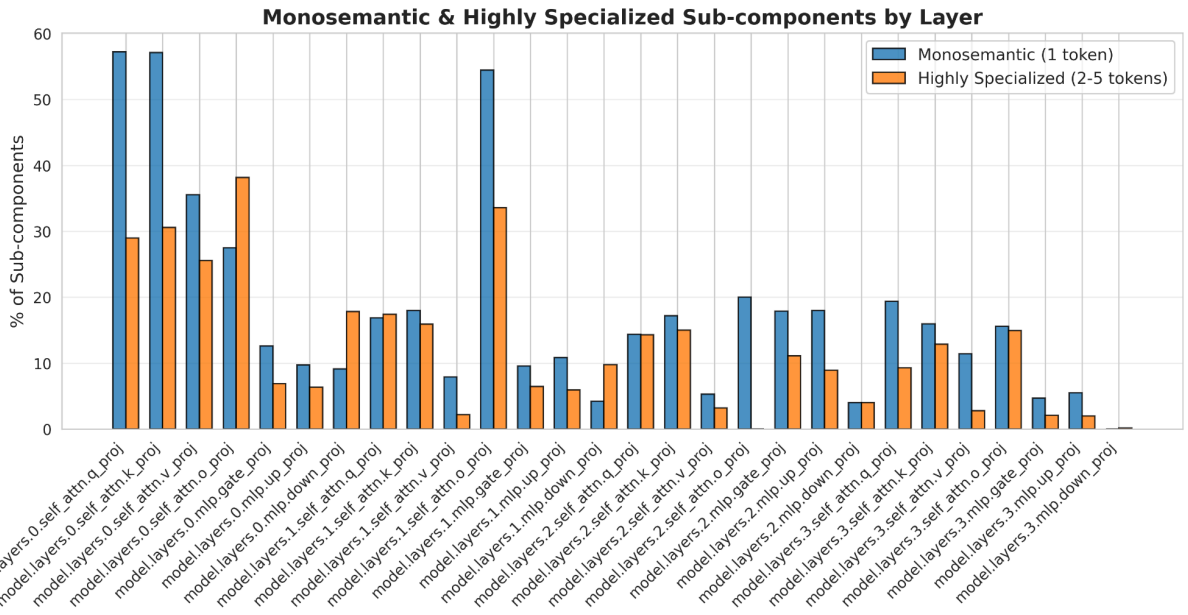

Figure 2: Zoomed in version of the activity of sub-components in monosemantic and highly specialized regime for each layer.

The relationship between sparsity and specialization shows a bimodal pattern: layers with many dead components (high sparsity) concentrate their active components at the extremes - both highly monosemantic AND highly polysemantic, with fewer in the moderate range. Layer 0 q_proj and k_proj (96-97% dead) demonstrate this clearly in Figure 2: they have high percentages of monosemantic components (the blue bars) alongside polysemantic components, but relatively little in between.

This suggests two distinct computational roles in sparse layers that **sub-components serve as lookup tables** like component 2600 activating only on 44 name tokens functions as a specialized detector and **polysemantic components as structural utilities** like Component 3192 activating across 3672 tokens (94% of vocabulary) providing broad structural processing.

Polysemantic components serve as structural utilities: Component 3192 activates 6.2M times across 3672 unique tokens (94% of vocabulary).This component doesn't specialize on token identity but rather provides broad structural processing. Component 1996 shows intermediate specialization, activating on 8 tokens that all follow the pattern "##[2-3 letters]" (##at, ##ie, ##ip, ##be, ##ned), suggesting it detects word-internal morphological structure rather than specific tokens.

Sub-Component	Activations	Unique	Top 5 Tokens
3192	6236485	3672	.(532627), ,(420882), the(383049), a(200066), "(180754)
1716	192990	65	##s(26227), asked(10944), by(9487), samuel(8888), let(6155)

1941	92343	62	little(5352), ##n(5312), knowing(5213), ##er(4540), finally(3924)
838	600732	48	.(532627), !(53579), swam(1785), don(1306), grandmother(1079)
2600	134019	44	mia(13802), leo(13111), alex(10348), kim(9566), samuel(8888)
3663	35253	37	finally(3924), days(3424), sometimes(2273), remember(2246), among(1927)
2796	1646	12	clear(853), z(535), envel(142), ##iting(83), ##ughter(9)
18	11936	11	next(3291), :(1913), realizing(1459), inspired(1164), however(771)
3385	42	11	tal(10), whist(6), invis(5), por(5), merm(5)
689	4208	9	dear(1005), however(771), po(643), during(474), pl(394)
287	24	8	##illy(13), ##more(4), pizz(2), ##round(1), foc(1)
1996	4124	8	##at(914), ##ie(820), ##ip(540), ##be(472), ##ned(414)
1434	53	7	##eter(16), gigg(15), tun(7), ##tory(4), ##anged(4)
3150	3216	6	realizing(1459), ex(533), seeing(431), honor(395), hour(223)
1136	186690	6	"(180754), curious(3736), determined(1031), though(671), please(463)
974	2778	6	po(643), ve(602), bl(511), gu(440), perhaps(384)
276	6666	6	-(3560), ##b(1243), e(621), ex(533), bl(511)
3138	6556	6	anne(2573), luis(1523), bo(1380), dear(1005), ##tering(59)
2927	7197	5	-(3560), ##st(1534), dear(1005), however(771), tonight(327)
2948	3008	5	get(1299), ma(742), po(643), awa(265), sou(59)

Table 2: Table with details of polysemantic sub-components for layers.0.self_attn.q_proj. This layer has very few polysemantic sub-components and consequently top ones activate for majority of tokens. sub-component 3192 activates for 3672 unique tokens out of 4096 vocabulary size. Sub-component 2600 specializes in names. Sub-component 1996 for ones that starts with ## and ends with 2-3 letters.

Component	**Token**	**Activations**
2309	"	180754
1587	painting	1236
3227	these	1180
886	red	1177
2890	inspired	1164
3475	##l	951
3820	clear	853
654	un	828

2712	n	738
1669	few	690
3265	may	687
3083	po	643
3456	e	621
2366	e	621
3822	z	535
319	please	463
1517	please	463

Table 3: Table showing the top monosemantic sub-components in layers.0.self_attn.q_proj. Each sub-component fires exactly for this token. Activations column shows how many times this sub-component has activated for this. Note that there can be multiple monosemantic sub-components that fire for the same token, "please" being a one example which activates both 319 and 1517.

The o_proj anomalous polysemanticity:

Layer 1 and 2 o_proj show an extreme pattern as opposite to Q and V of early layers as shown in Table 4 and 5: despite 96.7% and 99.6% dead components respectively, the active components are highly polysemantic. For example, Layer 2 o_proj component 3432 activates 6.6M times across 3894 unique tokens (essentially the entire vocabulary), with top activations on high-frequency structural tokens (periods, commas, articles). This suggests attention output projections implement 'broadcast' operations - when they write to the residual stream, they write relatively uniform information across many token types, rather than token-specific updates. These components activate on essentially the entire vocabulary (3893-3894 unique tokens out of ~3894 total), with millions of activations dominated by high-frequency structural tokens (punctuation, articles, common words).

Sub-Component	**Activations**	**Unique**	**Top 5 Tokens**
3272	6.2M	3893	.(442K), the(339K), ,(312K), a(198K), "(158K)
2575	4.4M	3856	.(378K), the(258K), ,(222K), a(185K), "(106K)
3151	4.8M	3853	.(533K), ,(315K), the(229K), "(181K), and(119K)
3983	4.0M	3816	,(350K), the(180K), .(141K), a(135K), and(132K)
234	2.5M	3728	the(382K), a(138K), .(137K), ,(133K), of(57K)

Table 4: Table with details of polysemantic sub-components for layers.1.self_attn.o_proj. This layer has very few monosemantic sub-components and the polysemantic sub-components activate for almost all the unique tokens.

Sub-Component	Activations	Unique	Top 5 Tokens
3432	6.6M	3894	.(524K), ,(390K), the(364K), a(190K), and(148K)
2250	6.2M	3887	.(493K), ,(362K), the(326K), "(172K), a(169K)
3771	5.0M	3885	.(440K), ,(345K), the(241K), a(196K), "(141K)
97	5.7M	3878	.(478K), ,(323K), the(307K), "(174K), a(154K)
3603	3.9M	3876	the(383K), .(324K), a(200K), ,(159K), her(74K)

Table 5: Table with details of polysemantic sub-components for layers.2.self_attn.o_proj. This layer behaves similarly to the layers.1.self_attn.o_proj.

As layers progress deeper, we observe increasing polysemanticity (visible in the purple bars growing in Figure 2 and the rising median diversity in Figure 4). This shift aligns with the accumulation of context in the residual stream: early layers can rely on token-specific lookups, but later layers processing context-rich representations require more polysemantic components that integrate information across many token types. Layer 3 MLPs (15% dead) show more even distribution across specialization categories - they can afford intermediate levels of specialization because they're performing context-dependent computations rather than simple lookups.

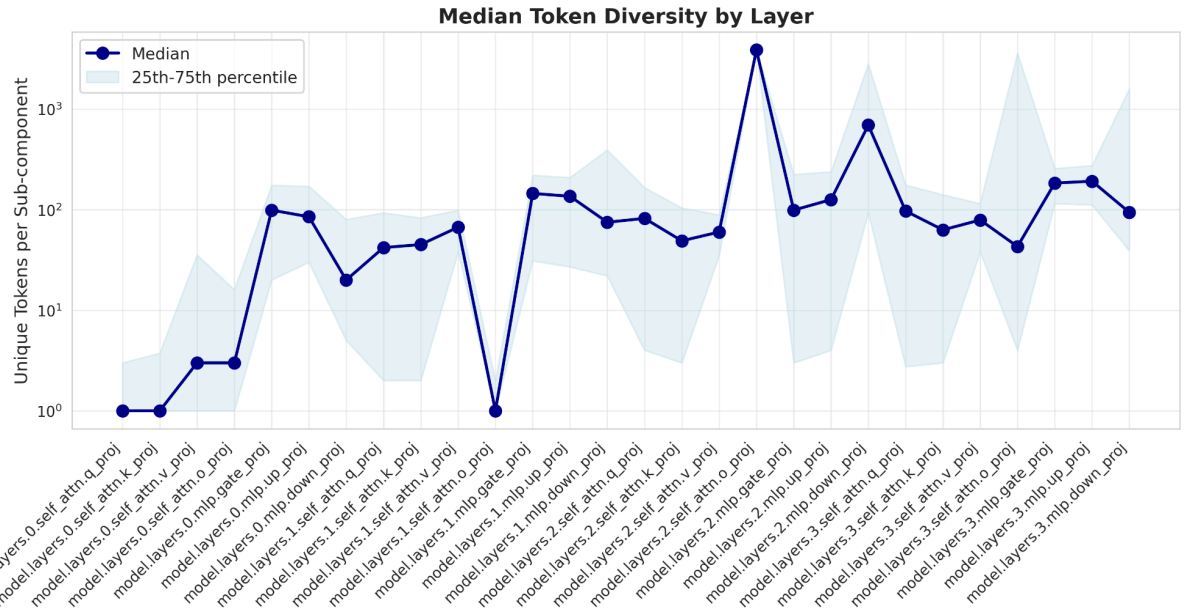

Figure 4: Median unique token diversity across layers. This is an alternative view of seeing how polysemantic sub-components are in a layer.

Finding 3: Layer dependant pattern for tokens

I wanted to see if there is specialization of sub-components from the perspective of tokens. That is for each token, would we see change in activation rates of the sub-component. So I tagged each token in our vocabulary with its part-of-speech category (noun, verb, adjective, punctuation, etc.). A subcomponent is classified as "high concentration" if its top 10 most frequent tokens account for >90% of all its activations, meaning it's activated by a very narrow set of tokens.

The line plots below show that given a token of a specific category (e.g., noun, punctuation), what percentage of the high-concentration subcomponents activate for that token? What percentage of all sub-components activate for that token?

This is a token's-eye view: it tells us which token categories tend to activate the subcomponents in each layer.

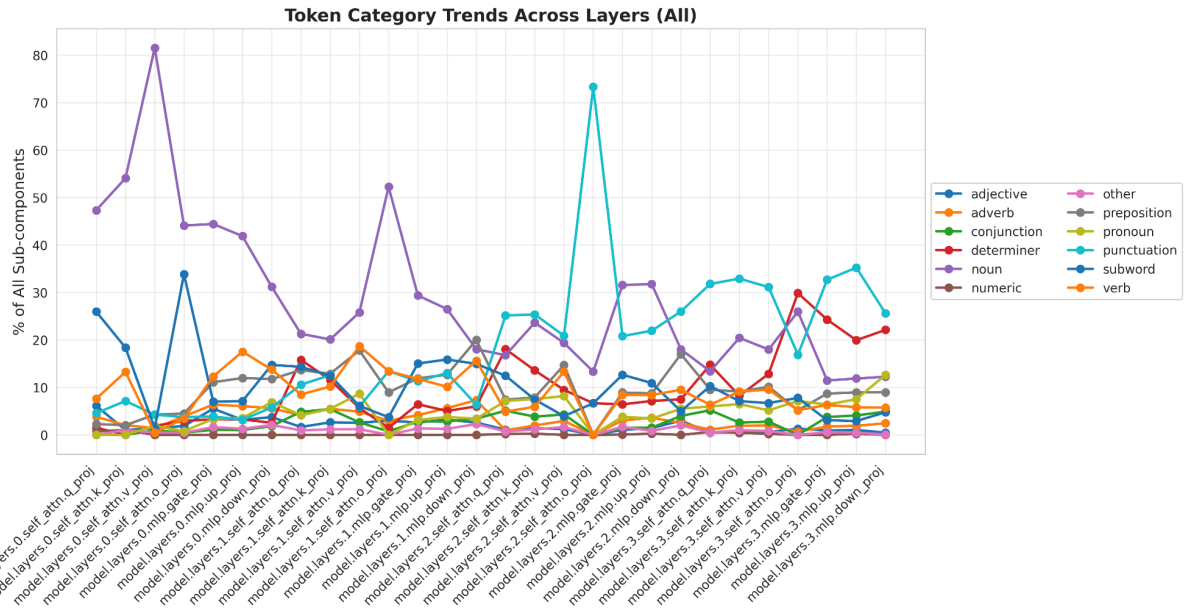

Figure 5: Token's perspective on all subcomponents activations. In this figure, given a token, we see the % sub-components that are activated for that token. The tokens are classified into different semantic categories.

Layer 0 - Noun dominance: When noun tokens appear, they activate ~82% of high-concentration subcomponents in self_attn.v_proj and 45-55% in k_proj. Adjectives and adverbs show moderate concentration in MLP components (~34% and ~17% in gate_proj respectively). This indicates highly specialized, token-specific processing in early layers.

Layer 1 - Transition phase: Noun concentration drops to 20-30% while category distribution becomes more balanced across token types, suggesting a shift from token-specific to more context-dependent processing.

Layer 2 - Punctuation spike: Punctuation tokens activate 73% of all subcomponents in attn.o_proj, the single highest category concentration observed in any layer. This extreme specialization is notable: punctuation tokens (periods, commas, quotes) trigger dedicated narrow-purpose components. It continues to stay high throughout after this spike.

Layer 3 - Determiner emergence: Determiners (the, a, an) activate ~30% of all components in attn.o_proj. Punctuation remains elevated (25-35%) but more distributed. Overall category distribution is more balanced than earlier layers.

Interpreting the pattern:

Why do nouns dominate Layer 0, punctuation Layer 2, and determiners Layer 3? Two possibilities:

Mechanistic: The model implements a linguistic processing pipeline—identify entities, parse structure, resolve references—with each layer handling successive stages.

Statistical: The model allocates specialized capacity to high-frequency categories (nouns and punctuation are extremely common in SimpleStories) at the layers where those tokens require the most processing.

These make different predictions: mechanistic interpretation predicts component behavior tracks linguistic properties; statistical interpretation predicts it tracks token frequency. Without ablations or experiments on different datasets, we cannot distinguish these explanations.

Next steps:

This exploratory analysis reveals intriguing patterns in SPD subcomponent behavior: bimodal specialization, anomalous o_proj sparsity, and layer-dependent token category preferences but observational analysis has its limits. To move from correlation to causation, several directions are worth pursuing:

1. Ablation studies testing causal effects
2. Include more stories from SimpleStories to strengthen the hypothesis
3. Threshold sensitivity analysis (does bimodality persist at different activation thresholds?)
4. Comparison to other decomposition methods to evaluate whether SPD achieves something distinctive.

Bottom line: This validates that SPD produces decompositions with clear, non-random structure. But we haven't proven the decomposition is *uniquely correct*, *causally meaningful*, or *superior to alternatives*. The real test will come from ablation studies, scaling to real LMs, and demonstrating capabilities that other interpretability methods can't achieve (like predicting failure modes or enabling targeted model editing).

Critically, I've only analyzed individual rank-1 subcomponents, SPD's paper claims interpretable "mechanisms" emerge from *clusters* of multi-rank subcomponents spanning multiple layers. Without implementing the clustering step, SPD's core theoretical claims can't be fully evaluated.

References:

[1] L. Bushnaq, D. Braun, and L. Sharkey, "Stochastic parameter decomposition," arXiv:2506.20790, 2025.

[2] Finke, Lennart, Chandan Sreedhara, Thomas Dooms, Mat Allen, Emerald Zhang, Juan Diego Rodriguez, Noa Nabeshima, Thomas Marshall, and Dan Braun. "Parameterized Synthetic Text Generation with SimpleStories." *arXiv preprint arXiv:2504.09184* (2025).