**Introduction and Objective**

There is growing research interest for using large language models (LLM) to study language and cognitive processes. We want to use LLM to simulate aphasia, a language disorder. By perturbing a LLM to mimic aphasia-like behavior, we seek to understand the underlying mechanism of aphasia in artificial system. This insight may help us understand similar processes in the human brain. The primary objective of this project is to simulate aphasia in pretrained LLM and analyze the resulting linguistic error.

**Methodology**

We have selected the Large Language and Vision Assistant (Llava) LLM among other multimodal LLM such as Flamingo, and Phi-3.5. Specifically, we used the Llava-1.6 Vicuna-13B version, which achieved 90% accuracy in the benchmark testing.

The benchmark we used is the Philadelphia Naming Test (PNT), which is widely used to evaluate language functions. PNT consists of 185 images (10 warmup,175 testing). Each image depicts an object. The participant is instructed to describe the image in one word. Based on the word produced, we can characterize and classify the type of word error, which are then used to identify aphasia type. Table 1 outlines the types of word-level errors, and the methods used to identify them. ‘Correct’, ‘No Response’, and other errors are mutually exclusive. Phonological, generalization, thematic, taxonomic, unrelated, non-word may overlap.

A screenshot of a computer

Description automatically generated

Table 1. word-level error and method to identify them.

To simulate aphasia, we perturbed the LLM by modifying its parameter weights using two approaches: adding noise and zeroing weights. All perturbations are applied to the transformer layers of the model. There are 40 transformer layers in the model. The architecture of the Llava model is illustrated in Figure 1.

A diagram of a computer

Description automatically generated Figure 1. Architecture of the Llava large language model.

**Results and discussion**

*Adding Noise*

We modified one transformer layer at a time, the specific parameters are:

* Attention weights: self\_attn.q\_proj, self\_attn.k\_proj, self\_attn.v\_proj, and self\_attn.o\_proj.
* Feedforward MLP: mlp.gate\_proj, mlp.up\_proj, and mlp.down\_proj.
* Normalization layers: input\_layernorm and post\_attention\_layernorm.

We experimented with two parameters: percentage of weights to modify and percentage of noise to add.

With unmodified weights, we have baseline inference accuracy of 166 out of 185 images correctly named. We discarded the mis-predicted images in the result analysis. With 40 layers, we have 40x166 = 6640, so a minimum of 6640 total count of classification. We may have more than that when an inference output belongs to multiple types of word error.

The result for changing 10% of the weights at different noise levels is shown in Figure 2. More experiment results can be found in (<https://github.com/csce585-mlsystems/Aphasia_LLM/tree/main/Paper_report_supplement>).

When the noise level increased, the number of correct predictions decreased. However, beyond a noise level of 0.11, errors shifted predominantly to "nonword" outputs, rather than increasing other aphasia-like errors. For noise below 0.1, the ratio of nonword to other errors is close to 50%.

A graph of different colored bars

Description automatically generated

Figure 2. Word error classifications across all 40 layers with 10% weight modification.

We are also interested in how each individual error changes across different layers of the LLM. The result is shown in Figure 3.

* Correct: Later layers were more resistant to noise.
* Nonwords: Errors also decreased in later layers.
* No-response: These errors occurred only in earlier layers.
* Taxonomic, generalization, thematic, and phonological: These errors showed a cone-shaped progression from middle to later layers.
* Unrelated: These errors were more scattered, occurring from early to middle layers.

A green and purple gradient

Description automatically generated

A close-up of a colorful background

Description automatically generated

A blue and yellow background

Description automatically generated with medium confidenceA blue and green light

Description automatically generated A blue and green background

Description automatically generated with medium confidence A blue and green background

Description automatically generated with medium confidence A blue and green gradient

Description automatically generated with medium confidenceA blue and yellow background with white text

Description automatically generated with medium confidence

Figure 3. Word-level errors across LLM layers at 10% weight modification for varying noise levels.

*Zeroing*

For the zeroing approach, we began by modifying one transformer layer at a time, starting with changing 10% of the weights. We found only the first two layers of the LLM output showed ‘nonword’ response, every other layer still generated correct predictions. Then we increased the percentage of weights zeroed in increment of 10%, all the way up to 100%. To our surprise, we observed the same behavior: only modifying the first two layers impacted prediction. So even when one layer is completely zeroed, information can still propagate to the next layer.

Then we tried to zero up to 5 consecutive (e.g., 1,2,3,4,5) and non-consecutive layers (e.g., 1,2,4,7,9). The result was the same. We verified that the code was correct. It seems the LLM is very resilient. It could also be naming one picture is not fully utilize the model. So, it still functions when large portion of it is disabled.

Next Step

For the adding noise approach, we can refine the experiments to maintain ~50% correct predictions by varying the noise level and percentage of weights modified. Then we analyze how different types of word errors evolve under these controlled conditions

For zeroing, we need to increase the number of layers modified to be higher, to identify the point where the output in the later layers shows changes.