

An Investigation of GPT-2 Applications and Training Improvements, and Exploring Multi-Token Entity Predictions

Stanford CS224N Default Project

Ben Wengreen
B.S./M.S.,
Stanford University CS
bensonw@stanford.edu

Jeffrey Meng
M.S.,
Stanford University CS
jkmeng@stanford.edu

Bhavya Ashish Shah
M.S.,
Stanford University CS
bhavya@stanford.edu

Abstract

High training costs and the intractability of exploring multi-token prediction spaces are two challenges LLMs face. Multi-token prediction spaces are difficult because of exponential branching, and difficulty determining whether a branch is complete can be pruned. In this project, we tackle both of these problems. First, we use GPT-2 as a foundation model for three distinct NLP tasks: sentiment classification, paraphrase detection, and sonnet generation. We found that this approach was acceptable. We then explore using LoRA, finding that it significantly reduces trainable parameters while maintaining baseline accuracy. Next, we experimented with Flash Attention and windowed attention, finding that they maintained performance and predict they might reduce training time for tasks requiring longer context windows. Next, we propose a novel method of detecting when a model finishes a multi-token answer to a cloze-style question using a feature vector of internal model signals – such as next-token entropy and confidence gap – from a three token window feed to an MLP trained on a custom 45,459 example dataset, finding that this text-unaware approach showed improvements over a text-aware approach, especially in otherwise ambiguous cases. We also use our multi-token prediction and clustering pipeline to explore the impact on accuracy of plausibility and correctness in multi-shot examples, finding that plausibility had a larger impact than correctness; and investigate the alignment of the un-finetuned Qwen2-1.5B model with human preferences on subjective questions, finding some alignment but significant discrepancies.

1 Key Information to include

- Mentor: Ali Sartaz Khan
- External Collaborators: None
- Sharing project: NA
- Late Days: See right

Team member	Late days remaining
Ben	5
Jeffrey	0
Bhavya	3
Total pooled	8
Team size	3
Late days used	2 group (6)

2 Introduction

Large language models are extremely good at a wide variety of natural language processing tasks. However, using these foundation models means bearing the immense computational cost of training them. Solving this problem is important as models scale to trillions of parameters and the resources required for training and fine-tuning balloon to thousands of GPUs and millions of dollars in compute for state-of-the-art models. The status quo sidelines cash-strapped academic researchers

and companies in developing countries from the state-of-the-art. This problem is difficult due to how transformers fundamentally work. During training, the masked (causal) multi-head self-attention mechanism scales quadratically with sequence length, creating severe memory and time bottlenecks.

To address the training bottleneck, our key idea is to explore architectural and parameter-efficient optimizations. We establish a GPT-2 baseline across three downstream tasks and (a) apply Low-Rank Adaptation, and (b) replace masked self-attention with more efficient variants like Windowed Attention and Flash Attention.

The other major challenge is the intractability of exploring multi-token prediction spaces. The possible multi-token predictions branch exponentially with sequences of undefined length. Despite this, these spaces represent an LLM’s knowledge, decisions, and in the age of agentic AI, actions. This makes understanding these spaces very important, despite the intractability.

We propose a novel solution to this problem for cloze-style questions using a text-unaware, signal-based MLP trained on a custom 45,459 example dataset to predict answer boundaries. This would allow us to cut off branches when they no longer impact the answer space, making the multi-token prediction for cloze-style questions with undefined answer spaces not completely intractable.

We then demonstrate our multi-token prediction and clustering pipeline on exploring the effects of correctness and plausibility on answer accuracy, and exploring un-finetuned model alignment to human preferences for subjective questions.

3 Related Work

To address the training bottleneck, our key idea is to explore architectural and parameter-efficient optimizations. We establish a GPT-2 baseline across three downstream tasks and (a) apply Low-Rank Adaptation, and (b) replace masked self-attention with more efficient variants like Windowed Attention and Flash Attention. For inference, our key idea is to use the LLM’s own internal probability distributions by using a text-unaware, signal-based method that uses internal metrics such as next-token entropy and confidence gaps as structural boundaries. We then ask two questions: how the plausibility of multi-shot examples influences generation accuracy, and whether a non-instruction-tuned model’s zero-shot outputs naturally align with human preferences.

To address the quadratic time and memory complexity of the causal self-attention mechanism [1], several architectural optimizations have been proposed. Beltagy et al. [2] proposed windowed attention in the Longformer, limiting the receptive field to local tokens to achieve linear scaling. Also, Dao et al. [3] proposed FlashAttention, an exact, IO-aware algorithm that speeds up computation by minimizing memory reads and writes without approximating the attention matrix. We implement and compare both approaches to evaluate their practical speed-accuracy tradeoffs. Hu et al. [4] proposed Low-Rank Adaptation (LoRA), which freezes pre-trained weights and injects trainable rank decomposition matrices. We replicate LoRA to verify its parameter-efficiency on our baseline architecture.

In terms of answer extraction from open-ended outputs, traditional extraction pipelines often rely on external, text-aware parsers like spaCy [5], which can break in ambiguous contexts. Recent work by Kadavath et al. [6] shows that an LLM’s internal token probabilities and entropy can tell us something about its confidence and thus its factual accuracy. Build on this idea by using next-token entropy and other internal metrics as signals for predicting multi-token answer boundaries.

In terms of prompting, Min et al. [7] showed that the exact correctness of multi-shot examples matters less than the overall format and distribution of the prompt. We run experiments to conduct more granular analysis on this front. Also, while methods such as Reinforcement Learning from Human Feedback [8] are the standard for aligning models to human preferences, we explored the innate alignment of a non-instruction-tuned foundation model.

4 Experiments and Approaches

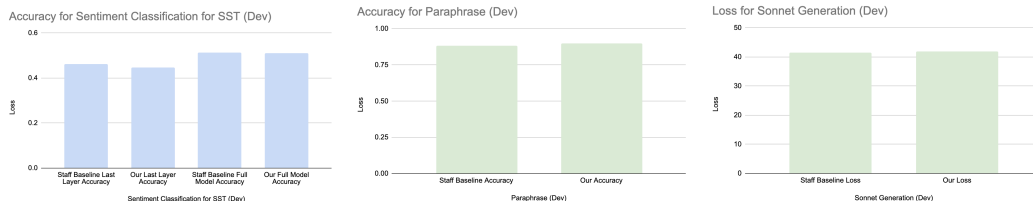
Our project explores the use of GPT-2 as a foundation model for various NLP tasks in 4.1, the use of LORA to improve training in 4.2, and the effects of alternative attention mechanisms in 4.3.

The next part of our project explores multi-token entity prediction. 4.4 explores using internal LLM signals to predict when a cloze-style question has been answered, which is used to terminate branches when exploring possible completions. 4.5 uses multi-token entity prediction to explore the impact of the correctness and plausibility of multi-shot examples on output accuracy. 4.6 uses multi-token entity prediction to explore alignment between human and model preferences on subjective questions.

4.1 Experiment 0: Default Project Tasks

For this experiment, we implemented, trained, and tested the default sentiment classification, paraphrase detection, and sonnet generation tasks with the default training parameters. The training parameters, datasets, training and model specifics have been left out for brevity, but can be found in the default project instructions and or starter code if interested.

The results of our default training compared to the staff baselines are shown in the following charts.



Our results with the default training parameters are very comparable to the staff baselines. We also trained larger GPT-2 models with tuned hyperparameters for our official score submissions.

An example of a sonnet created by the fine-tuned sonnet GPT-2 model can be found in A.1.1.

This experiment demonstrated that large language foundation models like GPT-2 can be fine-tuned and used to perform an assortment of specific NLP tasks.

4.2 Experiment 1: Parameter-efficient Fine Tuning with LoRA

As models get larger and larger, it becomes less and less practical to retrain all model parameters when doing fine-tuning. A common technique for making the fine-tuning process more efficient is Low-Rank Adaptation [4], which works by freezing the model weights and instead training smaller rank decomposition matrices. We implemented this technique in our model and found that it significantly reduced the time and resource requirements of training, allowing us to iterate faster and train larger base model sizes.

4.2.1 Setup and Data

We used LoRA to help with training all three tasks, but did most of our analysis on the paraphrase detection task, because it was the most resource-intensive to train, and therefore we got the most gains by applying LoRA here.

The main hyperparameter for our LoRA implementation is the rank. Using our model without LoRA as a model, on the smallest gpt size, we investigated how rank size affected accuracy.

We also did some minor investigation on the scaling factor (alpha).

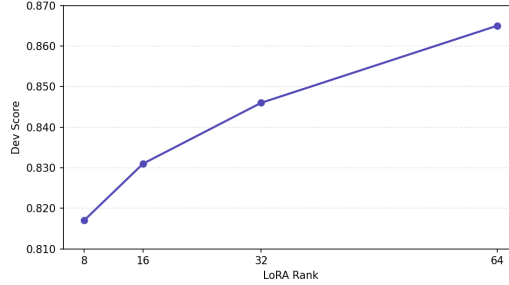
4.2.2 Results

First, we verified that LoRA significantly reduced our count of trainable parameters. On the small gpt size, with rank 32, we trained only 591,362 / 125,621,762 parameters (0.47%).

We found that in general, the larger the rank we used for LoRA (up to 64), the better the performance of the model. However, the real improvements came from training larger model sizes, which is only possible because LoRA improves the efficiency of training, allowing for training larger models with the same underlying resources.

We then experimented with the parameters we applied LoRA to. We initially only trained the attention query and value parameters, hypothesizing that fine-tuning only on these might be sufficient for reasonable accuracy and lead to faster training.

However, we ultimately found that training additional parameters for intermediate layers was necessary to improve performance, though it came at a cost of increasing the number of parameters we need to train (as shown in the graph).



4.3 Experiment 2: Alternative Attention Mechanisms for Training

As sequence lengths increase, so does the computational cost of training models due to standard causal self-attention, creating memory and time bottlenecks in training. We implemented and evaluated three different attention mechanisms to understand their trade-offs between computational efficiency and downstream accuracy to mitigate this bottleneck.

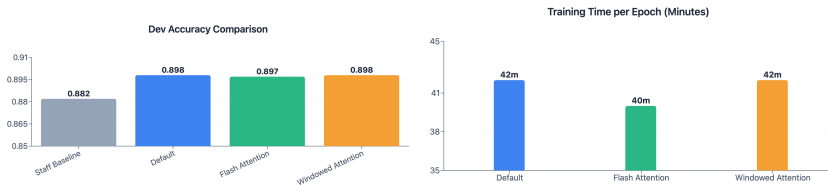
4.3.1 Setup and Data

We evaluated using the Quora paraphrase detection dataset, which consists of sentence pairs formulated into cloze-style sequences. Note that the average context window used in training and evaluation is fairly short for this task.

We modified our core causal self-attention layer to support two additional attention mechanisms:

- **Default Attention:** The baseline implementation computing standard scaled dot-product attention with an upper-triangular causal mask to prevent attending to future tokens.
- **Flash Attention:** A fast, memory-efficient attention mechanism that computes the exact same output as default attention. Instead of manually computing the intermediate attention scores and probabilities one by one, we implemented this using PyTorch’s built-in `scaled_dot_product_attention` function, which consolidates into a single step.
- **Windowed Attention:** A localized attention mechanism that restricts a token’s receptive field to a sliding window of 64 tokens. We implemented the window restriction by adding an additional lower-triangular mask that blocks tokens more than 64 positions in the past. We still compute the full attention score matrix before masking, so we can evaluate accuracy implications of local attention before evaluating computational speedups from sparse attention.

4.3.2 Results



The comparable accuracies across all three approaches suggest that the 64-token receptive field may be enough for capturing the connections required by the paraphrase detection task. In terms of training efficiency, flash attention provided a marginal speedup, which makes sense because if the computational cost of causal attention scales with sequence length, and the sequences in the Quora dataset are relatively short, attention computation is not the bottleneck. We expect that when applied to tasks with significantly longer context windows, flash attention and an optimized implementation of windowed attention would lead to larger improvements in training efficiency compared to the default attention mechanism.

4.4 Experiment 3: Learning Multi-Token Answer End Detection

The premise of this section is exploring possible multi-token LLM outputs using branches going along token probability paths. This is in contrast to other work that only looks at single token or numerical answers (which have limited applications) and naive sampling methods (which cannot efficiently explore low probability completions).

The method proposed in this section uses caching of past sequences to explore the branches of possible LLM outputs, even low probability outputs, very efficiently and combining this with techniques to decide when a branch should stop being explored and what the LLM's answer is for that branch.

Initially this was going to be explored for GPT-2, but it's completions weren't meaningful enough for an interesting exploration, so Qwen2-1.5B was used instead.

4.4.1 Setup and Data

First, spaCy (an NLP library which implements the "en_core_web_sm" tokenizer, tagger, parser, and noun entity recognizer [NER]) was used in conjunction with rule-based code to create an initial branch end condition and answer extractor.

Then, this was used to explore the completions of Qwen2-1.5B (not instruction tuned) for 29 questions (with four multi-shot examples), which created over 10,040 completion-answer pairs. Then, these were manually reviewed (with some code to find potential common mistakes – such as "[Answer]. A person" and "[Answer]. (A)" – before reviewing one by one) and some ambiguous cases where the LLM did not output a real answer were removed. This resulted in 10,040 examples that were processed by 3-token windows to create a final dataset of 45,459 labeled feature vectors. The dev and test splits were given the completions and vectors from four questions each, resulting in a 32,745/8,076/4,638 train/dev/test split.

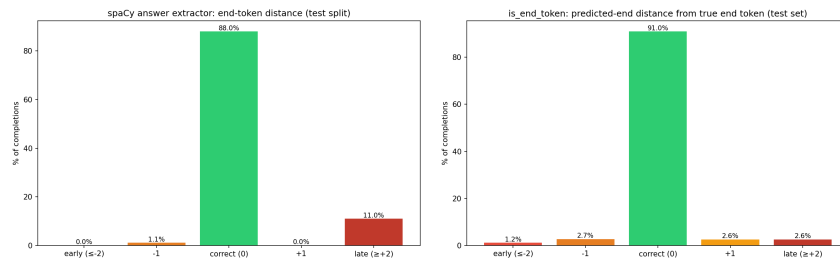
The extracted features were the completion character count, the completion tokens count, the entropy difference between the three tokens (two values), and, for each of the three token positions: the prob of the highest token, the confidence gap (1st highest prob minus 2nd highest prob), and the entropy.

This was used to train an MLP with the following architecture to predict answer boundaries: Input(13) → Linear(64) → ReLU → Dropout(0.1) → Linear(32) → ReLU → Linear(1). In training, each question was weighted equally, then each question's completions contributed equal (instead of weighting by completion probability) with Weighted Binary Cross-Entropy loss.

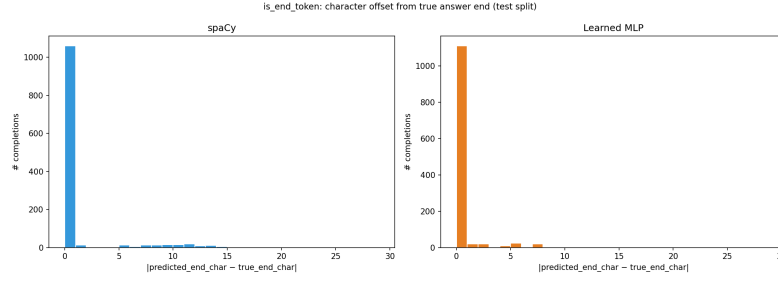
4.4.2 Results

In the initial investigation, the graphs for sentence perplexity, next-token entropy, confidence gaps, and token probability distribution variance showed signal but clearly had enormous noise when looking at individual completions, so it was unexpected for the model to perform well. (See A.2.1 for graphs and analysis.)

SpaCy vs Learned Answer Boundary Accuracy (Tokens)



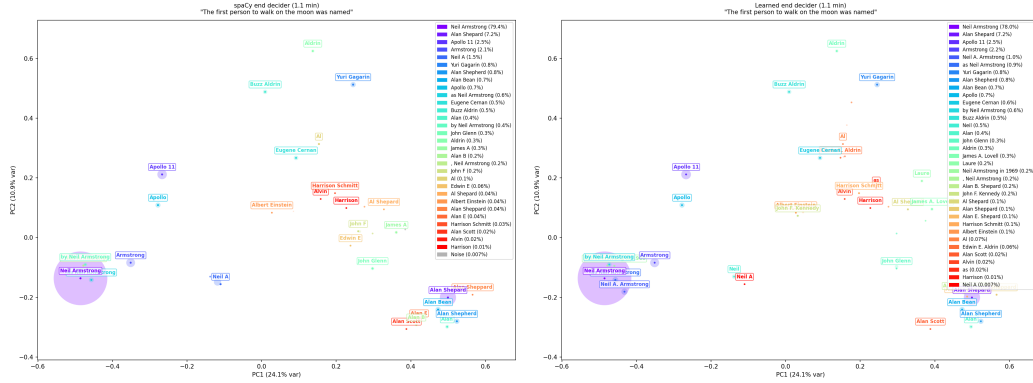
SpaCy vs Learned Answer Boundary Accuracy (Characters)



The evaluation of both methods on the reserved test data suggests that the text-unaware signal-based method outperformed the text-aware spaCy-based method, with the spaCy method being exactly right 88% of the time, compared to the learned method's 91%.

The learned method was also tested in practice, with some of the results shown in the cluster graph below.

SpaCy vs Learned Exploration



Notably, the learned method improved on the spaCy method in the ambiguous case of "Neil A. Armstrong", which could either be one name or two sentences with two names. While the text-aware spaCy method classifies this as two names (resulting in "Neil A"), the text-unaware, signal-based method correctly classifies this as one name – "Neil A. Armstrong" – using internal signals of the LLM instead of relying on the raw text.

This experiment proves that internal signals of an LLM can be used to determine when an answer to a cloze-style question is generated with reasonable accuracy, even without awareness of the actual text/tokens, potentially with better performance than traditional text-aware approaches.

4.5 Experiment 4: Multi-Shot Example Correctness Effect

4.5.1 Setup and Data

This experiment used the previous work to explore the effect of correct, plausible but incorrect, and implausible and incorrect multi-shot examples on the accuracy/correctness of the LLMs outputs. For brevity, some of the specifics of this experiment were left out.

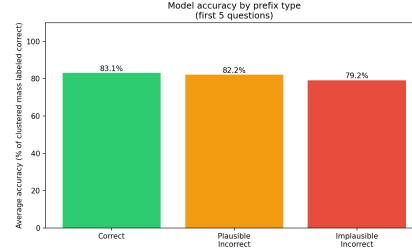
Without multi-shot examples, the non-instruct Qwen model (lacking instruction tuning) severely struggled to output answers and instead tried to continue the cloze-style question (e.g. "The author of Harry Potter is named" -> " after her grandmother and ...").

Nineteen correct example's of completed cloze-style questions were created, then two LLM calls were used to create plausible incorrect examples and implausible incorrect examples (see A.3.1 for examples).

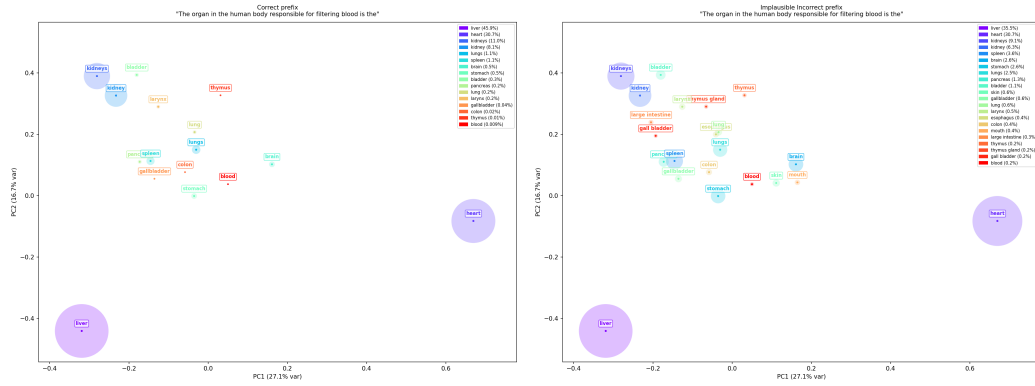
For each trial of each category, four random examples were given before one of the five cloze-style questions. The clusters were then manually labeled. (Only five questions [listed in A.3.2] were explored because of labeling time constraints.)

4.5.2 Results

Correct and plausible incorrect examples had similar results (83.1% and 82.2%), while implausible incorrect examples had a more significant reduction in accuracy (79.2% – a 23.1% increase in incorrect answer frequency).



Correct Examples vs Implausible Incorrect Examples



In the cluster graphs above, observe how giving implausible incorrect examples shifts probability from the correct answers – liver and kidney(s) – into previously near-zero-probability incorrect answers – such as stomach, brain, lungs, bladder, and spleen – and creates a larger set of incorrect answers. (A full set of 3 cluster graphs is in A.3.3.)

This experiment demonstrates that even incorrect examples can be used in a multi-shot fashion to guide model outputs, however, correct – or at least plausible – answers have better performance.

4.6 Experiment 5: Model Human Preference Alignment

4.6.1 Data

This experiment compares the LLM’s answers to subjective cloze-style questions to human preferences. The data for human preferences are from polls conducted by YouGov (see references for the polls).

The poll questions given to human participants were manually converted to cloze-style questions. The YouGov polls were selected such that the poll question could easily be converted into a cloze-style question for the LLM.

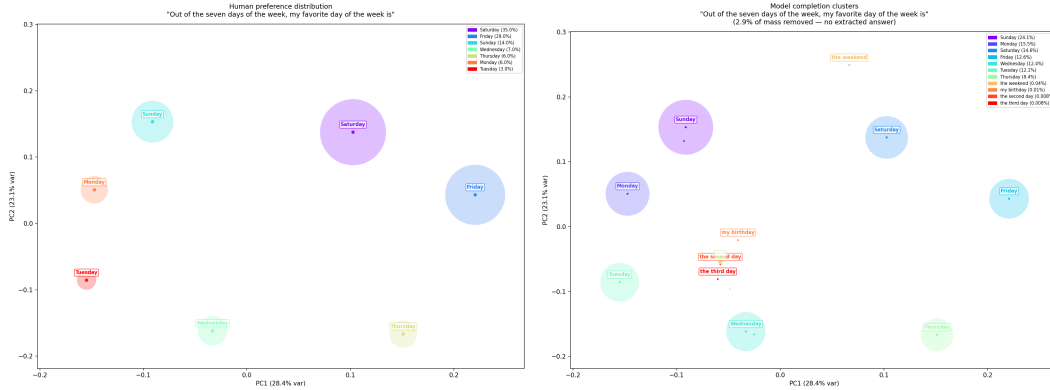
4.6.2 Setup

This experiment uses the spaCy-based explore and answer extractor with the Qwen2-1.5B model from Experiment 4.4 with a general multi-shot example prefix to encourage the model to output an answer instead of continuing the cloze-style question.

For each cloze-style question, the LLM answers were plotted on the two principal components of their SentenceBERT embeddings, then the human answers were embedded and plotted on the same axes.

4.6.3 Results

Five pairs of human preference and model answer cluster graphs were created. One graph is shown below (see A.4.1 for more).



Overall, the LLM's "preferences" were clearly associated with human preferences (such as ranking London over New York and Vanilla over Chocolate), however the preferences definitely didn't fully match, even when the trends did. For example, in the cluster graph above, the LLM ranked Monday the second most often as it's favorite day of the week, while polled humans marked it the second *least* frequently. But the model did rank the weekend highly like the humans, at least demonstrating association.

This experiment suggests that LLMs (specifically Qwen2-1.5B) exhibit some degree of alignment with human opinions even without fine-tuning, but the alignment is off for some cases.

5 Team Contributions

Ben did the default downstream tasks, the multi-token analysis experiments and much of the report writing. Jeffrey implemented the initial GPT-2 and attention code, PEFT with LoRA, managed many of the large training runs, submissions to leaderboards. Bhavya worked on the Adam optimizer, attention mechanisms, and some report writing.

6 Conclusion

We addressed computational bottlenecks in LLM training and the intractability of multi-token answer extraction during open-ended inference. We learned that while Low-Rank Adaptation significantly reduces trainable parameters without sacrificing performance, optimizations like Flash and Windowed Attention do not translate into meaningful gains on short context tasks. We found that our novel text-unaware answer boundary approach improved on the text-aware approach, especially in otherwise ambiguous cases. We also found that generation accuracy is more sensitive to the plausibility rather than just correctness of multi-shot examples. Finally, we found that even un-finetuned models have a latent, though imperfect, alignment with subjective human preferences.

In terms of limitations, we did not fully stress-test the alternative attention mechanisms due to our short context required for our tasks. Also, manual labeling constrained the scale of our exploration into the effects of correctness and plausibility, and our boundary-detection model was trained on a limited number of questions. Our multi-token prediction work was also limited to questions with noun-phrase answers.

Future work could evaluate the optimized attention methods on long-document tasks and explore multi-token prediction for questions with answers of different parts of speech (such as adjectives) using the more extendable learned signal-based approach, since the spaCy method depends on noun entity recognition.

References

- [1] A. Vaswani et al., “Attention Is All You Need,” 2017. <https://arxiv.org/abs/1706.03762>
- [2] I. Beltagy, M. E. Peters, and A. Cohan, “Longformer: The Long-Document Transformer,” 2020. <https://arxiv.org/abs/2004.05150>
- [3] T. Dao et al., “FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness,” 2022. <https://arxiv.org/abs/2205.14135>
- [4] E. J. Hu et al., “LoRA: Low-Rank Adaptation of Large Language Models,” 2021. <https://arxiv.org/abs/2106.09685>
- [5] M. Honnibal et al., “spaCy: Industrial-strength Natural Language Processing in Python,” 2020. <https://spacy.io>
- [6] S. Kadavath et al., “Language Models (Mostly) Know What They Know,” 2022. <https://arxiv.org/abs/2207.05221>
- [7] S. Min et al., “Rethinking the Role of Demonstrations: What Makes In-Context Learning Work?,” 2022. <https://arxiv.org/abs/2202.12837>
- [8] L. Ouyang et al., “Training Language Models to Follow Instructions with Human Feedback,” 2022. <https://arxiv.org/abs/2203.02155>
- [9] L. Sanders, “Americans’ Opinions on 20 Different Music Genres, from Classic Rock to Hip-Hop and Rap,” YouGov, 2023. <https://today.yougov.com/entertainment/articles/45699-americans-opinions-different-music-genres-poll>
- [10] T. Orth, C. Bialik, and L. Sanders, “America’s Favorite Ice-Cream Flavors: Vanilla, Chocolate, and Mint Chocolate Chip,” YouGov, 2022. <https://today.yougov.com/society/articles/43178-americas-favorite-ice-cream-vanilla-chocolate>
- [11] YouGov, “What Is America’s Most and Least Favorite Day of the Week?,” 2021. <https://today.yougov.com/society/articles/34696-most-and-least-favorite-day-week-poll>
- [12] YouGov, “What Is America’s Favorite (and Least Favorite) Month?,” 2021. <https://today.yougov.com/society/articles/34356-favorite-least-favorite-month-poll>
- [13] YouGov, “Do You Think London or New York Is Overall the Better City?,” YouGov Daily Question, 2025. <https://today.yougov.com/topics/travel/survey-results/daily/2025/07/11/d613a/2>

A Appendix

A.1 Default Project Tasks

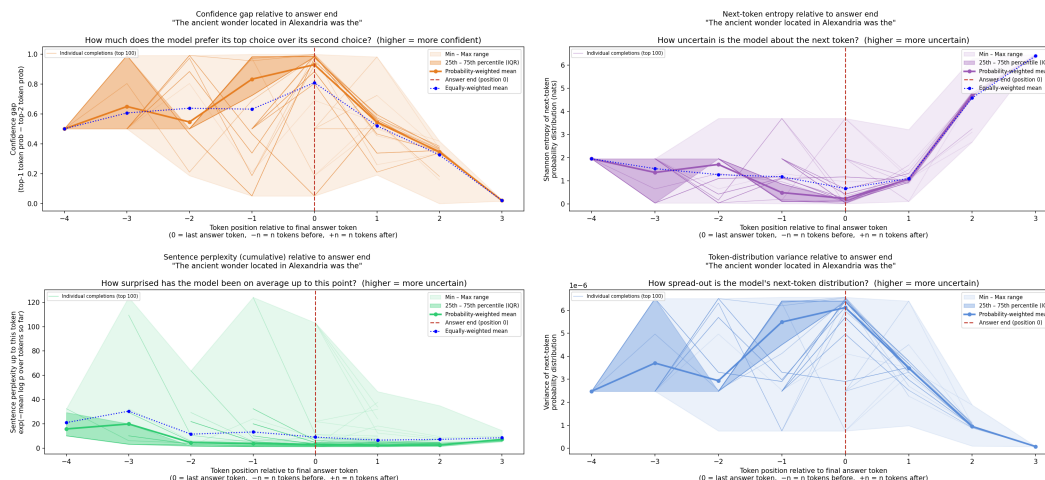
A.1.1 Example Generated Sonnet

Love is too young to know what conscience is;
Yet who knows not conscience is born of love?
Then, gentle cheater, urge not my amiss,
Against thrice purchased were my notions born
That would of love tell those days before their power!
Tull use to rule my heart's hope when I appeal;
Prove not my work says even my corruption's favor
And to correct you by direct entrust,
Hallelujah if any absence leave to thee.
Yet wit again my march, non without honor wholly mended;
No war's constant no feeble arrows nor laughed nigh died!

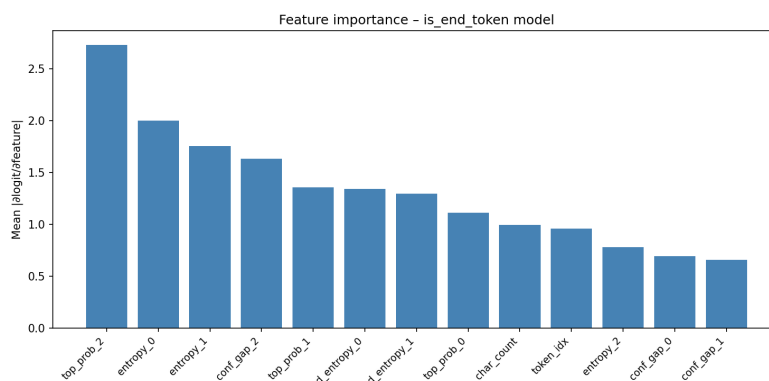
A.2 Learning Multi-Token Answer End Detection

A.2.1 Learned End Detection Investigation

The graphs for various potential answer-end signals showed large noise. (In the graphs below, just see that the individual completion lines do not show a clear pattern, but the averages do.)



It seemed reasonable to believe the per-completion signals would be very correlated, so even the combined signals would have very noisy unreliable results. However, the resulting model for predicting the end of an answer was very accurate and even improved on the text-aware spaCy model used to create it (with some human corrections).



Looking at an approximation of the feature importance above, the model appears to be combining signal from all of the features, instead of overly relying on just a few (such as character count or token count). This implies the model learned something meaningful about how to predict the end of answers.

A.3 Multi-Shot Example Correctness Effect

A.3.1 Correct, Plausible Incorrect, and Implausible Incorrect Examples

Correct Statements

1. The author of 1984 was George Orwell.
2. The capital of France is Paris.
3. The largest ocean on Earth is the Pacific Ocean.
4. ...

Plausible Incorrect Statements

1. The author of 1984 was Aldous Huxley.
2. The capital of France is Lyon.
3. The largest ocean on Earth is the Atlantic Ocean.
4. ...

Implausible Incorrect Statements

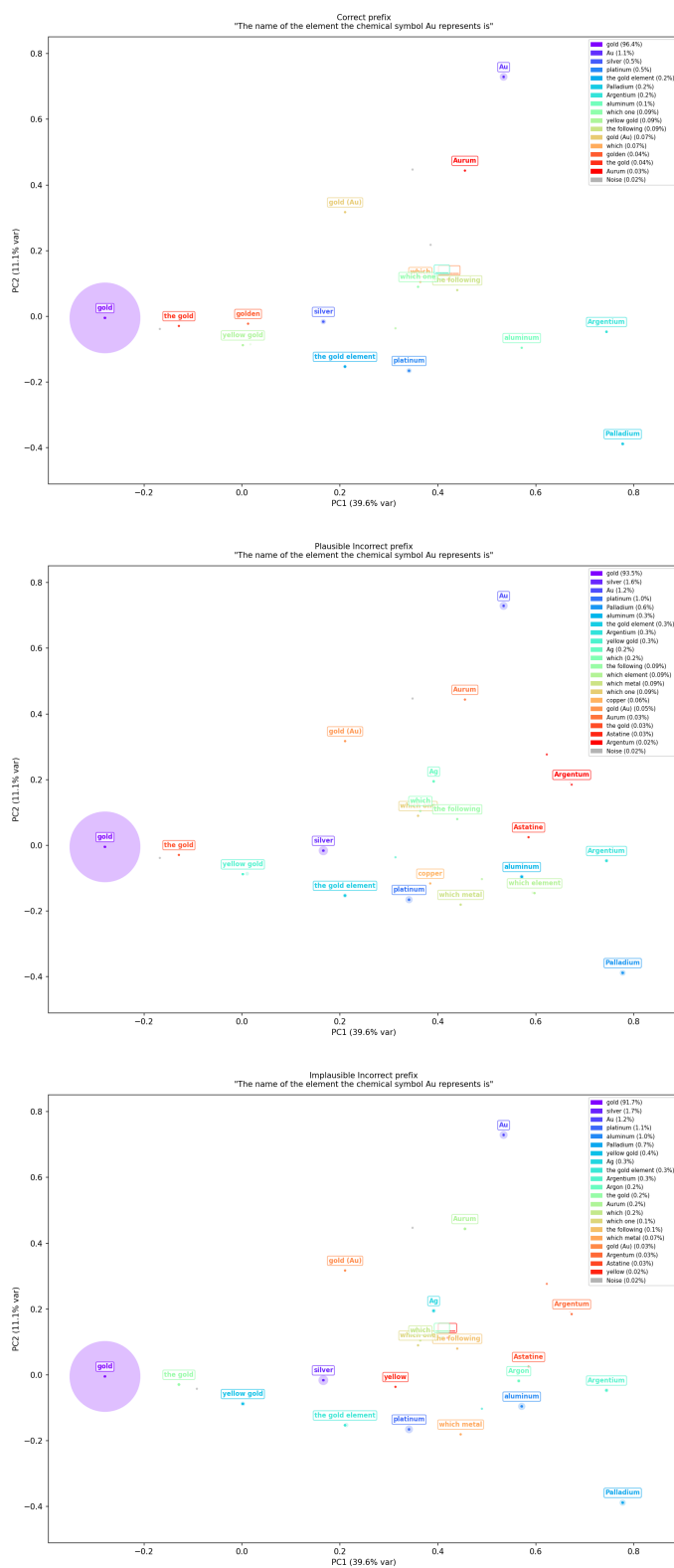
1. The author of 1984 was Dr. Seuss.
2. The capital of France is Marseille.
3. The largest ocean on Earth is the Arctic Ocean.
4. ...

A.3.2 Cloze-Style Questions For Multi-Shot Exploration

Only the first five of 20 cloze-style questions created were explored due to time constraints on labeling. The five questions are below:

1. The name of the first woman to win a Nobel Prize was
2. The art of Japanese paper folding is called
3. The name of the element the chemical symbol Au represents is
4. The organ in the human body responsible for filtering blood is the
5. The particle that carries a negative electric charge is the
6. ...

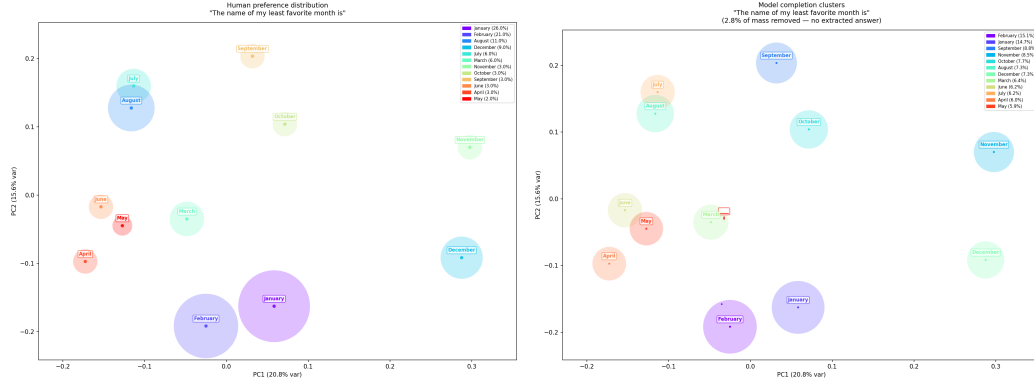
A.3.3 Cluster Graphs For Multi-Shot Exploration



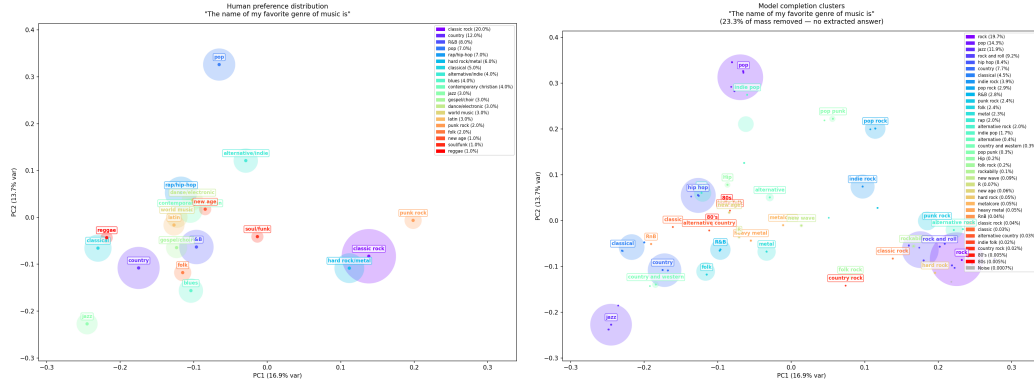
A.4 Model Human Preference Alignment

A.4.1 Human Alignment Cluster Graphs

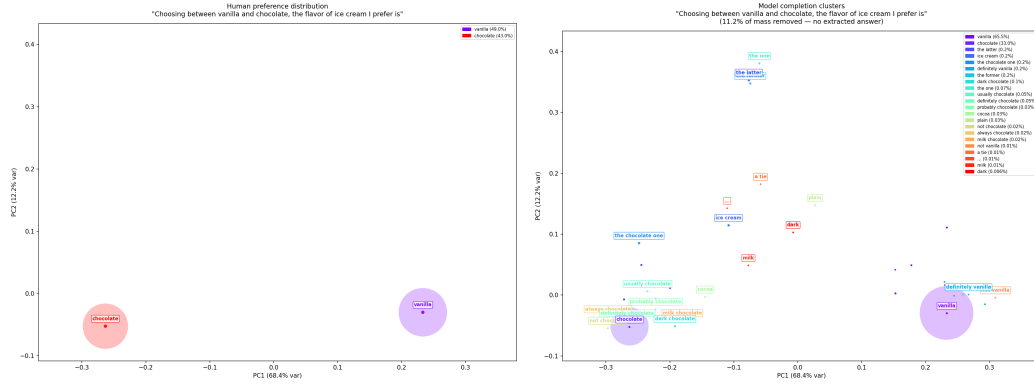
Human vs Model Least Favorite Month



Human vs Model Favorite Music Genre



Human vs Model Chocolate vs Vanilla Preference



A.5 Full Contributions

Ben:

- Sonnet generation, paraphrase detection, and classifier code
- Paraphrase task modal training code
- Alternative attention modal training code
- All the code related to multi-token entity prediction
- Dataset for learning answer boundaries and labeling for multi-token prediction work
- Running modal training for alternative attention mechanisms and other tasks
- Revising Abstract, Introduction, and Conclusion
- Making graphs for “Alternative Attention Mechanisms for Training”
- Writing “Experiments and Approaches” intro
- Writing “Experiment 0: Default Project Tasks”
- Writing “Experiment 3: Learning Multi-Token Answer End Detection”
- Writing “Experiment 4: Multi-Shot Example Correctness Effect”
- Writing “Experiment 5: Model Human Preference Alignment”

Jeffrey:

- Base GPT-2 and attention code
- Improvements to sonnet generation evaluation
- Code for parameter-efficient fine-tuning with Low-Rank Adaptation
- LoRA hyperparameter search
- Training runs for sonnet generation (including search over LoRA ranks and model sizes)
- Large training runs for paraphrase detection
- Runs and submissions to gradescope leaderboards
- Writing “Experiment 1: Parameter-efficient Fine Tuning with LoRA”

Bhavya:

- Implemented the Adam optimizer
- Implemented Flash Attention and sliding window attention
- Wrote modal training scripts for testing alternative attention mechanisms on the paraphrase detection task
- Wrote the Abstract, Introduction, Related Work, Experiment 2, and Conclusion sections for the final project report
- Wrote the research paper summary for the project proposal