University of Bonn
CAISA lab

Introduction to Natural Language Processing

(Winter Semester 2025/2026)

Project Report of Team 19

Group Members:

Jiadong Han	3198936
Rami Kallel	50019733
Zhuangyu Zhou	50359013
Mohammad Saeed Motevali Amin	50246276
Polad Ibrahimov	50427387

February 9, 2026

1. Introduction:

Humor is one of the most complex forms of language use, requiring creativity, cultural awareness, and the ability to establish and subvert expectations. While Natural Language Processing (NLP) has made significant progress in humor understanding, detecting whether a given text is funny or classifying joke types, humor generation remains a largely underexplored frontier.

This project addresses the challenge of automatic humor generation as defined by SemEval 2026 Task 1: **MWAHAHA (Models Write Automatic Humor And Humans Annotate)**, the first shared task dedicated to computational humor generation. Specifically, we tackle two subtasks: **Subtask A: Text-based Humor Generation** under the **Word Inclusion** constraint, where a system must generate a joke that naturally incorporates two specific words drawn from a list of rare word combinations, and **Subtask B: Headline Humor Generation**, where the system must create humorous headlines for news articles. The use of rare word pairs in Subtask A is designed to prevent systems from simply retrieving existing jokes, thereby encouraging genuine creative generation.

Our approach combines a fine-tuned Llama 3 8B model with retrieval-augmented generation and a secondary LLM-based generator, coordinated through an agentic pipeline that iteratively generates, scores, and refines candidate jokes. The central research question guiding this work is:

How effectively can a multi-model pipeline generate creative, high-quality humor that naturally incorporates arbitrary constraints, whether word pair inclusions or news headline contexts?

We hypothesize that combining specialized fine-tuning for humor with retrieval-augmented context, complementary generation from GPT-OSS-120B through carefully designed prompts that establish the model as an expert comedy writer with knowledge of multiple humor techniques, and automated quality scoring will produce creative, high-quality jokes.

2. Related Work

Early work in computational humor focused primarily on humor detection and rigid template-based generation, with classic rule-based systems such as JAPE [1] and HAHAcronym [2] producing structurally valid but narrowly scoped jokes. Neural approaches later introduced greater flexibility, including RNN- and GAN-based pun generation [3]. More recently, transformer-based systems have enabled contextual and stylistically richer humor generation, exemplified by AmbiPun [4] and hybrid symbolic–LLM systems such as Witscript 3 [5]. Prior work shows that fine-tuning with curated humor datasets can improve performance [6]. Parameter-efficient fine-tuning methods such as LoRA [7] and its quantized variant QLoRA [8] enable effective adaptation of large models under limited computational resources and have shown promise in humor-focused settings. In parallel, Retrieval-Augmented Generation (RAG)[9], though traditionally applied to factual tasks, has been explored for creative generation by retrieving stylistically similar examples; we adopt a similar approach using Maximal Marginal Relevance [10] to balance relevance and diversity. Finally, due to the subjectivity of humor, automatic evaluation remains challenging, motivating the use of LLM-based evaluators such as GPTScore [11], G-Eval [12], and humor-specific multi-dimensional scoring frameworks [13], which we follow in our evaluation setup.

3. Data Exploration and Preprocessing

This study utilizes three humor datasets for joke generation and evaluation:

rJokes Dataset (25,000 samples): Sourced from Reddit's r/jokes community (2008-2019)[1], this dataset contains jokes with user-voted quality scores (0-9). We first applied text normalization and then filtered jokes with scores ≥ 4 and word counts between 5-30 words, yielding 25,000 training examples for fine-tuning Llama 3 8B. We then extracted rare word pairs appearing five times or fewer across the corpus and formatted each example as an instruction-response pair using Llama 3's chat template. Each example prompts the model to generate jokes incorporating specific word pairs. Table 1 shows the score statistics, demonstrating a concentration in the mid-to-high quality range.

Table 1: Comparative statistics across the three humor datasets

Metric	**rJokes**	**Short Jokes**	**HaHackathon**
Dataset Information			
Total Entries	25,000	20,000	8,000
Rating Scale	0–9 (discrete)	–	0–4 (continuous)
Mean Rating	[5.15]	–	[2.26]
Std Rating	[1.46]	–	[0.56]
Word Count Statistics			
Mean	[20.46]	[20.03]	[20.89]
Std Dev	[5.73]	[5.55]	[9.69]
Character Count Statistics			
Mean	[111.77]	[113.22]	[112.93]
Std Dev	[31.97]	[30.55]	[52.88]
Vocabulary Statistics			
Total Tokens	[236,123]	[195,193]	[74,351]
Unique Words	[11,505]	[21,634]	[13,240]
Vocab. Richness	[0.0487]	[0.1108]	[0.1781]

Short Jokes Dataset (20,000 samples): Obtained from the "Jokes Short and Funny" collection[2], we extracted 20,000 jokes using similar filtering criteria (5-30 words, rare word pairs). This dataset serves as the retrieval corpus for RAG-based generation, providing contextual examples during inference. The RAG database was constructed by embedding all 20,000 Short Jokes using Sentence-BERT and indexing them with FAISS.

HaHackathon Dataset (8,000 samples)[14]: This dataset[3] contains humor samples with continuous ratings (0-4 scale). We used all samples without length filtering to train a Mistral-based evaluation model. We applied text normalization, and tokenized the text with a maximum length of 512 tokens for fine-tuning Mistral 3.

[1] https://github.com/orionw/rJokesData

[2] https://app.gigasheet.com/spreadsheet/jokes-short-and-funny/58c7db9f_ba90_4fbf_a2a8_54cc2acb71da

[3] https://aclanthology.org/2021.semeval-1.9

4. Methodology

This section describes the system architecture and the techniques behind each component of our humor generation pipeline for the subtasks.

4..1 System Architecture Overview

Our system follows a generate-then-score paradigm orchestrated as a stateful directed graph. Given a word pair or a headline, two generators, a fine-tuned Llama 3 model and GPT-OSS-120B, produce candidate jokes in parallel. The Llama generator runs in two modes (with and without RAG), while GPT-OSS produces additional candidates independently. All candidates are pooled and scored by a dual evaluation system: a fine-tuned Mistral-based humor scorer (primary) and a GPT-OSS multi-criteria scorer (secondary, used to investigate evaluation bias). If the best score meets a quality threshold or the maximum iterations are reached, the best joke is returned; otherwise, the pipeline loops for another round of generation and scoring.

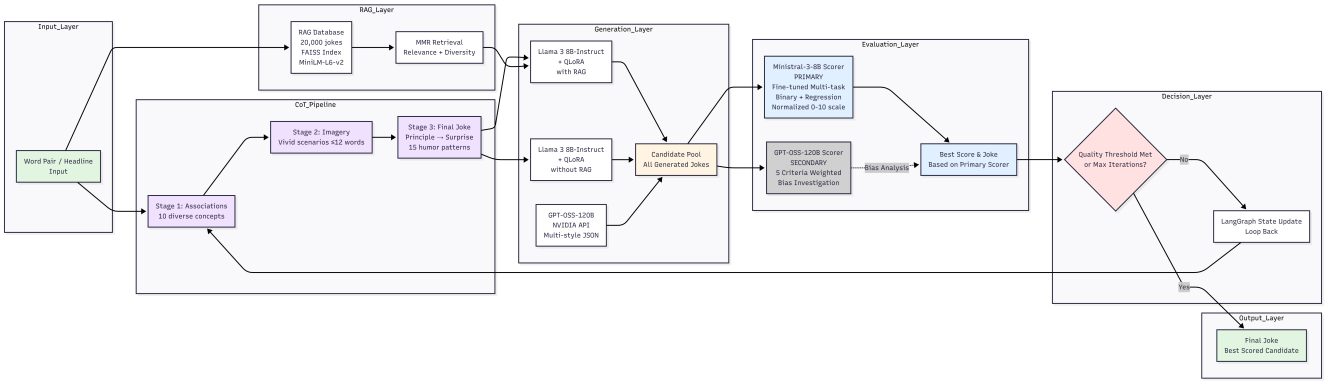

Figure 1: System overview: retrieval-augmented generation, multi-candidate sampling, and evaluation-driven selection.

4..2 Fine-Tuned Llama 3 with QLoRA

We use Meta's **Llama 3 8B-Instruct** as our base generator, adapted for humor generation through **QLoRA (Quantized Low-Rank Adaptation)**. QLoRA combines 4-bit NF4 quantization of the frozen base model with trainable low-rank adapter matrices applied to all attention and feed-forward projection layers. The model is trained on joke examples formatted as Llama 3 chat conversations, where the user message contains a word pair constraint and the assistant message contains a joke.

4..3 Chain-of-Thought Prompt Pipeline

Rather than prompting the model with a single instruction, we implement a three-stage chain-of-thought (CoT) pipeline that structures the creative reasoning process:

1. **Associations**: The model generates 10 diverse associations for the input word pair, mixing objects, emotions, places, actions, and metaphors to broaden the conceptual space.

2. **Imagery**: Associations are transformed into vivid, concise mental images (≤ 12 words each), converting abstract concepts into concrete scenarios suitable for humor.

3. **Final Joke**: Imagery candidates are combined with a pattern-specific prompt following a **Principle → Surprise** framework: the joke establishes a clear expectation and then subverts it logically but unexpectedly.

The pipeline supports 15 humor patterns (puns, absurd logic, mini stories, sarcasm, deadpan, rule of three, literalism, etc.), each with a defined principle-surprise structure and style guide. The humour patterns are extracted from comedy textbooks and chosen at random by the model

4..4 Retrieval-Augmented Generation (RAG)

To provide stylistic inspiration at inference time, we implement a RAG system over a separate database of 20,000 jokes. Embeddings are created using **sentence-transformers/all-MiniLM-L6-v2** and indexed with **FAISS** for cosine similarity search. To avoid retrieving near-identical examples, we apply **Maximal Marginal Relevance (MMR)**, which iteratively selects results balancing relevance with diversity. Retrieved jokes are included in the prompt as inspiration, with instructions to analyze humor mechanisms rather than copy content.

4..5 GPT-OSS-120B Joke Generator

As a complementary generator, **GPT-OSS-120B** (accessed via the NVIDIA API) brings the broad creative capabilities of a larger model. It uses a system prompt establishing the model as a comedy writer and requests jokes in multiple comedic styles, with JSON-formatted output for reliable parsing.

4..6 Dual Evaluation System

Evaluating humor is inherently subjective, and using the same model family for both generation and evaluation risks systematic bias. We therefore implement two independent scorers.

Fine-Tuned Mistral Scorer (Primary). A **Mistral-3-8B** model fine-tuned as a multi-task model on the HaHackathon dataset, which contains human humor ratings on a 0–4 scale. The architecture uses a shared transformer backbone with two task-specific heads: a classification head for binary humor detection and a regression head for predicting humor ratings. Raw scores are normalized to a 0–10 scale using a piecewise linear mapping calibrated against the training data distribution.

GPT-OSS-120B Scorer (Secondary / Bias Experiment). GPT-OSS provides multi-criteria quality assessment across five dimensions: creativity (25%), word integration (20%), humor impact (30%), structure (15%), and cleverness (10%). This scorer serves a dual purpose: complementary quality assessment and investigation of whether same-model generation and evaluation introduces scoring bias. In our final evaluation, only the Mistral scores are considered as the primary metric; GPT-OSS scores are reported separately for the bias analysis.

4..7 LangGraph Orchestrator

The pipeline is orchestrated using **LangGraph**, a framework for stateful multi-step workflows as directed graphs. Both generators are invoked in parallel, their outputs are pooled, scored by both

evaluators, and routed through a conditional edge: if the quality threshold is met or maximum iterations are reached, the pipeline outputs the result; otherwise, it loops back. The state tracks all jokes, scorer outputs, iteration history, and the best result throughout.

5. Experimental Setup

5..1 Llama 3 Fine-Tuning

The Llama 3 8B-Instruct model is fine-tuned on 25,000 joke examples using QLoRA. The base model is loaded in 4-bit NF4 quantization with bfloat16 compute dtype. LoRA adapters (rank 16, alpha 32, dropout 0.05) are applied to all seven projection layers (q, k, v, o, gate, up, down). Training runs for 3 epochs with a batch size of 20, a learning rate of 2e-4 with cosine scheduling and 3% warmup, and a maximum sequence length of 256 tokens. The paged AdamW 8-bit optimizer and gradient checkpointing are used to fit training within a single GPU. The dataset is split 95/5 into training and evaluation sets, with the best checkpoint selected by lowest evaluation loss.

5..2 Mistral Evaluator Fine-Tuning

The Ministral-3-8B model is fine-tuned on the HaHackathon dataset (8,000 texts with binary humor labels and 0–4 humor ratings). The base model uses the same 4-bit NF4 quantization. LoRA adapters share the same configuration as Llama (rank 16, alpha 32, dropout 0.05, all projection layers). The multi-task loss combines cross-entropy for classification and MSE for regression with equal weighting. Training runs for 3 epochs with a batch size of 4, a learning rate of 2e-4 with cosine scheduling and 136 warmup steps, and a maximum sequence length of 512 tokens. The dataset is split 90/10 into training (7,200) and validation (800) sets. we use early stop for finding breaking out point. After plotting the learning curve, we choose the weight with the least validation loss

5..3 Inference Configuration

The fine-tuned Llama generates with temperature 0.65 and top-p 0.9. The CoT prompt pipeline uses progressively increasing temperatures: 0.7 for associations, 0.8 for imagery, and 0.9 for final joke generation. GPT-OSS-120B generates at temperature 0.8 and scores at 0.4 for consistency. The Mistral evaluator runs deterministically with no dropout. The RAG system retrieves from a pool of 12 candidates, applies MMR with $\lambda = 0.4$, and includes the top 3 examples in the prompt. The pipeline runs with a maximum of 1 iteration per word pair or headline and a score threshold of 7.0/10.

5..4 Compute Environment

Both Llama and Mistral fine-tuning, as well as all inference, were run locally on an NVIDIA RTX 5090 Laptop GPU (26 GB VRAM). The combination of 4-bit quantization and LoRA adapters enabled both training and multi-model inference within single-GPU memory constraints.

6. Evaluation and Results

Metrics. We report results using a fine-tuned Mistral evaluator (raw scale $[0, 4]$), linearly normalized to $[0, 10]$. Due to potential self-preference bias when using GPT as both generator and evaluator, GPT-based scoring is treated only as an auxiliary bias check and is not used for final reporting. Figures and example jokes are deferred to the Appendix due to the 7-page limit.

Overall performance (Mistral-only). Table 2 summarizes overall results. Word-Inclusion achieves a mean score of 5.29 over 198 candidates, while Headline humor is lower on average (mean 4.08 over 61 samples), suggesting the headline setting is more challenging.

Task	N	Mean	Std	Range
Word-Inclusion	198	5.286	0.580	[2.68, 7.05]
Headline	61	4.080	1.877	[0.00, 6.42]

Table 2: Overall Mistral-normalized scores (0–10).

Ranking alignment and source effects. Under Mistral scoring, the Word-Inclusion rank-score correlation is weak (corr ≈ -0.05), indicating the current rank ordering is not strongly aligned with the final metric. For the headline task, Mistral scores are relatively stable across generator sources (Table 3), supporting the use of Mistral as a more comparable evaluator across generators.

Source	N	Mean	Std	Max
GPT	25	4.234	1.636	5.66
Llama	36	3.972	2.044	6.42

Table 3: Headline task: Mistral scores (0–10) by generator source.

7. Analysis and Discussion

Key observations. Headline humor is harder than Word-Inclusion under the Mistral metric (Table 2). For Word-Inclusion, the weak rank-score correlation (corr ≈ -0.05) suggests that our ranking stage is not well calibrated to the final evaluation signal and remains a bottleneck.

Failure modes (lightweight manual inspection). From a small qualitative review (see Appendix examples), low-scoring outputs commonly show: (i) forced keyword insertion without narrative necessity, (ii) weak or missing punchlines (descriptive rather than humorous), (iii) incoherent setup-payoff transitions, and (iv) generic templates that reduce surprise.

Evaluator bias and metric choice. Automatic evaluation can be biased. In particular, using the same model family for generation and evaluation may introduce self-preference effects, potentially distorting Top-K selection. Therefore, we use the independent Mistral evaluator as the primary metric for final reporting and treat GPT-based scoring only as an auxiliary bias-detection experiment.

Implications. To improve reliability, future iterations should align reranking directly with the final metric (or use preference-based ranking / small-scale human evaluation) and incorporate more robust evaluation beyond a single automatic scorer.

8. Conclusion

We presented a constrained humor generation pipeline for two settings: (i) word-inclusion jokes and (ii) humorous headline rewriting. To reduce evaluator-source bias, we report final results primarily using a fine-tuned Mistral evaluator (normalized to 0–10), treating GPT-based scoring only as an auxiliary bias check. Quantitatively, Word-Inclusion achieves a mean Mistral score of 5.29 over 198 candidates, while the Headline task is more challenging with a mean of 4.08 over 61 samples. A key limitation is that the current ranking order is weakly aligned with the final (Mistral) metric for Word-Inclusion (corr ≈ -0.05), suggesting reranking remains a bottleneck. Overall, our results indicate that candidate diversification is necessary for constrained humor generation, but reliable selection and evaluation are the critical factors for consistent quality.

9. Future Work

Align reranking with the final metric. Since rank-score alignment is weak under the adopted Mistral evaluator, a direct next step is to rerank candidates using the same scoring signal (or train a lightweight reranker to predict the Mistral preference), rather than relying on mismatched or mixed scoring criteria.

More reliable evaluation. Automatic humor evaluation is inherently noisy and can be biased. Future work should incorporate small-scale human preference judgments, pairwise comparisons, or multi-evaluator consensus to improve robustness and reduce evaluator-specific artifacts.

Better control of constraint satisfaction. For word-inclusion, we can strengthen constraint adherence while preserving fluency by adding explicit constraint checks (e.g., exact keyword match, placement rules) and penalizing forced insertion patterns.

Task-specific improvements for headlines. For headline humor, adding controllable style constraints (brevity, factual anchor preservation, and a clearer punchline structure) and using retrieval for stylistic examples may improve consistency without sacrificing relevance.

Bibliography

[1] Kim Binsted and Graeme Ritchie. An implemented model of punning riddles. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 1–6. AAAI Press, 1994.

[2] Oliviero Stock and Carlo Strapparava. Hahacronym: Humorous agents for humorous acronyms. *Humor: International Journal of Humor Research*, 16(3):297–314, 2003.

[3] Yuchen Su, Yonghua Zhu, Ruofan Wang, Zijian Huang, Diana Benavides-Prado, and Michael Witbrock. A survey of pun generation: Datasets, evaluations and methodologies. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7375–7395, 2025.

[4] Anirudh Mittal, Yufei Tian, and Nanyun Peng. Ambipun: Generating humorous puns with ambiguous context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1053–1062. Association for Computational Linguistics, 2022.

[5] Joe Toplyn. Witscript 3: A hybrid ai system for improvising jokes in a conversation. *arXiv preprint arXiv:2301.02695*, 2023.

[6] Dmitry Vikhorev, Daria Galimzianova, Svetlana Gorovaia, Elizaveta Zhemchuzhina, and Ivan P Yamshchikov. Cleancomedy: Creating friendly humor through generative techniques. *arXiv preprint arXiv:2412.09203*, 2024.

[7] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

[8] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in neural information processing systems*, 36:10088–10115, 2023.

[9] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc., 2020.

[10] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, page 335–336, New York, NY, USA, 1998. Association for Computing Machinery.

[11] Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. Gptscore: Evaluate as you desire, 2023.

[12] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023.

[13] Ritsu Sakabe, Hwichan Kim, Tosho Hirasawa, and Mamoru Komachi. Assessing the capabilities of llms in humor:a multi-dimensional analysis of oogiri generation and evaluation, 2025.

[14] J. A. Meaney, Steven Wilson, Luis Chiruzzo, Adam Lopez, and Walid Magdy. SemEval 2021 task 7: HaHackathon, detecting and rating humor and offense. In Alexis Palmer, Nathan Schneider, Natalie Schluter, Guy Emerson, Aurelie Herbelot, and Xiaodan Zhu, editors, *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 105–119, Online, August 2021. Association for Computational Linguistics.

Appendix

1. Additional Figures and Example Outputs

Due to the 7-page limit, we include additional plots and representative examples here.

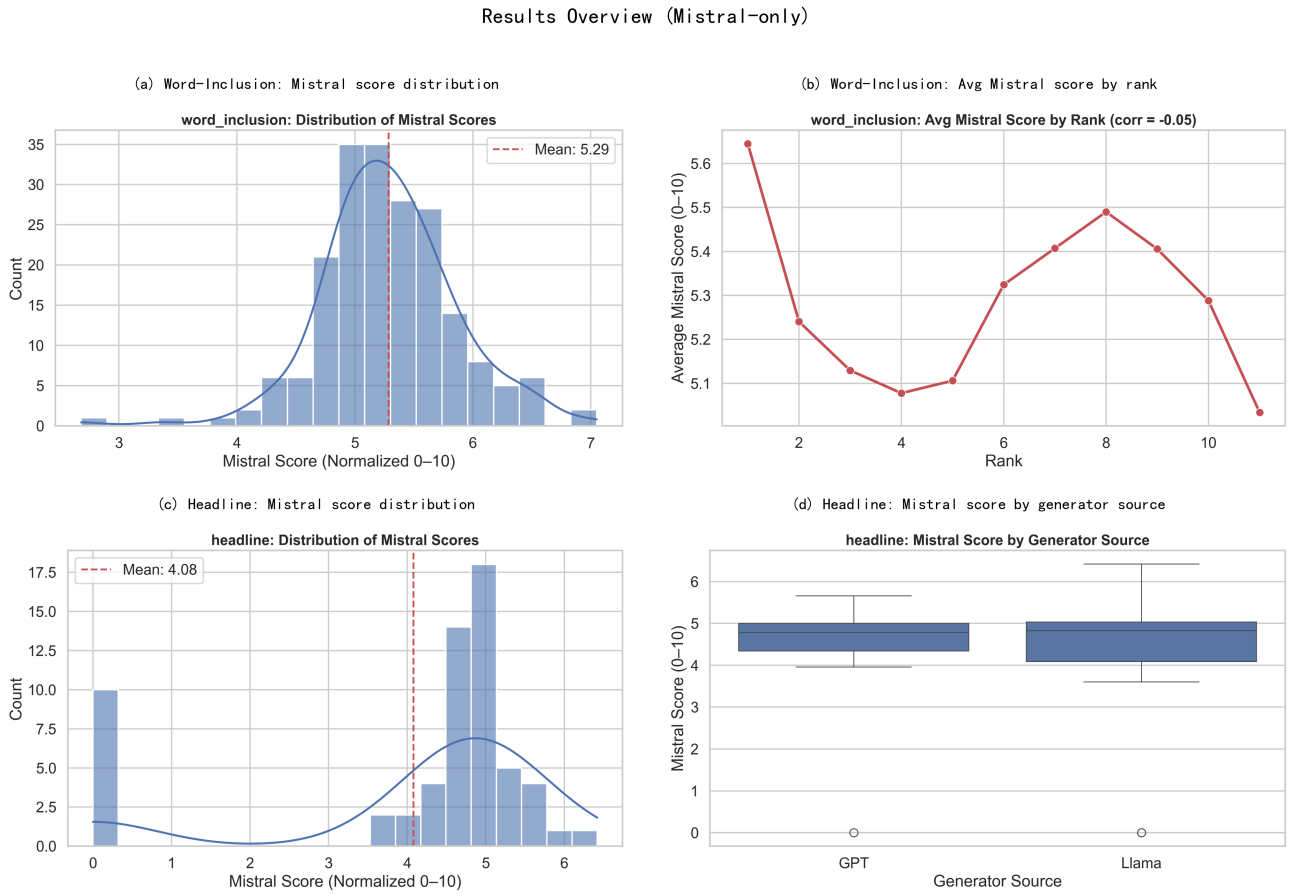

Figure 2: Mistral-only results overview: (a) Word-Inclusion score distribution, (b) Word-Inclusion score vs. rank, (c) Headline score distribution, (d) Headline score by generator source.

Figure 3: Dataset comparison across sources: token/character distributions and representative vocabulary statistics.

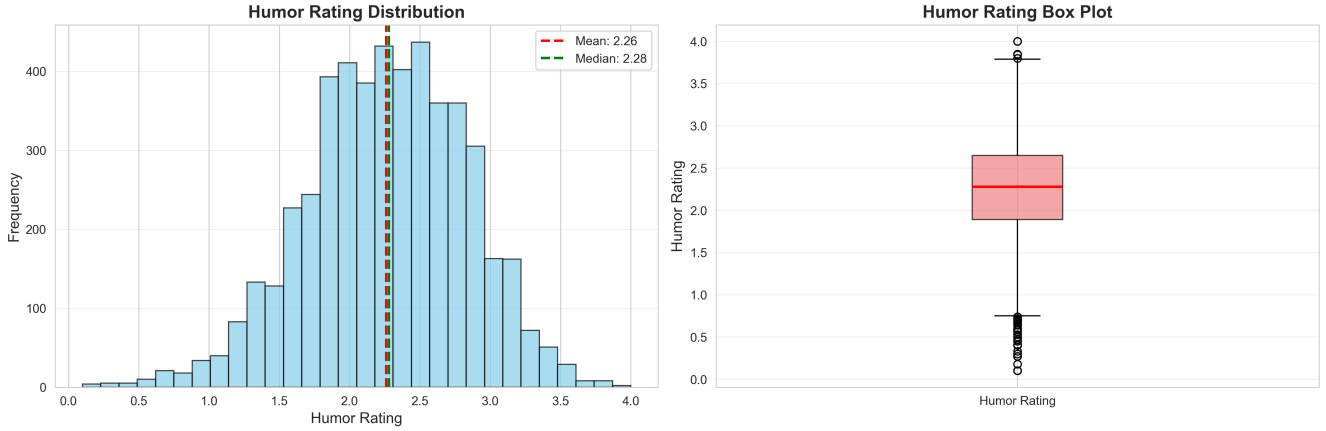

Figure 4: Humor rating distribution in the HaHackathon dataset (0-4 continuous scale).

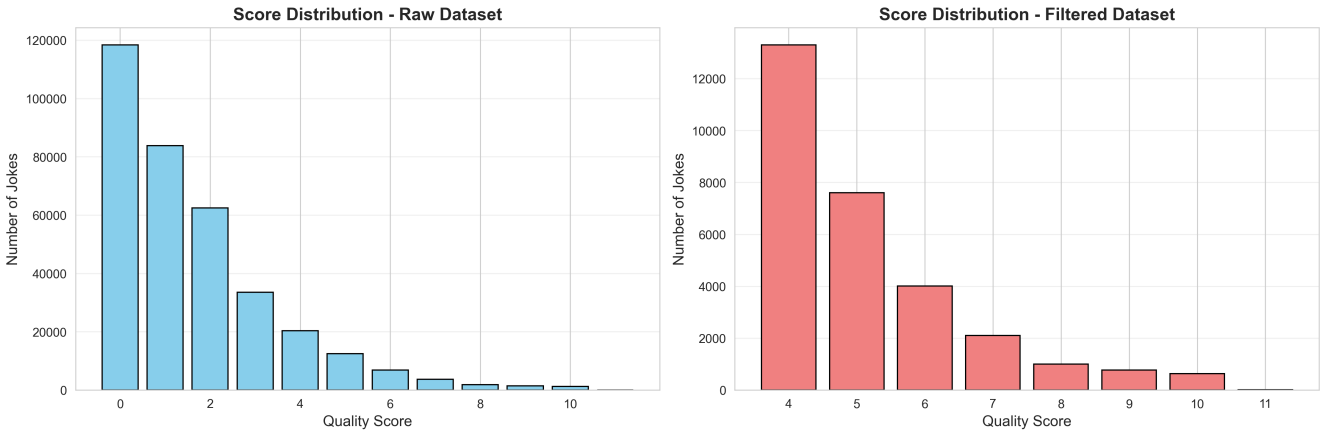

Figure 5: Score distribution for rJokes dataset. Left: raw dataset (scores 0-9). Right: filtered dataset.

2. Technical Steps

2..1 Checkpoint selection

Mistral (evaluator). We selected the LoRA checkpoint `checkpoint-2300`. Empirically, the evaluation loss reached its minimum around step ≈ 2350. Later steps showed mild overfitting (train loss continuing to decrease while eval loss stopped improving).

Ministral-3-8B (evaluator). We selected `checkpoint-600` as the best checkpoint based on the highest composite validation score (combining evaluation loss, regression R^2, and classification F1). Performance after ≥ 1200 steps indicated overfitting on the smaller rated subset.

2..2 Evaluation metrics

Classification metrics

- **Accuracy:** proportion of correct humor/not-humor predictions.
- **Precision/Recall/F1:** computed for the humor class (positive class).

Regression metrics

- **MAE:** mean absolute error between predicted and gold rating (rated subset).
- **RMSE:** root mean squared error (rated subset).
- R^2**:** coefficient of determination on the rated subset.

2..3 Learning curves

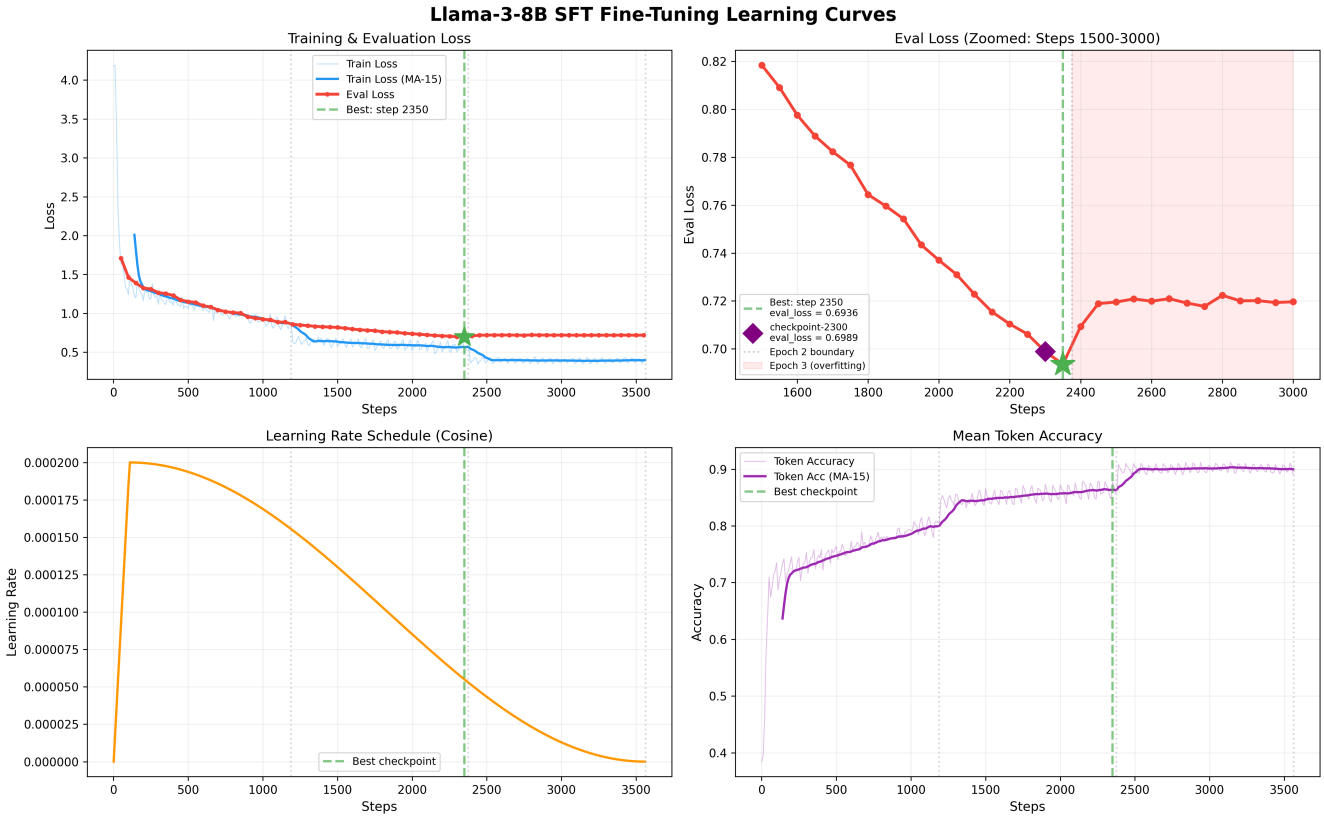

Figure 6: llama learning curves with training loss and validation metrics to find the optimal model weight

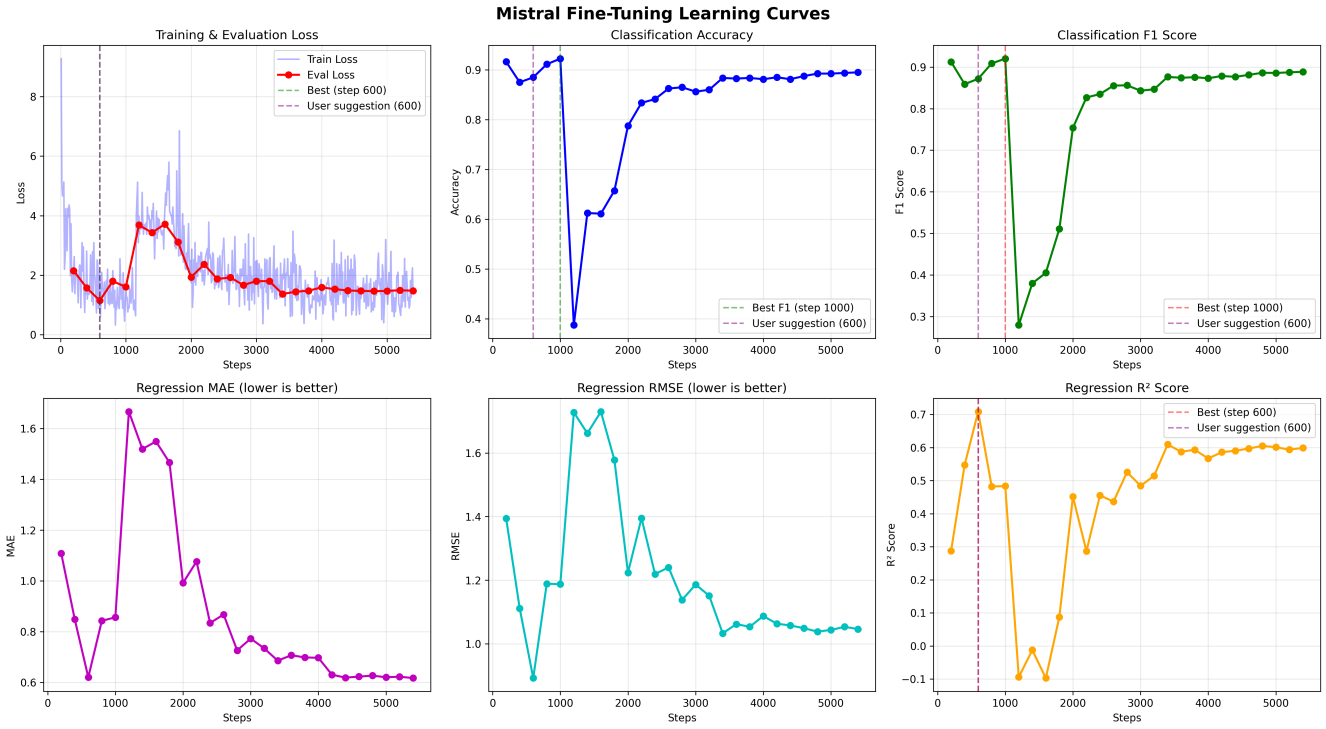

Figure 7: mistral learning curves with training loss and validation metrics to find the optimal model weigh

2..4 Fine-tuning Configuration

Parameter	Value
Base model	`meta-llama/Meta-Llama-3-8B-Instruct`
Method	SFT + LoRA (QLoRA 4-bit)
LoRA rank (r)	16
LoRA α	32
LoRA dropout	0.05
Target modules	`q,k,v,o,gate,up,down_proj`
Max seq length	256
Learning rate	2×10^{-4}
Scheduler	cosine
Epochs	3
Batch size	20
Precision	BF16
Gradient checkpointing	enabled
Optimizer	`paged_adamw_8bit`
Save steps / Eval steps	100 / 50

Table 4: Llama-3-8B LoRA SFT configuration.

Table 5: Ministral3 multi-task fine-tuning LoRA Configuration for

Parameter	**Value**
r	16
lora_alpha	32
lora_dropout	0.05
bias	none
task_type	SEQ_CLS
target_modules	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj
modules_to_save	cls_head, reg_head
output_dir	./checkpoints_ministral3_multitask
num_train_epochs	3
per_device_train_batch_size	4
per_device_eval_batch_size	8
gradient_accumulation_steps	1
learning_rate	2e-4
weight_decay	0.01
logging_steps	10
save_steps	200
eval_strategy	steps
eval_steps	200
bf16	True
fp16	False
max_grad_norm	1.0
warmup_steps	136
lr_scheduler_type	cosine
report_to	tensorboard
dataloader_num_workers	4