ChuckleBot: Training and Fine-Tuning Large Language Models with LoRA and DPO to Generate Humorous Jokes

Anju Santhosh Kumar
asanthoshkum@umass.edu

Rigved Manoj
rmanoj@umass.edu

Mitali Juvekar
mjuvekar@umass.edu

Mehak Virender Nargotra
mnargotra@umass.edu

Preston Yee
pyee@umass.edu

1 Problem Statement

Our project is about building a large language model (LLM) that will produce jokes given a prompt. We thought it would be fun to see if LLMs can have a good sense of humor. We tried running a couple of existing LMs and AIs, such as ChatGPT, to see how well can it make jokes. However, we found that these models have limitations in the types of jokes they can generate.

ChatGPT is able to make very basic jokes if given a generic prompt to make one. However, when applied some constraints, such as making a joke based on multiple topics, it does not perform as well. Sometimes, it is successful in making them, but other times, it does not. We also observed that it is only good at making question-and-answer based, one-liner jokes. It does not expand or share other formats of jokes, such as knock-knock or standup. For example, when inputting a prompt "make a joke about computer science students, airplane, tables, purses, pants, and water bottles.", which contains numerous constraints, ChatGPT generates a very weak joke.

Our ambition is to expand farther than that, and see if we can generate more diverse set of text-based jokes, like knock-knock and stand-up, with the ability to handle more constraints. The potential significance of this project is to provide people with a resource to create theme-based, funny write ups with complex range of topics and subjects. Possible uses for this model would be to share funny jokes with your friends and family at parties, as well as help with writer's block to generate some inspirations or ideas for those who perform in stand-up routines or skits.

2 Proposed vs. Accomplished

- Collect and preprocess joke dataset from Kaggle and GitHub Repo (DONE)
- Build and train 3-5 different models on collected dataset (DONE)
  - Mistral, Llama2, Llama3, Gemma
  - Reason: Wanted to run Unsloth LoRA fine-tuning, and these models are supported by Unsloth.
- Run normal fine-tuning; Generate multiple answers on ranking-based output (CHANGED)
  - Now: Perform LoRA and LoRA+DPO fine-tuning; Generate one answer.
  - Reason: Generate one answer per model baseline and fine-tuning method per prompt for simpler comparison. Normal fine-tuning is very inefficient, and LoRA is much faster.
- Add Offensive Filter (DONE)
  - Used alt-profanity-check instead of profanity-check because original library was broken.
- Compare LLM-based and Human-based Evaluations with Cohen's Kappa (DONE + ADDED)
  - Added: Generate different types of measurements (Prompt Relevance, Funniness, and Grammar)
  - Reason: Needed more quantitative evaluation measurements to compare model performance.

3 Related Work

In the domain of computational humor, diverse methods have been explored in both joke identification and generation. Previous research on joke identification has utilized statistical analysis, N-gram analysis, Regression Trees, Word2Vec combined with K-NN Human Centric Features, and Convolutional Neural Networks (Taylor and Mazlack, 2004; Purandare and Litman, 2006; Yang et al., 2015; Chen and Soo, 2018; Mihalcea and Strapparava, 2005). These approaches have been applied in various contexts, including analyzing audience laughter in conversation transcripts and utilizing datasets such as Pun of the Day, 16000 One-liners, and even Ted Talks (Yang et al., 2015; Chen and Soo, 2018; Mihalcea and Strapparava, 2005).

At the same time, computational joke generation has emerged as a challenging problem in AI and NLP, raising the need for innovative solutions, particularly in data collection. Some researchers have curated datasets from sources such as late-night show monologues, including compilations from Conan O'Brien, "Late Night Show with David Letterman," and "The Tonight Show with Jay Leno" (Yuetian Chen, 2023).

Among the approaches employed, Recurrent Neural Networks (RNNs) with attention mechanisms have gained prominence, enabling the intelligent creation of puns by capturing contextual nuances (He Ren, 2017; Ting-yu YEH and COX, 2018). However, challenges such as long training times and RNN size limitations have persisted, leading to truncated datasets and suboptimal results. To address these challenges, subsequent research has explored techniques like Part-of-Speech (POS) tagging, aiming to extract proper nouns from training samples and generate topic-specific jokes based on user-provided keywords (Ting-yu YEH and COX, 2018).

In humor recognition, deeper word and context dependencies are needed, as humor often relies on combinations of words rather than individual words. Language models (LMs) have served as baselines in previous humor recognition studies to capture these dependencies (Blinov et al., 2023). An experiment aims to estimate the impact of increased dataset size and construction methods, introduce a strong baseline based on a deep neural network approach, compare it with previous work, and evaluate the model's generalization abilities (Blinov et al., 2023).

The possibility of automatically recognizing humor in text by modeling it as a traditional machine learning task was explored in the paper "Technologies That Make You Smile: Adding Humor to Text-Based Applications" (Mihalcea and Strapparava, 2006). This paper considers using automatic classification techniques to distinguish between humorous and non-humorous texts and proposes methods for determining the most appropriate one-liner for a given context, as well as automatically assessing a text's effective semantic orientation to avoid inappropriate addition of humor (Mihalcea and Strapparava, 2006).

Understanding humor presents several challenges for computational systems, including dealing with ambiguity, context dependence, and cultural references. The New Yorker dataset provides a real-world test bed for evaluating systems' ability to comprehend and generate humorous content(Hessel et al., 2023). Evaluation metrics for assessing the humor and quality of generated captions, including human ratings and automated metrics based on linguistic features and semantic similarity, were introduced (Hessel et al., 2023). The results underscore the complexity of humor understanding and the need for further research in this area.

The explanation generation process involves conditioning a language model on the input question and answer pair, then generating additional text that explains the reasoning behind the model's decision (Latcinnik and Berant, 2020). This process aimed to mimic human-like explanation generation. Evaluation metrics include fluency, relevance and faithfulness to the QA model's output (Latcinnik and Berant, 2020).

4 Our Dataset

The dataset for the joke generator comprises over 231,000 jokes sourced from Kaggle and stored in a CSV file. These jokes are relatively short, ranging from 10 to 200 characters, and have been collected from various websites specializing in humorous content. They encompass a diverse range of types, including knock-knock jokes, one-liners, punchlines, and quips, making them ideal for our research goal of enhancing the language model's ability to generate a broader spectrum of humorous content. While the authors have preprocessed the dataset to ensure cleanliness, there are still some instances of inappropriate or offensive content. Rather than filtering out all inappropriate jokes, we have chosen to retain them to allow the model to learn from diverse sources, while ensuring that any profane jokes are filtered at the output stage.

We generated 3 different datasets for our task.

1. **Fine-tuning dataset:** Comprising 5000 data points (prompt-joke pairs) used for fine-tuning each model using LoRA.

2. **DPO Dataset:** Comprising 500 data points used for training the LoRA-finetuned model using the Direct Parameter Optimization to align the model further towards human-like jokes.

3. **Test Dataset:** Comprising 100 data points used to test the performance of each model, where the prompt of each joke is given to the model to generate a joke.

4.1 Data Preprocessing

The original dataset 'ShortJokes' has over 200,000 jokes. To create our training dataset, these jokes were shuffled and the first 5000 jokes were selected. These jokes were then preprocessed to remove any non-ascii values that were present in them. Non-ascii characters are removed to ensure that there are no "noises" or complexity that will hinder the model to understand and learn jokes effectively.

A similar approach was followed to generate 500 jokes for the DPO dataset and 100 jokes for the test dataset. The jokes were chosen to ensure that there was no overlap between any of the datasets.

No other data preprocessing was done to ensure that the jokes do not lose any syntax or semantics that make it funny.

4.2 Data Annotation

To enable the model to learn to create jokes relevant to the provided setup, it is much more effective to train the model on prompt-joke pairs instead of jokes alone. However, the shortjokes dataset is a collection of jokes and does not include any associated prompts. To address this challenge, few-shot prompting was used to generate prompts for each joke in the training, DPO, and test datasets using an LLM. Varied prompts were used in few-shot examples to ensure comprehensive coverage of joke scenarios, enhancing the model's versatility in generating humor from different prompts.

Here is a sample of the prompt-joke pair that needs to be produced:
Joke: "What do you call a potato in space? Spudnik"
Prompt: "Give me a space-themed joke about a potato"

Prompting:
The prompts were generated by "few-shot prompting" the **'mistralai/Mixtral-8x7B-Instruct-v0.1'** model provided by Together AI. The exact prompt given to the model is as below:

```
prompt = """
    INSTRUCTIONS: Given below is a list
        of jokes along with their
        prompts. Generate a similar
        prompt for the joke I gave. Only
         give me the prompt, do not have
         'Prompt: ' in it.

    Joke: [me narrating a documentary
        about narrators] I can't hear
        what they're saying cuz I'm
        talking
    Prompt: Tell me a joke about a
        documentary.

    Joke: Telling my daughter garlic is
        good for you. Good immune system
         and keeps pests away.Ticks,
        mosquitos, vampires... men.
    Prompt: Give me a funny quip about
        parents and kids.

    Joke: If I could have dinner with
        anyone, dead or alive... ...I
        would choose alive.
    Prompt: Can you craft a joke about
        dinner?

    Joke: 'Thought I saw a walking
        burrito but it was just a pug in
```

```
    a raincoat.'
Prompt: Make me laugh with a joke
    about a pug.

Joke: {}
Prompt: {}

""".format(joke, '')
```

This strategy allows for a comprehensive coverage of joke scenarios which will enhance the model's versatility in generating humor from different prompts.

5 Baselines

We evaluated four pretrained models for our task: Gemma, Mistral, Llama2, and Llama3. These models were chosen based on their availability on the Unsloth platform and their suitability for fine-tuning using advanced techniques like LoRA and DPO. While GPT-2 was initially considered, it was excluded due to significant computational challenges.

1. **Gemma:** Gemma, part of a state-of-the-art LLM family released by Google, includes several variants. For our task, we used Gemma 7B and Gemma Instruct 7B. Architecture: Gemma 7B, despite its name, has 9.3 billion parameters (8.5 billion with weight tying). It employs multi-query attention and rotary positional embeddings for efficient processing. The model uses RMSNorm for normalization, applying it to both inputs and outputs of transformer sub-layers. Additionally, Gemma uses GeGLU activations, which enhances performance by allowing gradient propagation for negative input values. This architecture ensures efficient memory usage and faster inference. We chose Gemma because of its efficient resource usage, which made it suitable for our computational setup.

2. **Mistral:** Mistral, introduced by the Mistral AI team, is highly regarded for its powerful performance relative to its size. We chose this model for its efficiency and capability to handle large context lengths. Architecture: Mistral-7B is a decoder-only Transformer featuring Sliding Window Attention (SWA) trained with an 8k context length and a theoretical attention span of 128K tokens. SWA extends attention beyond a fixed window size, achieving an effective attention span of approximately 131K tokens with modifications in FlashAttention and xFormers, enhancing speed by 2x. The model employs a Rolling Buffer Cache to limit cache size, optimizing memory usage by overwriting older values. For efficient sequence generation, it uses pre-fill and chunking techniques to manage long prompts. Additionally, Mistral incorporates Grouped-Query Attention (GQA), balancing the high quality of Multi-Head Attention (MHA) and the efficiency of Multi-Query Attention (MQA) by partitioning query heads into groups sharing a single key and value head.

3. **Llama2:** Llama2, developed by Meta, incorporates new advancements over its predecessor, Llama, making it comparable to GPT-3.5 in performance. Architecture: LLaMA-2 shares similarities with previously mentioned models, such as the use of Grouped Query Attention (GQA) and Rotary Positional Embeddings (RoPE), Root Mean Square Layer Normalization (RMSNorm). The model, also, uses the SwiGLU activation function, offering improved performance through its combination of Swish and linear transformations. Additionally, LLaMA-2 is trained with a longer context length of 4K tokens, compared to Mistral's training with an 8K context length and sliding window attention. We chose this model to see if it's pretraining model worked better than the first two and if we would be able to fine tune it more precisely.

4. **Llama3:** Llama3 is the latest iteration from Meta, offering further optimizations over Llama2. Architecture: LLaMA3 shares similarities with Llama2 except when it comes to the training infrastructure and data. LLaMA-3 uses a 128K-token vocabulary for efficient language encoding and is pretrained on 15 trillion tokens, including 5 percent high-quality non-English data across 30 languages. The training dataset is significantly larger

than LLaMA-2's, with extensive filtering for quality.

GPT-2 was excluded from our baseline models due to its high computational demands, which strained our system even with additional resources. Training sessions were time-consuming and frequently timed out, leading to inefficiencies and making it impractical for our project.

Implementation: We utilized models from the Unsloth library, employing FastLanguageModel for efficient loading and generation. The models were loaded with specific hyperparameters: a maximum sequence length of 2048, automatic dtype detection, and 4-bit quantization to reduce memory usage. Four models (Mistral, Llama2, Llama3, and Gemma) were used. The test dataset, consisting of 100 prompts, was processed by tokenizing inputs and generating jokes using the pretrained models. The generated jokes were saved to CSV files for each model, enabling us to evaluate the performance of these models without any fine-tuning.

6 Our Approach

Generating funny jokes is a challenging task. To create humor, a model needs to understand language nuances, cultural references, and the context surrounding the joke. Due to these complexities, we thought that smaller models would struggle to generate genuinely funny content effectively. We briefly experimented with the GPT-2 small model with 117M training parameters and found that fine-tuning did not improve the performance on the task at all. Hence, we begin with our baseline models Mistral-instruct, Gemma-instruct, Llama2 and Llama3 which are large models with around 7B params.

These models are pretrained on vast datasets and have already been fine-tuned for specific tasks. They excel in tasks such as understanding instructions and generating relevant continuations of user input. While they perform text generation well, only a subset of them performs instructions adequately.

To improve the performance of our models in the specific task of joke generation, we fine-tune them using a dataset consisting of user input paired with jokes. By fine-tuning on a dataset of around 5000 examples, we aim to enhance the model's ability to generate short, humorous content relevant to given prompts.

Despite fine-tuning, large language models (LLMs) often still fall short of meeting our expectations for joke generation. To address this, we employ Differential Privacy Optimization (DPO). In this approach, we train the model on the DPO dataset that has the human-writted jokes as the "chosen" examples and jokes generated by the LLM as the "rejected" ones. This strategy helps guide the model towards producing jokes that are more similar to those crafted by humans.

We expect our LoRA-DPO trained models to perform better than the baselines, as the current baselines tend to produce long outputs that include the jokes (instruction-tuned models) or merely non-humourous content related to the prompt (other models). Fine-tuning on the downstream task of joke generation should help the model learn the structure of jokes and produce shorter, more humorous responses that are more aligned with the human-writted jokes.

We performed training and inference using colab notebooks and colab pro. One hack we used for training our models is to use the L4 GPU for training instead of T4 and A100 as it consumed lesser FLOPS that A100 but was considerably faster than T4. We used T4 for inference as it required lesser compute. In this project, we were able to generate user prompts and create our training and test datasets, fine-tune 4 models using LoRA, train them on DPO, evaluate the results and implement profanity filter to ensure that the jokes generated were not profane/offensive. The following is the process we followed to train all our models using LoRA and DPO and generate results at every point.

6.1 Hyperparameter tuning procedure

We used trial and error method to conduct hyperparameter tuning for all models as evaluation of the jokes generated by the models is a manual and subjective task. Given the time consuming nature of this evaluation, we did not explore all the hyperparameters in their entire range. Instead, our approach for setting hyperparameter values was to improve any obvious problems we noticed in the loss of each training step or in the evaluation of results. We iteratively refined the hyperparameters based on these observations. The following is the list of hyperparameters we tuned in the LoRA and DPO training process for all models:

- Rank of matrix in LoRA: Chosen = 16. We tested the training on two values of r, 16 and 32 on gemma. We did not find any significant improvements in the results generated with 32.

- Batch size in LoRA and DPO: Chosen = 10 in LoRA and 5 in DPO. We trained the models with values between 2 and 15. We chose the value that gave the most consistent training loss improvement over the training steps.

- Gradient accumulation steps: Chosen = 20 for LoRA and 10 for DPO. Batch size and gradient accumulation steps were tuned together based on consistency in training loss improvement.

- Number of training epochs: Chosen = 2 for LoRA and DPO. We trained the models using epochs ranging from 1 to 5 (25 to 125 training steps) and chose 2 based on a threshold improvement in training loss as well as good enough evaluation results on the jokes produced after training.

- Learning rate: Chosen = 2e-5 for LoRA and 2e-7 for DPO. Learning rate for LoRA was set to the default value used by unsloth which gave a good enough performance. Learning rate for DPO was lowered from 2e-5 to 1e-7 in a few steps to mitigate the exploding gradients problem observed with llama2. The learning rate was also lowered based on recommendations in literature for optimal DPO training.

- Max gradient norm: Chosen = 3 for DPO. Values varying from 0.5-4 were tried to mitigate the exploding gradients problem observed with llama2.

6.2 Fine-tuning Pretrained Models

After collecting our jokes dataset, we move forward with implementing our training models, and LoRA/DPO fine-tuning. We wanted to take advantage of the Unsloth library for LoRA fine-tuning, which has a very nice and easy-to-follow implementation of how to train and fine-tune models with LoRA. We started by using a copy of a sample Colab notebook on training Llama3-8b with LoRA that was freely provided from Unsloth's GitHub repository. This notebook served as a template for how we would implement our code.

We modified the preprocessing data section to take in our joke dataset, and modified the training section to train Llama3 and save the pretrained model. We also modified the tokenizer sections of the code so that we can feed in our generated joke prompts from our test set to generate our jokes and export the results as a csv file. Once we were able to successfully run Llama3 end-to-end with our dataset, we then replaced Llama3 with our other models and testran them to ensure that they would run as well. The models we chose were Mistral-7b-instruct, Llama2-7b, Llama3-8b, and Gemma-7b-instruct from Unsloth. We chose the instruct versions of Gemma and Mistral because we wanted to see if they would follow the prompts better than non-instruct. With the models successfully able to run, we implement a for loop to run multiple models per run to save time.
Libraries used: Pytorch, unsloth, pandas, datasets, numba, numpy, trl, transformers and peft.

Our training strategy goes as follows:

- Create a dictionary of Unsloth models with their string names. For each name, we extract their Unsloth model and tokenizer, and add LoRA to the model.

- We extract the preprocessed joke training dataset.

- We create TrainingArgument classes that contain fine-tuned parameters. Each model has their own TrainingArgument, with their own unique parameter values for their own purpose. The fine-tuning parameters we primarily used were learning rate, batch size, gradient accumulation steps, and training epochs.

- We train a Supervised Fine-Tuning Trainer (SFT Trainer) that takes in the model, tokenizer and dataset.

- Once the training is complete, we save the model and tokenizer locally.

6.3 Generate DPO Training Dataset and LoRA-Only Test Results

Once our trained models have been saved, we now look into generating jokes with those models using our test dataset and DPO dataset.

For our test set, we want to generate joke results to see how well the jokes are made after running LoRA fine-tuning alone. For our DPO dataset, we want to run them the same way as we run our test set. The only difference is that our test set is the results we will compare to our pretrained baseline, while the DPO set is used to create instructional training data to run DPO on the models. For DPO, the results generated from this model will be classified as the "losing" joke in the DPO training dataset while the human-written joke is the "winning" joke.
Libraries used: unsloth, Alt-profanity-check, pandas, datasets, pytorch, datetime, tqdm

For generating our jokes for each prompt, our strategy is as follows:

- For LoRA-only results, we first extract the test set.
- We extract every saved model and tokenizer sequentially.
- For each tokenizer, we input the prompt. We then feed the tokenizer into their corresponding model to generate a joke.
- When the joke is being generated, it is inside a while loop and being checked if the level of profanity is above a threshold. We found that the threshold limit that keeps our jokes appropriate is 0.5.
- If the joke is above the threshold, we try to generate another joke for the same prompt and check again. If the joke is below or equal to the threshold, we use it.
- Note that we also have a count limit, as to prevent being stuck in the loop. The generator has 3 attempts to try. If it is unable to find a reasonably appropriate joke within the number of attempts, we will then just accept the last joke generated and continue.
- Once the jokes are generated, they are then exported as a csv file.
- The same process is used for generating DPO training data.

6.4 Training using DPO and generating results

After fine-tuning the pre-trained models on the 5000 jokes dataset, we proceed with training the fine-tuned model with DPO.

We use the template provided by the Unsloth library for DPO training as our starting point to perform DPO training on our fine-tuned models. We added code to read and map prompts and jokes from the .csv file to the required format for DPO training - prompt, chosen and rejected. Here the prompt is the joke's prompt, chosen is the original human joke and rejected is the joke generated by the fine-tuned model.
Libraries used: pytorch, unsloth, pandas, os, datasets, transformers, trl, numba, datetime, Alt-profanity-check.

Training:

- We load the fine-tuned model and tokenizer using sloth's FastLanguageModel.
- We use the DPOTrainer module by HuggingFace''s TRL to create a training instance for DPO and provide the model and hyper-parameter values.
- We perform training for the set number of steps. After training is complete the trained model and tokenizer are saved locally.

Generating results using DPO trained model:

- We load the DPO trained model and tokenizer using sloth's FastLanguageModel.
- We load the test dataset of 100 jokes and generate joke for each prompt in the test dataset.

- We also implemented the profanity filter provided by alt-profanity-check library such that the model regenerates the joke if the profanity score is above a threshold of 0.5 for a maximum of 3 attempts. The joke produced after 3 attempts is chosen if all 3 attempts yielded profane jokes.

- We process the jokes generated by the model to remove any unwanted tokens like EOS tokens and save the results locally.

6.5 Evaluation

The jokes generated by the pretrained, finetuned and DPO version of each model are evaluated on two different evaluation metrics **"Performance Score"** and **"Rank Score"**.

6.5.1 Performance Score

The objective of this project is to train LLMs to create short funny jokes that match the given prompt. To analyse these aspects, 4 metrics are considered.

- **Prompt Score:** This metric evaluates how well the generated joke matches the given prompt. The score scale is from 0 to 2 where 0 is no match, 1 is imperfect match and 2 is perfect match.

- **Funny Score:** This metric evaluates how funny the generated joke is. The score scale is from 0 to 2 where 0 is not funny, 1 is somewhat funny and 2 is funny.

- **Grammar Score:** This metric evaluates if the generated joke is grammatically correct. The score is 1 if grammatically correct, 0 otherwise.

- **Length Score:** This metric evaluates if the joke is short or long. The score is 1 if the length of joke is below 100 characters, 0 otherwise.

These 4 metrics were considered because the ideal joke is a funny short sentence that matches the prompt perfectly. The scores are also weighted more towards "Prompt Match" and "Funniness" as they are the more critical metrics. A **Total Score** which is the sum of all 4 scores is computed.

Performance Score is computed as the ratio of sum of total score by maximum score possible.

$$PerformanceScore = \Sigma \frac{TotalScore}{MaxScore}.$$

Performance Score was calculated by both Humans and LLMs as these metrics are often subjective, and would differ from person to LLM. An agreement score between Human generated and LLM generated performance score is also computed with Cohen's Kappa.

Prompting:

The LLM Performance Score (Prompt Score and Grammar Score) was generated by "few-shot prompting" the **'meta-llama/Llama-3-70b-chat-hf'** model provided by Together AI. The examples given through "few-shot prompting" are varied and cover all scenarios.

HuggingFace Model:

The LLM Funny Score generated through 'few-shot prompting' was over-inflated categorizing almost all generated jokes as "funny". Hence, a different approach was utilized. The score was computed through humor-no-humor model in huggingface which boasts an F1 Score of 0.95.

6.5.2 Rank Score

Another way to evaluate the performance of a model is to directly compare jokes from the pretrained model and the SFT model to determine which jokes are funnier. To do this, A combined dataset for each model was created that contained the prompt, pretrained joke, finetuned joke and DPO joke. For a particular prompt, jokes were chosen in pairs and evaluated by an LLM to determine which joke was better.

When Model A is compared to Model B, Scoring is as follows:

- A score of +1 is assigned if joke generated by Model A is funnier.

- A score of -1 is assigned if joke generated by Model B is funnier.

- A score of 0 is assigned if both jokes are equally funny/unfunny.

Prompting:

The LLM Rank Score was generated by "few-shot prompting" the **'meta-llama/Llama-3-70b-chat-hf'** model provided by Together AI. The exact prompt given to the model is as below:

```
Instruction = """INSTRUCTIONS: Given a
    prompt and 2 jokes (PreJoke and
    FineJoke), return the better joke.
    return
    only the label of the better joke.
        Do not give explanation.

    Prompt: Give me a funny parent-child
         joke?
    PreJoke: Telling my daughter garlic
        is good for you. Good immune
        system and keeps pests away.
        Ticks, mosquitos, vampires...
        men.
    FineJoke: Shush you! I gave birth to
         you.
    Answer: PreJoke

    Prompt: Tell me a joke about a
        documentary?
    PreJoke: narrators are narrating
        storytellers.
    FineJoke: [me narrating a
        documentary about narrators] I
        can't hear what they're saying
        cuz I'm talking
    Answer: FineJoke

    Prompt: Can you come up with a joke
        about dolphins?
    PreJoke: What do you call a dolphin
        who is sick? A sick dolphin.
    FineJoke: Do you know about those
        people who take pictures of
        dolphins that look like naked
        people? Well, they're called
        whale-fashion photographers.
    Answer: FineJoke

    Prompt: Make me laugh with a joke
        about the DMV.
    PreJoke:  I think the D.M.V. makes
        us all feel like that little boy
         from the movie "A Nightmare on
        Elm Street". Every time you go
        there, you think... GGggggh!
        Wake me up! Wake me up!!!
    FineJoke: What do you call a person
        who has been waiting in line at
        the DMV for 10 years?
    Answer: PreJoke

    Prompt: {}
    PreJoke: {}
    FineJoke: {}
    Answer: {}

    """.format(prompt, preJoke, fineJoke
        , '')
```

This approach of comparing jokes gave comparatively better results compared to scoring the funniness of a joke.

7 Results

7.1 LLM vs. Human: Performance Scores

Model	Type	LLM	Human	Agreement Scores
-	Human	82.0	95.2	0.82
Mistral	Pretrained	72.2	60.8	0.63
	LoRA	59.8	57.7	0.56
	DPO	**86.0**	**75.2**	0.71
Gemma	Pretrained	**71.5**	51.8	0.68
	LoRA	69.7	**67.3**	0.69
	DPO	65.5	66.6	0.74
Llama2	Pretrained	54	30.3	0.59
	LoRA	60	48.8	0.63
	DPO	**84.6**	**63.5**	0.70
Llama3	Pretrained	**71.3**	44.3	0.50
	LoRA	67	56.1	0.67
	DPO	66.8	**66.7**	0.73

Table 1: Performance scores by LLM and Humans for 4 models

Below is the analysis of performance scores shown in table 1:

- The performance scores between LLM and Humans have a low agreement. Percentage agreement score is around 0.65, but Cohen's Kappa is low, around 0.2.
- The scores are also varied between Models, ranging from values of 30 to 92.
- Even with these limitations, both the LLM scores and Human scores appear to have almost the same agreement on the ranking of models.
- From table 3, It can be seen that Mistral DPO is rated as the best model by both the LLM and Human scores.
- Similarly the ranking of pretrained, LoRA and DPO is almost consistent for both LLM and Human.
- Even though the actual performance score seems to be inconsistent, the overall ranking is consistent. It also matches with the qualitative analysis conducted.
- The DPO model is scored the highest in all cases except Gemma.

Model	SFT/Pretrain	DPO/Pretrain	DPO/SFT
Gemma	-11	-8	69
Llama2	51	90	59
Llama3	75	85	67
Mistral	8	42	74

Table 2: Rank scores across different steps in the training process

7.2 Rank Score

Below is the analysis of rank scores shown in table 2:

- Comparing jokes and choosing the best joke seems to work well as a evaluation metric compared to assigning scores.
- The values of each model is also consistent with the manual analysis conducted.
- Llama2 and Llama3 trained models outperform the pretrained models consistently as shown by the high rank scores.
- Similarly Mistral DPO also outperforms both the pretrained and the SFT versions.
- Except for gemma, the other 3 models show that pretrained < finetuned < DPO.

7.3 Qualitative Observations

In observation to our generated jokes, we noticed there are a number of patterns in the text based on the configuration between the LLM and fine-tuning method used. In our three prompt examples shown in Tables 3, 4 and 5, we see that the pretrained models generate the largest amount of text, which is expected since they have not been trained to create short jokes, especially among Mistral, Llama2 and Llama3. Gemma seemed to be the only model that remained having a minimized amount of text. In the pretrained models, we also notice that Llama2 seems to be generating the worst performance in comparison to the other three models, responding with either a definition of the topic instead of a joke in Table 3, emojis relating to topic in Table 4, or repeated patterns of a certain text pattern but no joke in Table 5. We did, however, noticed that Mistral and Gemma appear to be generating some form of jokes for all three prompts because of their instruction tuning. Llama3 is a bit of a mix bag of sometimes making a joke in Table 5, but other times not so much in Tables 3 and 4.

For the fine-tuning on just LoRA, we immediately see a difference in the change in joke responses. Mistral, Llama2 and Llama3 have all significantly reduced their texts to become simple and concise. However, in their shortened texts, the jokes seem to not be that consistent in improvement. For Llama2, we noticed that LoRA has improved it's results in Table 4 to make an actual pun, but not improve in the other two tables. For Mistral, it seems that LoRA has made its response shorter, but worse, leave text that doesn't resemble much of a joke anymore compared to pretrained. Suprisingly, we noticed that Gemma performed worse when trained on LoRA in generating jokes. It wasn't able to improve on or maintain the jokes as well as before. In tables 3 and 5, we see that Gemma is repeating text, while Table 4 is still able to make jokes, but not as well as before.

Once we added DPO with the LoRA finetuned model, we noticed that Mistral has improved quite significantly, seeing that the generated text response is now in the format of quick pun or joke, in comparison to LoRA-only and pretraining. Llama2 has also shown significant improvements in the jokes it produces, with some being as funny as human jokes. Llama3 has progressively gotten its response shorter, with some texts being puns that are roughly around the same level of humor as the texts from LoRA-only, particularly in Tables 3 and 4, but some improvement in Table 5. Gemma with DPO has appeared to improved and return back to roughly similar level of humor as its pretrained responses with the same length of text in Tables 3 and 5, but seems to be getting worse than LoRA-only in 4.

From the various observations in these three prompts, along with many other prompts from our test set, we notice a considerable improvement in making shorter one-liner jokes with just LoRA fine-tuning. We notice that the models were able to widen their results beyond the usual joke format of question and answers to punchlines, one-liners, puns, observations jokes etc. These observations are consistent across all models. After DPO tuning, we observe a significant improvement in the funniness of jokes produced with some jokes actually at par with human written ones models. An interesting feature we noticed was that after DPO, a lot of the outputs were stringing together words that could work together as jokes, ie. were almost jokes but missed some elements that would make them funny.

8 Error Analysis

Human Evaluation was done on 100 examples per training per model (over 1000 examples overall). The analysis is given below:

Pretrained Models (Baseline):

- They fail at giving shorter jokes: It usually gives a long sentence, while both SFT and DPO give short jokes.
- Llama2 and Llama3 pretrained models are not instruction tuned: they just repeat the pattern of the given prompt format. Mistral and Gemma are instruction tuned and hence do not face this issue.
- The baseline models rarely create jokes, and even when they do, it is of the same format of "question,joke". Eg: Why did the procrastinator cross the road? To get to the other deadline.

SFT and DPO Models (Trained):

- The models create much shorter sentences and jokes.
- All the models now follow instructions.
- The models have somewhat learnt to diverge from the "question, joke" format after being trained.
- The DPO trained models generate some funny jokes. Mistral in particular creates funny jokes for most prompts.
- The DPO models have learnt some semantics and syntax of creating jokes even if the resulting sentence is not funny.
- The models revert to the "question, joke" format when encountering some prompts.
- The models have learnt to create diverse formats of jokes but the resulting combination of words are not actually funny. This is because understanding humor is a very difficult and subjective task.
- Gemma Model has not improved much even after SFT and DPO. This could be because the pretrained model itself is capable of creating diverse jokes. It requires a larger variety of data along with more training to create better jokes.

We can conclude that SFT on the short jokes dataset along with DPO helps the model create sentences that are almost funny. A diverse dataset with more training could help create funnier jokes for any prompts.

9 Contributions of Group Members

List what each member of the group contributed to this project here. For example:

- **Anju:** Refined prompting for generating varied test dataset prompts and generated test and DPO dataset. Performed hyperparameter tuning for LoRA and DPO training by trial and error method. Wrote the code for DPO training and generating results. Performed DPO training for all 4 models and generated results on test dataset. Manually evaluated results for llama2 model. Wrote parts of Our Dataset, Our Approach and Error analysis.
- **Rigved:** Designed few-shot prompts for generating "jokePrompts" for training, DPO and test dataset. Designed the evaluation metrics and developed code to perform LLM evaluations. Tested different hyperparameters for training Gemma with SFT+LoRA. Manually evaluated Gemma results for Human evaluation. Wrote "Evaluation", "Results", "Conclusion" and part of "Error Analysis" sections of report.
- **Mitali:** Generated GPT-2 pretrained model results for a small part of the dataset, finetuned LlaMa2 using QLora on a default dataset, and wrote "Baseline" section of the project report. Additionally, did human evaluation for all the LlaMa3 generated jokes.

- **Mehak:** Fine-tuned a GPT-2 model to generate jokes and wrote "Related Works" and some other small parts of the project report. Additionally, conducted hyperparameter testing for the Llama2 model and computed agreement scores between human-generated jokes and those generated by various Language Model (LLM) architectures.

- **Preston:** Built main structure of training SFT & LoRA for all four models on Colab Notebook. Wrote code to perform profanity check when generating LLM outputs. Manually evaluated Mistral results for Human evaluation. Wrote part of "Our Approach" and "Qualitative Analysis" sections of report.

10 Conclusion

Training a Large Language Model to generate funny jokes turned out to be more challenging task than initially anticipated. The idea behind the project was that fine-tuning a LLM on large dataset of jokes would enable it to generate jokes given any prompt. This intuition was partially correct as the model was able to learn some semantics and syntax of jokes, but the complexity and subjectivity of jokes made it difficult to consistently generate funny jokes.

Of the 4 models chosen, Llama2 performed the worst in generating jokes without any training. But surprisingly, the SFT+DPO trained Llama2 was able to generate funny jokes for a variety of prompts. It was able to outperform models like Gemma Instruct, which performed well without training but did not improve with SFT or DPO.

One of the key takeaways from the project was that the models were able to generate different styles of jokes that tended to be almost funny. Expanding the dataset by including a wider variety of jokes and different jokes of the same variety is one potential way to improve the model. DPO training also increased the performance of the model, Hence, applying DPO training with a larger dataset is also one avenue for future research. These methods could potentially lead to even more effective joke generation by LLMs.

11 AI Disclosure

Below is our AI Disclosure from the proposal.

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.
  - Generated Latex table code using Chatgpt 4o.
  - Created Prompts and Evaluation Scores by few shot prompting Together AI.
  - Asked Chatgpt to explain hyperparameters of unsloth training arguments.
  - Asked Chatgpt to refine sentences used in report by correcting grammar or rewriting words.

If you answered yes to the above question, please complete the following as well:

- If you used a large language model to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for each prompt, and specify which part of the proposal is associated with which prompt.
  - Prompts for Latex table Code:
    - Generate the latex table code for the table shown in this image.
  - Prompts for Prompt Generation and Evaluation Scores:
    - *specified in the dataset and evaluation sections.*
  - Prompts for explaining hyperparameters:
    - *Given a set of hyprparameters*, tell me what each hyperparameter is, in detail.
  - Prompt to rewrite sentences
    - Llama2 has also shown a significant improvements in the jokes produced where some of them were as funny as human jokes. better grammar

- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?

– The LLM (chatgpt 4o) was able to parse the image and give the exact latex code to make the table. This was extremely helpful as it is time-consuming to create tables in latex.
– The LLM (chatgpt) was able to explain a few of the hyper-parameters well, but made some incorrect interpretations for the rest of the hyper-parameters. It was helpful in giving a general idea and hence a head start on identifying the hyper-parameters.

References

Blinov, V., Bolotova-Baranova, V., and Braslavski, P. (2023). Large dataset and language model funtuning for humor recognition.

Chen, P.-Y. and Soo, V.-W. (2018). Humor recognition using deep learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*.

He Ren, Q. Y. (2017). Neural joke generation. *Stanford University*, (4):1–8.

Hessel, J., Marasovic, A., Hwang, J. D., Lee, L., Da, J., Zellers, R., Mankoff, R., and Choi, Y. (2023). Do androids laugh at electric sheep? humor "understanding" benchmarks from the new yorker caption contest. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Latcinnik, V. and Berant, J. (2020). Explaining question answering models through text generation.

Mihalcea, R. and Strapparava, C. (2005). Making computers laugh: Investigations in automatic humor recognition. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 531–538. Association for Computational Linguistics.

Mihalcea, R. and Strapparava, C. (2006). Technologies that make you smile: Adding humor to text-based applications. *IEEE Intelligent Systems*, 21:33–39.

Purandare, A. and Litman, D. (2006). Humor: Prosody analysis and automatic recognition for f*r*i*e*n*d*s*. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 208–215.

Taylor, J. M. and Mazlack, L. J. (2004). Computationally recognizing wordplay in jokes. In *Proceedings of CogSci 2004*.

Ting-yu YEH, Nicholas FONG, N. K. and COX, B. (2018). Generating original jokes. *COEN 296 NATURAL LANGUAGE PROCESSING TERM PROJECT*, (2):1–8.

Yang, D., Lavie, A., Dyer, C., and Hovy, E. (2015). Humor recognition and humor anchor extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2367–2376.

Yuetian Chen, Bowen Shi, M. S. (2023). Prompt to gpt-3: Step-by-step thinking instructions for humor generation. *Rensselaer Polytechnic Institute*, (1):1–5.

Prompt 1: Give me a joke about a job interview			
Human: [shows up 2 hours late for interview] Sorry I was trying to get out of a beanbag chair.			
LLM	**Baseline (Pretrained)**	**LoRA**	**LoRA & DPO**
Mistral	Why did the interviewer ask the candidate to bring a ladder to the job interview? The interview was for a position as a gopher in a golf course. The candidate brought a ladder anyway, thinking it was a typo and it was actually a job for a laddar. (Laddar is not a real word, it's a typo intended to be a ladder) I hope that brought a smile to your face! If you have any other requests or need more jokes, feel free to ask. Have a great day! 😊	3 Interviews. No Jobs. 3 Dogs. 14 Houses.	**Interviewer** "So why do you want this job?" **Me** "I couldn't get the other one"
Llama2	1. What is a job interview? 2. A meeting between two people where one person is trying to get the other person to hire them. 3. What do you call someone who goes to job interviews? 4. A job seeker. 5. What is the difference between a job interview and a job interview? 6. A job interview is when you go to a job interview and the other person is trying to get you to hire them. 7. What do you call someone who is trying to get you to hire them? 8. A job interviewer. 9. What is the difference between a job interview and a job interview? 10. A job interview is when you go to a job interview and the other person is trying to get you to hire them. 11. What do you call someone who is trying to get you to hire them? 12. A job interviewer	1. Interviewer: "Tell me about your biggest weakness." 2. Candidate: "I'm not a very good listener." 3. Interviewer: "Can you tell me about a time you were able to turn a negative situation into a positive one?" 4.	I went for a job interview and the interviewer asked me,"So, what is it that you do for a living?" I said,"I'm a professional photographer." He asked,"What do you photograph?" I said,"Oh, I'm a professional photographer." ### Did
Llama3	1. What do you call a job interview where you don't have to wear a suit? 2. What do you call a job interview where you don't have to wear a suit? 3. What do you call a job interview where you don't have to wear a suit? 4. What do you call a job interview where you don't have to wear a suit? 5. What do you call a job interview where you don't have to wear a suit? 6. What do you call a job interview where you don't have to wear a suit? 7. What do you call a job interview where you don't have to wear a suit? 8. What do you call a job interview where you don't have to wear a suit? 9. What do you call a job interview where you don't have to wear a suit? 10. What do you call a job interview where you don't have to wear a suit?	5. Interviewer: What would you do if you saw a bear? Me: I would run. Interviewer: And if you were the fastest man in the world? Me: I would still run. Interviewer: But if you were the fastest man in the world and you saw a bear, wouldn't you	How many programmers does it take to change a light bulb?None. That's a hardware problem.
Gemma	Why did the applicant bring a banana to his job interview? The answer: In case they got hung up.	I was a little nervous about my job interview today. I was a little nervous about my job interview today. I was a little nervous about my job interview today.	A man was applying for a job at a bookstore. When asked about the reason for his interest, he responded,"I am a bookworm."

Table 3: Generated joke for each LLM and fine-tuning method pair given prompt about job interviews

Prompt 2: Give me a joke about the water cycle			
Human: Water evaporates, condenses, precipitates, and evaporates again. It's a viscous cycle.			
LLM	**Baseline (Pretrained)**	**LoRA**	**LoRA & DPO**
Mistral	Why did the ocean break up with the cloud? The ocean felt the rain was bringing in a bad crowd. But in all seriousness, the water cycle is a fascinating process that keeps our planet's ecosystems thriving. It's important to remember that every drop of water plays a role in this continuous cycle.	3 billion years ago... Water vapour was more popular... It was the coolest thing...	I heard the water cycle got promoted to the Air Force.
Llama2	💦💦	Q: What did the ocean say to the beach? A: Nothing, the ocean has no ears.	How did the ocean say goodbye to the water cycle? "I'm not in the mood right now, I'll call you back later." ### Did you know? The water cycle is the continuous movement of water on, above and below the surface of the Earth. Water can change states among solid,
Llama3	### The Water Cycle The water cycle is the process by which water is recycled on Earth. It involves the movement of water between the atmosphere, land, and oceans. Water is constantly moving through the cycle, and it is important for life on Earth. The water cycle is driven by the sun's energy and the Earth's gravity. Water evaporates from the surface of the oceans and lakes, and it is transported through the atmosphere. When the water vapor condenses, it falls as precipitation. The water then flows back to the oceans and lakes, where the cycle starts again. The water cycle is an important part of the Earth's hydrological cycle, which is the continuous movement of water between the atmosphere, land, and oceans. The hydrological cycle is important for the maintenance of life on Earth. It provides the water that plants and animals need to survive, and it helps to regulate the Earth's temperature. Without the water cycle, the Earth would be a much drier and hotter place.	10 years ago, I was a rain drop. Now I'm a puddle.	I'm not the only one who's going through a dry spell.
Gemma	Why did the water vapor escape from the cloud? The answer is: Because it was tired of being mist. This joke is a play on words that is related to the water cycle. The joke is funny because it is unexpected and creative.	What do you call a water molecule that is in the process of evaporating? A water vapor.	A meteorologist was studying clouds in the sky, after his research he wanted to take a water droplet vacation.

Table 4: Generated joke for each LLM and fine-tuning method pair given prompt about water cycle

Prompt 3: Give me a joke about a job performance review			
Human: I always give 100% Which is why I was sacked from being an exam marker.			
LLM	**Baseline (Pretrained)**	**LoRA**	**LoRA & DPO**
Mistral	During a job performance review, the boss said to an employee,"You have done an excellent job this year. I am raising your salary by $1,000." The employee replied,"Thank you, sir. But may I ask why?" The boss thought for a moment and then replied,"Well, to be honest, I've been meaning to do it for a while now, but every time I see your face, I think of a thousand dollars." The employee smiled and said,"That's a great answer, sir. I'll make sure to keep a good face on from now on."	7. Your ability to keep the employees well informed on company happenings is a strong asset, your inability to let go of the past is a significant liability.	Boss: I would like to compliment the employee on his excellent job performance I have not seen such competency in many years. [pause] Employee: And why did you hire me 3 years ago?
Llama2	1. "I'm sorry, but I can't give you a raise. Your work is excellent, but the company's finances are such that we can't afford to pay anyone more." 2. "I'm sorry, but I can't give you a raise. Your work is excellent, but the company's finances are such that we can't afford to pay anyone more." 3. "I'm sorry, but I can't give you a raise. Your work is excellent, but the company's finances are such that we can't afford to pay anyone more." 4. "I'm sorry, but I can't give you a raise. Your work is excellent, but the company's finances are such that we can't afford to pay anyone more." 5. "I'm sorry, but I can't give you a raise. Your work is excellent, but	A man was having a performance review at work. His boss said,"Your performance has been very good, but you have a problem. You're just not a team player. You don't get along with your co-workers. So, I'm going to give you a chance to improve	I'm not sure how to tell my boss I'm quitting. I'll just pretend I'm getting a performance review. ### Did you like this joke?
Llama3	1. The boss asked the new guy, "How many times have you been married?" "Two," he replied. "First marriage to the boss's daughter, second to the boss's daughter." ###	1. I'm a great employee, I've never been late, never been sick, never called in. 2. I've never had a bad review, never been disciplined, never been written up. 3. I've never been promoted, never had a raise, never been given a bonus. 4	When the boss asked me how my performance review was, I told him it was a joke.
Gemma	Why did the boss give the employee a bad performance review? Because he was always on their case.	The office manager is doing a performance review. He's not very good at it. He's not very good at it. He's not very good at it.	Did you know that when you get fired, you get the severance pay if your manager loses weight?

Table 5: Generated joke for each LLM and fine-tuning method pair given prompt about job performance