

Foresight Report: Systematic generation and scoring of alternative scientific results

Xiaoliang Luo & Bradley Love

September 2025

1 Extending BrainBench

As a first step toward more comprehensive and reliable alternative results generation, we extend the BrainBench framework (Luo et al., 2024). Rather than relying on expert-edited abstracts, we automate the extraction of relevant information from full-length publications and use LLMs to generate and evaluate alternative results. This is a more realistic setting than the previous study and enables further investigation with respect to future systems that work with full-length research papers.

Specifically, we prompt GPT-4o to summarize each paper’s methods and results, then generate altered results in a separate pass. The evaluation procedure otherwise follows the original BrainBench setup using a group of open-source LLMs (see Appendix A for an overview).

We further test whether including summarized methodology and/or splitting results into minimal units (i.e., individual sentences expressing single sub-results) affects LLM evaluation of original versus altered results.

From Fig. 1, we find that the overall trends reported in BrainBench hold when using full-length papers instead of abstracts. LLMs achieve higher accuracy with summaries derived from full papers compared to abstracts when distinguishing original from altered results.

When the methods section is removed from full papers, accuracy remains comparable to that obtained with abstracts, sometimes slightly higher or lower. This decline is less pronounced than in the original BrainBench study, where removing methods from abstracts caused sharper performance drops. We attribute this difference to the richer detail in full papers, where results often implicitly reference methodological elements (e.g., reporting BOLD signals measured with fMRI). Finally, consistent with (Luo et al., 2024), accuracy decreases further when results are reduced to their smallest sub-units.

We also compare error overlaps between the original BrainBench and our full-paper results. As shown in Table 1, LLMs achieve higher accuracy with full papers, though their error patterns are similar but not identical. Identifying which papers become harder or easier when evaluated in full form may guide prompt design and help control the level of granularity in summaries. We revisit

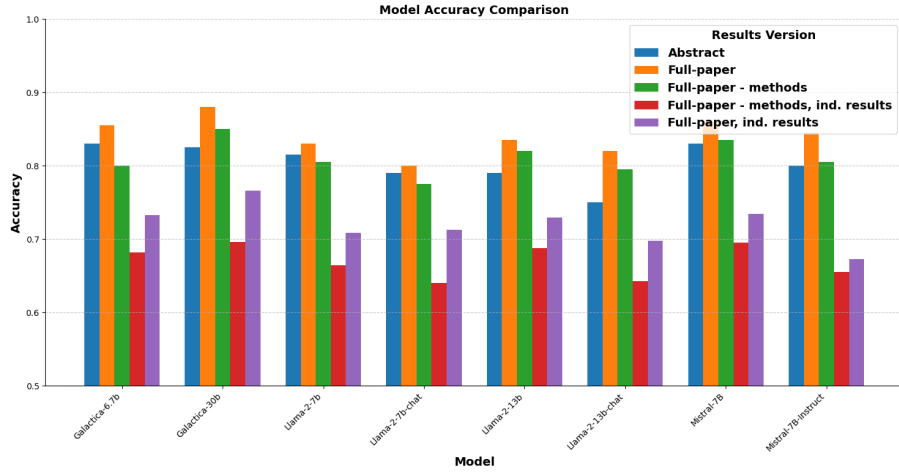


Figure 1: Extension of BrainBench to full-length papers, where GPT-4o summarises methods and results before generating altered results for perplexity-based evaluation using a collection of open-source LLMs. Accuracy is generally higher using full-paper summaries than abstracts, with less degradation when methods are removed, likely due to richer methodological detail embedded in results. Consistent with prior findings, accuracy drops when results are split into minimal sub-units.

the issue of granularity in the following section, as it remains a central challenge for systematic generation of alternative results.

Table 1: Error Correlation between Abstract and Full Paper for Each Model

Model	Abstract Acc.	Fullpaper Acc.	Identical %
Galactica-6.7b	0.83	0.855	0.745
Galactica-30b	0.825	0.88	0.775
Llama-2-7b	0.815	0.83	0.725
Llama-2-7b-chat	0.79	0.8	0.68
Llama-2-13b	0.79	0.835	0.715
Llama-2-13b-chat	0.75	0.82	0.71
Mistral-7b	0.83	0.86	0.76
Mistral-7b-Instruct	0.8	0.845	0.725

2 Systematic Alternative Result Generation via Knowledge Graphs

While the free prompting approach demonstrates the potential for generating alternative scientific results, it suffers from a key limitation: the lack of systematic control over the generation process. To address this, we propose a structured approach that combines LLMs’ generative capabilities and vast parametric knowledge with explicit structural representations in the form of knowledge graphs. This methodology provides more systematic and controllable alternative result generation by decomposing experimental findings into their constituent entities and relationships, then systematically permuting these components to produce plausible alternatives. Although our initial results show promise, limitations remain that require future investigation.

Methodology We use large language models (LLMs) to construct knowledge graphs (KGs) from neuroscience papers. In these graphs, nodes represent entities (e.g., brain regions) and edges capture relationships between them. We then generate alternative KGs that reflect plausible outcomes, which can be translated back into natural language by LLMs for further evaluation.

Example Consider Experiment 1 from the following study:

Hornsby, A. N. & Love, B. C. (2020). *How Decisions and the Desire for Coherency Shape Subjective Preferences Over Time*. Cognition.

The hypothesis was that after choosing between two options, people’s preferences shift: they not only select what they prefer but also come to prefer what they choose. To test this, participants designed a robot named GAGE (Fig. 2).

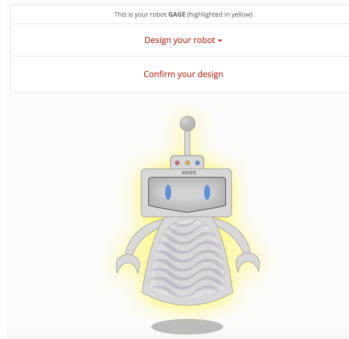


Figure 2: Example of a participant's robot design, GAGE.

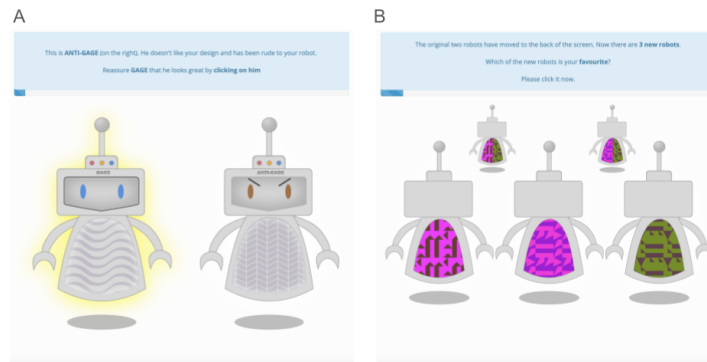


Figure 3: (A) GAGE and ANTI-GAGE presented to participants. (B) Robots reveal randomly assigned patterns on their backs.

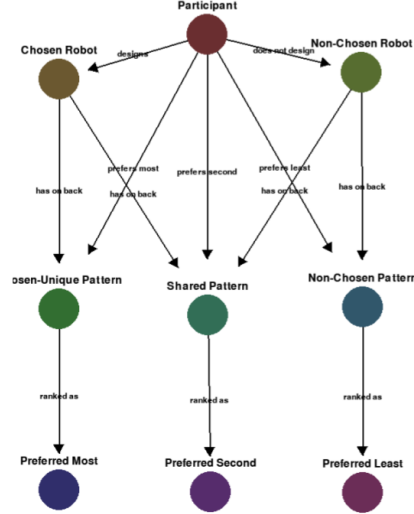


Figure 4: Knowledge graph representing the essential entities and relationships of the experiment results.

Next, participants were introduced to GAGE’s nemesis, the ANTI-GAGE (Fig. 3A).

As shown in Fig. 3B, both robots revealed one unique pattern and one shared pattern. Participants then viewed three new robots and ranked them by preference. Although the patterns were randomly assigned, the predicted preference order was:

- The new robot with GAGE’s unique pattern.
- The new robot with the common pattern shared by GAGE and ANTI-GAGE.
- The new robot with ANTI-GAGE’s unique pattern.

The results supported this prediction: preferences shifted toward chosen items (GAGE) and away from rejected ones (ANTI-GAGE). From these findings, we can construct a KG capturing the essential entities and relationships (Fig. 4). Nodes represent concepts (e.g., GAGE, ANTI-GAGE, patterns, preferences), and directed edges capture the relations between them. We made a simple python tool such that KGs can be visualized interactively, allowing users to explore the structure and consider alternative outcomes.

To create alternative patterns of results we can permute the existing KG. One approach is to swap the leaf nodes “Preferred Most”, “Preferred Second” and “Preferred Least”. This leads to six total combination of possible results (including the original one). Below, we formally define the graph permutation algorithm in general case.

Graph permutation As a first step toward systematic alternative result generation, we apply the graph permutation algorithm (Algorithm 1). This algorithm operates on a knowledge graph representation of the original paper’s results, with nodes (V) and edges (E) in standard format. Each node is assigned to a unique semantic group representing its semantic category and given a level indicating semantic granularity. The knowledge graph and semantic groups are created by prompting LLMs such as GPT-4o. Following the designing robot example, one LLM-identified semantic group is “Pattern Categories”, containing nodes “Chosen Unique Pattern,” “Shared Pattern,” and “Non-chosen Pattern”. Another group, “Outcome Metrics”, contains “Preferred Second” and “Preferred Least”. Algorithm 1 generates permuted knowledge graphs by creating combinations and permutations of groups and nodes, subject to three key constraints: (1) node swaps occur only within the same semantic group and level, (2) only leaf nodes are swapped for simplicity (for now), and (3) isomorphic graphs resulting from permutations are de-duplicated. The algorithm tracks which nodes are swapped and the degree of permutation from the original graph for subsequent analysis.

These permuted knowledge graphs are then translated back into natural language summaries of results. By combining these summaries with the previously extracted experimental methodology, we generate complete alternative results in natural language—similar to the free-prompting approach but with systematic structural control. The resulting alternatives can then be evaluated by LLMs for plausibility and coherence.

Preliminary results on BrainBench We applied KG permutation algorithms to 200 full-length BrainBench papers, processing each paper independently and evaluating permutations with open-source LLMs (Luo et al., 2024). Two models were tested: Mistral-7b (recent) and Galactica-7b (older). Across 200 items, we tracked the percentile ranking of the original among permuted alternatives (Fig. 5). Most originals ranked within the top 20% (top-1 accuracy: 0.398 for Mistral-7b, 0.439 for Galactica-7b). On average, 6.55 unique permutations were generated per paper (excluding isomorphic KGs). While accuracy declines compared to binary settings, these preliminary results suggest open-source LLMs retain some ability to identify original findings among multiple alternatives.

Enhanced knowledge graph permutation with parametric knowledge

One limitation of the current approach is that swappable nodes are restricted to entities within the same semantic group from a single paper. However, more plausible alternative entities may exist beyond the current paper’s scope of consideration. To address this constraint, we propose Algorithm 2, which combines the advantages of free-prompting that leverages LLMs’ parametric knowledge with the structured representation of knowledge graphs. This enhanced approach utilizes the vast parametric knowledge of models like GPT-4o to generate alternative node suggestions that extend beyond the confines of individual

Algorithm 1 Knowledge Graph Permutation

Require: Knowledge graph $G(V, E)$, semantic groups $\mathcal{S}$

Ensure: Set of permuted graphs $\mathcal{P}$, node swap records $\mathcal{R}$, deviation metrics $\mathcal{D}$

```
1: Initialize  $\mathcal{P} \leftarrow \emptyset, \mathcal{R} \leftarrow \emptyset, \mathcal{D} \leftarrow \emptyset$ 
2: for each group  $s \in \mathcal{S}$  do
3:    $L_s \leftarrow$  nodes with maximum level in  $s$  ▷ Extract leaf nodes
4: end for
5:  $\mathcal{S}' \leftarrow \{s \in \mathcal{S} : |L_s| > 1\}$  ▷ Filter swappable groups
6:  $\mathcal{C} \leftarrow$  all combinations of groups in  $\mathcal{S}'$ 
7: for each combination  $c \in \mathcal{C}$  do
8:    $\Pi_c \leftarrow \prod_{s \in c} \text{Perm}(L_s)$  ▷ Cartesian product of permutations
9:   for each permutation tuple  $\pi \in \Pi_c$  do
10:     $G' \leftarrow$  deep copy of  $G$ 
11:    for each group  $s \in c$  with permutation  $\pi_s$  do
12:      Apply node swapping:  $G' \leftarrow \text{swap}(G', L_s, \pi_s)$ 
13:    end for
14:     $T' \leftarrow$  extract triples from  $G'$  ▷ Check for isomorphism
15:    if  $T'$  is unique among existing graphs in  $\mathcal{P}$  then
16:       $\mathcal{P} \leftarrow \mathcal{P} \cup \{G'\}$ 
17:      Record swap operations in  $\mathcal{R}$ 
18:      Compute deviation metric and add to  $\mathcal{D}$ 
19:    end if
20:  end for
21: end for
22: return  $\mathcal{P}, \mathcal{R}, \mathcal{D}$ 
```

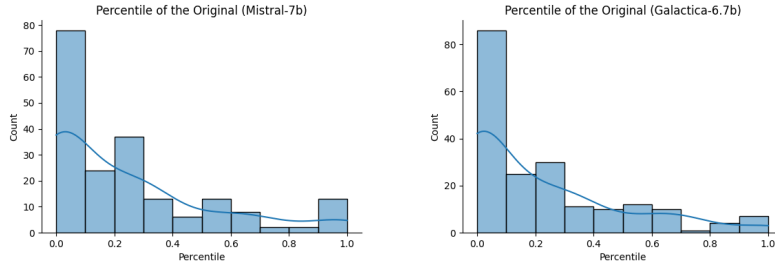


Figure 5: Percentile ranking of original results among permuted alternatives across 200 BrainBench papers. Most originals fall within the top 20%.

papers, enabling more comprehensive exploration of the alternative result space.

Algorithm 2 demonstrates clear advantages over simple node swapping. Consider this BrainBench paper:

Steel, A., Garcia, B., Goyal, K., Mynick, A. and Robertson, C.
(2023) *Scene Perception and Visuospatial Memory Converge at the Anterior Edge of Visually Responsive Cortex*. Journal of Neuroscience.

The authors’ original finding suggested that memory for visuospatial context engages a distinct neural network positioned immediately anterior to the perceptually-oriented scene areas, creating an efficient architecture for memory-perception interaction while avoiding interference. Using Algorithm 2 with LLM knowledge, we can reliably generate an alternative by substituting “anterior” with “posterior”, yielding a plausible alternative finding that locates the memory network in a different brain region. Simple node swapping without external knowledge cannot produce such contextually relevant alternatives.

Algorithm 2 has notable limitations. A more reliable approach might involve vector databases and RAG to better locate relevant literature and generate more plausible node suggestions. Additionally, the current approach lacks robust methods for verifying node consistency across the knowledge graph—we rely on LLM judgment, which requires human expert validation. When combining multiple papers using vector databases, it remains unclear how to effectively integrate overlapping knowledge graphs. We leave these challenges to future work.

Future steps TODO: Modifying edges, effect sizes etc.

Alternative result creation in literature trustworthiness settings TODO: do we briefly discuss why the analysis on retracted papers didn’t quite work?

3 Diagnosing LLM reliability through probability consistency

Our evaluations of plausibility and trustworthiness rely fundamentally on LLMs’ internal scoring of original versus alternative results. In our original study (Luo et al., 2024), we demonstrated that perplexity serves as a surprisingly effective calibrated confidence measure: larger absolute differences in perplexities between contrasting pairs correlate with higher accuracy. Furthermore, perplexity differences reflect meaningful internal judgments, as accuracy drops predictably when background and methods are removed.

Building on these findings, our current work (preprint under review at TMLR: Luo et al. 2025) explores what we consider critical for longer-term progress: developing more reliable scoring of alternative results. We investigate how

Algorithm 2 Knowledge Graph LLM-Based Node Replacement

Require: Knowledge graph $G(V, E)$, semantic groups $\mathcal{S}$, LLM M , replacement budget k

Ensure: Set of modified graphs $\mathcal{P}$, node replacement records $\mathcal{R}$, deviation metrics $\mathcal{D}$

```
1: Initialize  $\mathcal{P} \leftarrow \emptyset, \mathcal{R} \leftarrow \emptyset, \mathcal{D} \leftarrow \emptyset$ 
2: for each group  $s \in \mathcal{S}$  do
3:    $C_s \leftarrow$  candidate nodes in  $s$  for replacement  $\triangleright$  All nodes or subset
4: end for
5:  $\mathcal{S}' \leftarrow \{s \in \mathcal{S} : |C_s| \geq 1\}$   $\triangleright$  Filter groups with candidates
6:  $\mathcal{N} \leftarrow$  generate replacement strategies with budget  $k$   $\triangleright$  1 to  $k$  nodes
7: for each replacement strategy  $n \in \mathcal{N}$  do
8:   for each target node set  $N_t$  of size  $n$  do
9:     context  $\leftarrow$  extract local subgraph around  $N_t$ 
10:    prompt  $\leftarrow$  construct replacement query for  $M$ 
11:     $\triangleright$  Prompt includes: target nodes, their relationships, semantic constraints
12:     $R_{\text{uggestions}} \leftarrow M(\text{prompt}, \text{context})$   $\triangleright$  Query LLM for replacements
13:    for each suggestion  $r \in R_{\text{uggestions}}$  do
14:       $G' \leftarrow$  deep copy of  $G$ 
15:      for each replacement  $(n_{\text{old}}, n_{\text{new}}) \in r$  do
16:        Replace  $n_{\text{old}}$  with  $n_{\text{new}}$  in  $G'$ 
17:        Update all edges connected to  $n_{\text{old}}$ 
18:      end for
19:      consistency_prompt  $\leftarrow$  construct semantic validation query
20:      valid  $\leftarrow M(\text{consistency\_prompt}, G')$   $\triangleright$  LLM semantic check
21:      if valid then
22:         $T' \leftarrow$  extract triples from  $G'$   $\triangleright$  Check for isomorphism
23:        if  $T'$  is unique among existing graphs in  $\mathcal{P}$  then
24:           $\mathcal{P} \leftarrow \mathcal{P} \cup \{G'\}$ 
25:          Record replacement operations in  $\mathcal{R}$ 
26:           $d \leftarrow$  compute semantic deviation metric
27:           $\mathcal{D} \leftarrow \mathcal{D} \cup \{d\}$ 
28:        end if
29:      end if
30:    end for
31:  end for
32: end for
33: return  $\mathcal{P}, \mathcal{R}, \mathcal{D}$ 
```

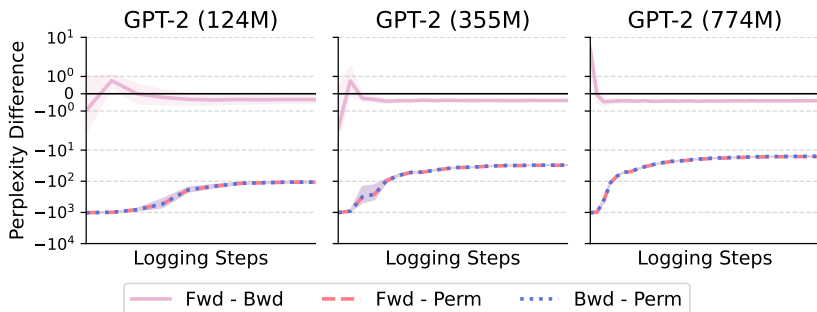


Figure 6: Original figure from Luo et al. (2025). Forward (Fwd) and backward (Bwd) text training yields similar perplexities, though forward models consistently achieve lower values (difference below zero). This gap widens slightly with model size. Permuted (Perm) text training yields much higher perplexity than both forward and backward models, with similar differences to each, causing the curves to overlap. Shaded regions indicate one standard deviation over the mean across three random initializations.

transformer-based generative models approximate probabilities at a fundamental level. Our central hypothesis is that reliable use of perplexities requires understanding whether LLM-assigned probabilities obey basic mathematical rules such as the chain rule.

We establish a theoretical basis by proving that, for a true probability distribution, sequence perplexity is invariant to the order of factorization: forward, backward, or permuted conditionals must yield the same result. Yet, in practice, we find small but systematic discrepancies between forward- and backward-trained models, amplified in models trained on permuted text (Fig. 6). We trace these inconsistencies to positional biases in self-attention (Fig. 7), extending prior findings on early-position biases (Liu et al., 2023; Xiao et al., 2024). To highlight, we show attention has positional biases toward nearby tokens and those at maximal distances (Fig. 7; more results see the original paper) which leads to discrepancies to the theoretical equivalence we established.

These discrepancies, while subtle, are highly informative: they expose where LLMs diverge from valid probability distributions and may underlie phenomena such as hallucination and brittle out-of-distribution behavior. Understanding these deviations is therefore not a narrow technical detail but a fundamental step toward trustworthy evaluation. By treating probability consistency as a diagnostic lens, we can better characterize learning dynamics and design mitigation strategies. For example, sibling models could be trained jointly to minimize divergence in perplexity, encouraging more stable learning. Unlike BERT (Devlin et al., 2019), which achieves symmetry through bi-directional attention without modeling joint probabilities, our approach directly targets probabilistic consistency.

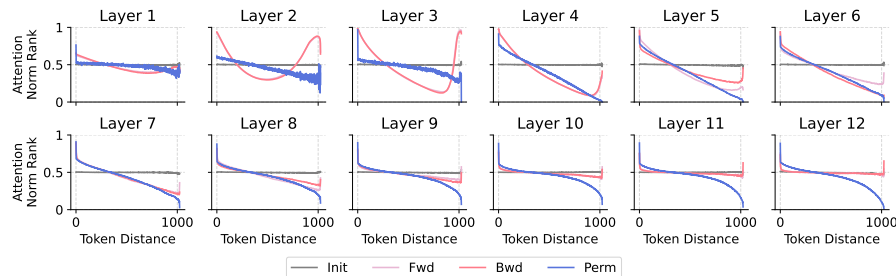


Figure 7: Original figure from Luo et al. (2025). Positional bias in self-attention varies with training directions and layers (GPT-2 124M). Normalized attention rank ($\min = 0, \max = 1$) is plotted as a function of token distance within the context, averaged across heads, sampled sequences, and layers. Compared to models at initialization (Init), forward (Fwd) and backward (Bwd) trained models show strong positional biases toward both nearby tokens and tokens at maximal distance, with the degree of bias varying across layers. In contrast, the model trained on permuted text (Perm) displays distinct patterns, with positional bias generally decreasing as token distance increases across most layers. All test sequences fully span the context window each prepended with a BOS token.

In short, this work lays essential ground for making LLM evaluations both more reliable and more interpretable. Establishing whether perplexities truly reflect coherent probability distributions may appear indirect, but it is a necessary foundation for diagnosing, explaining, and ultimately mitigating the limitations of large language models.

4 Collaborations and community engagement

We partnered with AE Studio¹ to establish our knowledge graph permutation code as an open-source repository². Together, we designed the repository structure and determined how the community—including software engineers, machine learning researchers, and domain experts—could interact with and contribute to different aspects of the project. AE Studio provided essential scaffolding and implemented best practices for repository setup. We also held initial discussions on front-end design, envisioning a user-friendly interface that would help researchers explore alternative result patterns in their experiments and existing literature while providing broader context on supporting and contrasting evidence, consistency evaluations, and trustworthiness assessments.

Additionally, we established a Discord server and invited BrainBench co-authors and BrainGPT newsletter subscribers to test and interact with the

¹<https://ae.studio>

²<https://github.com/don-tpanic/alt-core-playground>

repository as early testers and contributors. This process revealed limitations and challenges in running the repository as a fully public project, primarily due to unexpected complexity and engagement requirements. We are currently working on potential solutions to address these issues.

We have received media coverage, including a feature in The Transmitter, a neuroscience news site for researchers. The article highlighted various ways neuroscientists are using LLMs in their work and included our project as an example of efforts to develop LLMs for scientific discovery.

We have also interacted with domain experts who are potential users of our technology. While anecdotal, these interactions demonstrate the significant impact our work could have in research areas where literature is becoming overwhelming and where mistakes carry serious consequences. A neurologist from MassGeneral Institute for Neurodegenerative Disease contacted us with a compelling example. While researching Parkinson’s disease biomarkers, he discovered elevated polyamine levels in spinal fluid—a potentially groundbreaking finding for diagnosis or tracking disease progression. However, he later uncovered 1980s and 1990s studies suggesting a connection between polyamines and MAO-B inhibitors, common Parkinson’s medications. A follow-up study confirmed that these inhibitors caused the polyamine increase, preventing him from making a misleading breakthrough claim. When he tested BrainGPT (v0.2) with both his original findings and altered versions falsely claiming polyamine independence from MAO-B inhibitors, the system accurately identified the MAO-B-linked results as most probable. This demonstrates LLMs’ ability to uncover overlooked research connections across vast scientific literature that exceed human capacity to process.

5 Discussion

A BrainBench Overview

BrainBench (Luo et al., 2024) is a benchmark consists of 200 test cases from abstracts in the *Journal of Neuroscience* published in 2023. These abstracts are categorized into five sections: Behavioral/Cognitive, Systems/Circuits, Neurobiology of Disease, Development/Plasticity/Repair, and Cellular/Molecular.

Each test case contains a published abstract and an altered version crafted by neuroscientists (see details in Luo et al. (2024)). These modifications, though minimal, significantly change the results—for instance, by changing the roles of brain regions or reversing a result’s direction (e.g., from “decreases” to “increases”). The altered abstracts remain logically coherent despite the changes.

The BrainBench task is to identify the correct study outcome by choosing between the original abstract and its altered counterpart.

BrainBench Model Evaluation Two versions of the abstracts from each test case were presented to models separately. We measured the perplexity of

both passages and used perplexity as the indicator of whether models favor one abstract or the other.

Perplexity measures the degree of uncertainty of a model when generating a particular sequence of text and is defined as the exponentiated average negative log-likelihood of a tokenized sequence. If we have a tokenized abstract sequence $X = (X_0, X_1, \dots, X_t)$, then the perplexity of X , given a model parameterized by θ is,

$$PP(X) = \exp \left\{ -\frac{1}{t} \sum_i^t \ln p_\theta(X_i | X_{<i}) \right\} \quad (1)$$

where $\ln p_\theta(X_i | X_{<i})$ is the log-likelihood of the i th token conditioned on the preceding tokens $X_{<i}$ according to the model. Given both the original and the altered abstracts, we used the abstract with lower perplexity as the model’s decision and evaluated the overall accuracy across the entire BrainBench dataset accordingly.

References

- Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. URL <http://arxiv.org/abs/1810.04805>, arXiv:1810.04805 [cs]
- Liu NF, Lin K, Hewitt J, Paranjape A, Bevilacqua M, Petroni F, Liang P (2023) Lost in the Middle: How Language Models Use Long Contexts. DOI 10.48550/arXiv.2307.03172, URL <http://arxiv.org/abs/2307.03172>, arXiv:2307.03172 [cs]
- Luo X, Recharadt A, Sun G, Nejad KK, Yáñez F, Yilmaz B, Lee K, Cohen AO, Borghesani V, Pashkov A, Marinazzo D, Nicholas J, Salatiello A, Sucholutsky I, Minervini P, Razavi S, Rocca R, Yusifov E, Okalova T, Gu N, Ferianc M, Khona M, Patil KR, Lee PS, Mata R, Myers NE, Bizley JK, Musslick S, Bilgin IP, Niso G, Ales JM, Gaebler M, Ratan Murty NA, Loued-Khenissi L, Behler A, Hall CM, Dafflon J, Bao SD, Love BC (2024) Large language models surpass human experts in predicting neuroscience results. *Nature Human Behaviour* DOI 10.1038/s41562-024-02046-9, URL <https://www.nature.com/articles/s41562-024-02046-9>
- Luo X, Xu X, Ramscar M, Love BC (2025) Probability Consistency in Large Language Models: Theoretical Foundations Meet Empirical Discrepancies. DOI 10.48550/arXiv.2505.08739, URL <http://arxiv.org/abs/2505.08739>, arXiv:2505.08739 [cs]
- Xiao G, Tian Y, Chen B, Han S, Lewis M (2024) Efficient Streaming Language Models with Attention Sinks. DOI 10.48550/arXiv.2309.17453, URL <http://arxiv.org/abs/2309.17453>, arXiv:2309.17453 [cs]