ORPAILLEUR & SyNaLP at CLEF 2024 Task 2: Good Old Cross Validation for Large Language Models Yields the Best Humorous Detection

Github: https://github.com/PierreEpron/irony-categories

Pierre Epron[1], Gaël Guibon[1,3] and Miguel Couceiro[1,2]

[1]*LORIA, Université de Lorraine, CNRS, France*
[2]*INESC-ID, IST, Universidade de Lisboa, Portugal*
[3]*LIPN, Université Sorbonne Paris Nord, France*

Abstract
In the context of the JOKER 2024 Task 2 Challenge, this paper presents an emerging approach that leverages the latent representations derived from different Large Language Models (LLMs) to drive a classification mechanism. Our methodology involves exploiting the "knowledge" encoded in LLMs to effectively discriminate humor genres. Experimental results show promising results, demonstrating the effectiveness of our approach. However, inherent complexities remain, such as the proximity between certain classes and biases arising from the dataset distributions. These complexities warrant further investigation to refine the classification process and improve overall performance.

Keywords
Humor genre classification, Large language models, Text classification

1. Introduction

In this paper, we propose to address the JOKER 2024 task 2, which concerns the classification of humor categories, with an approach that utilizes the hidden representations generated by different Large Language Models (LLMs) as input to a classification head. Our hypothesis is that the sophisticated representation capabilities of LLMs can facilitate more efficient and accurate classification of complex semantic categories, such as those associated with humor. Humor, with its various genres and subtle nuances, presents a unique challenge for Natural Language Processing (NLP) systems. By leveraging the deep, contextual embeddings produced by LLMs, we aim to improve the classification performance for humor genres. To investigate this, we address several detailed research questions:

- RQ1: **Does the "knowledge of the world" embedded in LLMs facilitate more accurate classification of humor genres?** We hypothesize that the extensive pretraining on a diverse range of texts enables LLMs to grasp subtle nuances of humor, potentially leading to more precise classification outcomes.
- RQ2: **Are the classification results consistent across different LLMs or do we actually see an improvement with the recency of the model?** To this end, we compare the performance of Llama2, Llama3, and Mistral to ascertain whether newer models, such as Llama3, outperform their predecessors.
- RQ3: **How does the use of a QLoRA adapter on hidden layers affect classification results?** QLoRA adapters are capable of modifying the representations within the hidden layers, and this study aims to verify whether this adaptation enhances the classification accuracy.

CLEF 2024: Conference and Labs of the Evaluation Forum, September 09–12, 2024, Grenoble, France
pierre.epron@loria.fr (P. Epron); gael.guibon@loria.fr (G. Guibon); miguel.couceiro@loria.fr (M. Couceiro)
https://pierreepron.github.io/ (P. Epron); https://gguibon.github.io/ (G. Guibon); https://members.loria.fr/mcouceiro/ (M. Couceiro)
0000-0003-2420-9311 (G. Guibon); 0000-0003-2316-7623 (M. Couceiro)

- RQ4: **Which hidden layer depth of LLMs yields the best classification results?** We investigate whether certain layers of the network provide more relevant features for classification and if this optimal depth is consistent across different LLMs and humor categories.

By addressing these questions, our study aims to deepen the understanding of how LLMs can be effectively utilized for complex classification tasks and to identify the factors that contribute to their performance. This research addresses the applicability of LLMs to the complex challenge of humor classification.

1.1. Related Work

The necessity for automatic humor detection arises from the increasing influence of conversational agents and the omnipresence of social media platforms. In the digital realm, where interactions are increasingly mediated by algorithms, discerning humor has become crucial. This imperative extends to various applications, including chatbots, recommender systems, social media reputation management, and the crucial task of identifying and combating fake news and hate speech [1, 2]. Early efforts in humor detection primarily focused on the intricate dynamics of wordplay. The seminal evaluation campaign explored tasks including pun detection, pun location, and pun interpretation [3]. However, a significant challenge in this domain has been the scarcity of appropriate training data, particularly evident for languages beyond English. Recent advances in automatic humor detection have been driven by the development of contextualized embeddings, which have facilitated a broader recognition of humor across diverse contexts [4]. Moreover, the development of multilingual models, which leverage the pre-trained BERT architecture [5], has expanded the scope of humor recognition to languages such as Chinese, Russian, and Spanish [6]. Additionally, there has been a notable shift towards addressing domain-specific tasks, examplified by endeavors to identify humorous queries within Q&A systems [7]. While the field of irony and sarcasm detection has received considerable attention [8, 9, 10], the area of automatic humor detection remains a vital and evolving area of research with implications that extend to various facets of human-computer interaction and online discourse [11, 12].

The landscape of LLM development is undergoing a period of rapid evolution, with the introduction of new models such as LLaMA2, Mistral, and the GPT family models, including GPT-3 and GPT-4. Touvron et al. [13] introduced LLaMA2, which builds upon its predecessor by enhancing the model architecture and training methodologies, thereby achieving improved performance across a range of natural language processing tasks. Jiang et al. [14] presented Mistral, a model known for its efficiency and effectiveness, particularly in low-resource settings, demonstrating impressive capabilities in several benchmarks. Concurrently, the GPT family of models, developed by OpenAI, has made significant contributions to the field. GPT-3, introduced by Brown et al. [15] set new standards with its 175 billion parameters, enabling unprecedented performance in generating human-like text and performing complex language tasks with minimal prompt engineering. Building on this, GPT-4 [16] and Llama3 [17], as detailed in its model card, further enhanced these capabilities by incorporating more sophisticated training techniques and a larger training corpus, resulting in superior performance across a wider range of applications. These models have played a pivotal role in advancing the state of the art in natural language understanding and generation, solidifying their position as indispensable tools in the Natural Language Processing community [18, 19, 20].

Zero-shot and few-shot learning methods enable LLMs to perform tasks with minimal task-specific training data. Some studies [21, 22] introduce the concept of using LLMs for zero-shot learning, demonstrating that models can generalize from pre-trained knowledge to new tasks without explicit training examples. This further enhances their versatility and application scope. Probing and feature-based fine-tuning involve using LLMs as classifiers by extracting and utilizing internal representations for specific tasks. A study [23] presented a method where prompts are augmented to probe LLMs for specific linguistic features, effectively turning them into classifiers for various natural language processing tasks. This technique demonstrates the adaptability of LLMs in understanding and categorizing complex linguistic patterns. LoRA (Low-Rank Adaptation) and QLoRA (Quantized Low-Rank Adaptation) are techniques designed to enhance the efficiency of fine-tuning of LLMs by reducing the number of trainable parameters. Hu et al. [24] introduced LoRA, which introduces trainable rank-decomposition matrices into the layers of the transformer, significantly reducing the computational cost of fine-tuning. Dettmers et al. [25] subsequently optimized this approach with QLoRA, which incorporates quantization to further enhance efficiency while maintaining model performance.

2. Dataset

The dataset comes from JOKER 2024 shared task 2 [26, 27]. It consists of 1,742 humorous texts labelled in 6 different categories.

- IR - Irony relies on a gap between the literal meaning and the intended meaning, creating a humorous twist or reversal.
- SC - Sarcasm involves using irony to mock, criticize, or convey contempt.
- EX - Exaggeration involves magnifying or overstating something beyond its normal or realistic proportions.
- AID - Incongruity refers to the unexpected or contradictory elements that are combined in a humorous way and Absurdity involves presenting situations, events, or ideas that are inherently illogical, irrational, or nonsensical.
- SD - Self-deprecating humour involves making fun of oneself or highlighting one's own flaws, weaknesses, or embarrassing situations in a lighthearted manner.
- WS - Wit refers to clever, quick, and intelligent humour and Surprise in humour involves introducing unexpected elements, twists, or punchlines that catch the audience off guard.

The primary challenge from the dataset is the imbalance in the number of examples for specific classes. For instance, the WS (Wit and Surprise) class contains 650 examples, while the EX (Exaggeration) class contains only 122 examples. The complete list of classes distribution is presented in Table 1. Another significant challenge is the proximity of certain classes, such as irony and sarcasm. Some definitions of irony include sarcasm as a form of irony [28]. For this corpus, the definition of irony aligns with the prevailing understanding of situational irony.

Table 1
The distribution of classes after removing duplicates and texts that are too long. "Count / 5" refers to the support of each class for 1 split since we use 5 splits for cross-validation.

Class	Count	Ratio	Count / 5
WS (Wit and Surprise)	632	0.371	126.4
SC (Sarcasm)	350	0.205	70.0
AID (Incongruity)	229	0.134	45.8
IR (Irony)	209	0.123	41.8
SD (Self-Deprecating)	162	0.095	32.4
EX (Exageration)	122	0.072	24.4
Total	1704	1.000	340.8

The majority of texts have a length between 20 and 40 tokens (Figure 1). However, some texts are very large, with a number of tokens exceeding 500. Due to the limitations of GPU memory, we have excluded examples with a number of tokens greater than 170 from the train corpus. Additionally, there are instances of duplication in the examples of the train set. Some of these instances are quasi-duplicates, as evidenced by the presence of the same example with the quotation mark. We removed the duplicates, but we could not identify nor remove the quasi-duplicates. At the conclusion of the cleaning process, we retained 1,704 examples from the original 1,742 set.

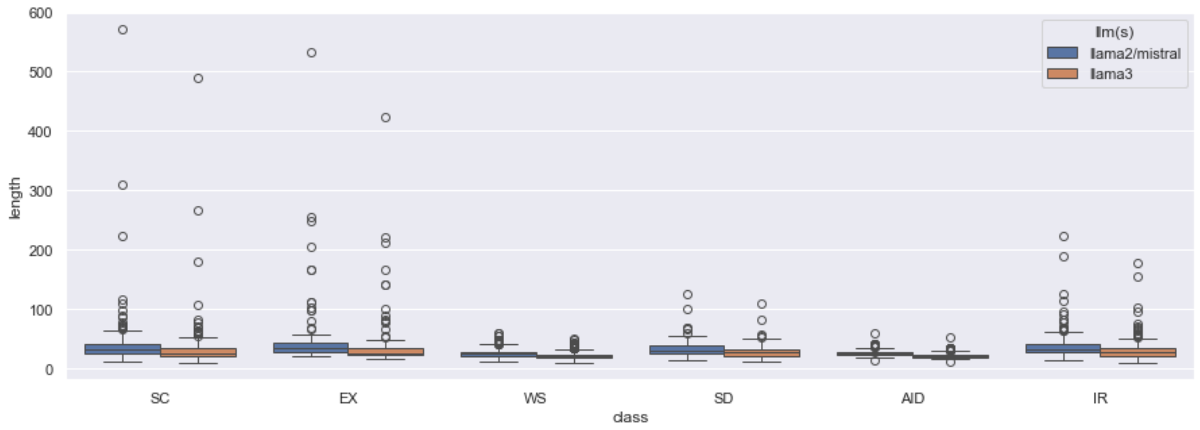

Figure 1: Length distribution of training set for each LLM tokenizer

3. Methodology

Our objective is to explore the potential of advanced LLMs within a consistent methodological framework. We employed a 4-bit quantized version of three distinct LLMs: Llama2-7b[1], Mistral-7b[2], and Llama3-8b[3]. These models are selected due to their varying characteristics, which allowed for a comparative analysis.

3.1. LLMs

The distinctions between the Mistral and Llama2 models are primarily due to several advanced optimization techniques employed by Mistral.

- **Sliding Window Attention** improves the efficiency of attention mechanisms by focusing on a moving window of tokens, rather than considering all tokens at once. This reduces the computational complexity and enhances the model's ability to handle longer sequences effectively.
- **Rolling Buffer Cache** maintains a cache of recently processed data, enabling faster retrieval and processing of these data chunks, thus improving overall model performance.
- **Pre-fill and Chunking** involve pre-processing and breaking down input data into manageable chunks, which can be processed more efficiently by the model, leading to better performance in terms of both speed and accuracy.

A significant difference between Llama2 and Llama3 lies in their tokenization strategies and vocabulary sizes.

- **Llama2 and Mistral:** Both models utilize the same tokenizer, which is based on Byte-Pair Encoding (BPE) and implemented using the sentencepiece approach. They share a vocabulary size of 32,000 tokens. Sentencepiece is a data-driven method that segments text into subword units, ensuring a balance between word-level and character-level tokenization.
- **Llama3:** This model employs a new tokenizer with a vastly increased vocabulary size of 128,256 tokens. While it also employs BPE, it uses the tiktoken approach developed by OpenAI for their GPT models. The key distinction between tiktoken and other tokenizers is its capacity to bypass the BPE algorithm when a token already exists in the vocabulary, potentially enhancing efficiency and tokenization speed.

The differences between vocabulary size could not be explained by the tokenizer implementation only. The larger vocabulary in Llama3 likely reflects differences in the scope and variety of training data used, although specific details about these datasets are not publicly available.

[1]https://huggingface.co/meta-llama/Llama-2-7b-chat-hf
[2]https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2
[3]https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

3.2. Classification

The primary approach involved the addition of a Feed Forward (FF) layer on the final token of the LLM representation, situated atop the last hidden state. This method sought to leverage the LLMs' capacity to generate rich contextual embeddings for downstream tasks. The Causal Language Model (CLM) is extended in the following manner.

Consider a sequence of tokens $X = (x_1, x_2, \ldots, x_T)$, where x_t represents the token at position t in the sequence. The CLM models the probability of the next token given the previous tokens:

$$P(X) = \prod_{t=1}^{T} P(x_t | x_1, x_2, \ldots, x_t - 1).$$

The CLM architecture we use in this paper can be streamlined as follows:

- **Input Embedding Layer:** Converts tokens into dense vector representations:

$$E(x_t) = Embedding(x_t).$$

- **Positional Encoding:** Adds positional information to the token embedding to retain the order of tokens:

$$PE(t) = PositionalEncoding(t).$$

  The input to the model at position t is:

$$H_t^0 = E(x_t) + PE(t).$$

- **Attention Mechanism:** Utilizes masked self-attention to ensure that the prediction for position t only depends on positions 1 to $t - 1$. The masked attention weights α_{ij} are computed by:

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{j} \exp e_{jk}},$$

  where e_{ij} is the compatibility function (e.g., dot product of queries and keys) and the mask ensures that $j \leq t - 1$.
- **Feed-Forward Layer**: Applied after the attention mechanism to introduce non-linearity:

$$H_l = FF(Attention(H_{l-1})).$$

- **Final Hidden Layer:** The last hidden state for each position t is denoted as H_t^L. To adapt the CLM for classification, we use the final hidden state of the last token H_T^L as the input to a classification Feed-Forward Layer.
- **Classification Feed-Forward Layer:** Projects the last hidden state to the class probabilities:

$$z = FF_c(H_T^L),$$

  where z is the logit vector representing unnormalized scores for each class c.
- **Output Layer:** Applies a softmax function to obtain the class probabilities:

$$P(y|X) = Softmax(z).$$

- **Training Objective for Classification:** The training objective is to minimize the cross-entropy loss between the predicted class probabilities and the true class labels:

$$L = -\sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} \log P(y = c | X_i),$$

  where N is the number of training samples, C is the number of classes, $y_{i,c}$ is a binary indicator (0 or 1) if class label c is the correct classification for sample i.

- **Inference for Classification:** During inference, the model predicts the class label by selecting the class with the highest probability:

$$\hat{y} = \operatorname*{argmax}_{c} P(y = c|X).$$

In addition, we experimented with integrating a QLoRA adapter into the query and value components of the attention heads within the LLMs. QLoRA adapters are designed to enhance model performance by allowing fine-grained tuning during training.

3.3. Cross-validation

Given the limited number of examples in the training set, we decided to use the same set for testing and validation. We implemented a stratified cross-validation approach to ensure the evaluation of our model. Specifically, we used 5 stratified splits to perform cross-validation using a leave-one-out cross-validation. This means that we ran each experiment 5 times, using 4 splits for training and 1 split for evaluation. This method ensures that each class is equally represented in both the training and validation sets across all splits, thus providing a comprehensive assessment of model performance.

During the validation phase, our primary monitoring metric was the Matthews Correlation Coefficient (MCC) [29, 30]. It is a performance metric for classification that considers all four components of the confusion matrix: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). It is defined by:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}.$$

The MCC ranges from -1 to 1, with 1 indicating a perfect prediction, 0 indicating no better than a random prediction, and -1 indicating total disagreement between the prediction and the actual values. The MCC is particularly valuable for evaluating models on unbalanced datasets, as it is less susceptible to the limitations of traditional metrics like accuracy or F-Score, which may give high scores to models that are biased towards the majority class. The MCC, however, accounts for the balance ratios between the classes and the quality of the predictions for both classes. By considering all aspects of the confusion matrix, it provides a more comprehensive evaluation of the classifier's performance, making it robust against class imbalance and more reflective of the true predictive capability of the model.

3.4. Training Parameters

We used a batch size of 16 and a maximum of 10 epochs for training. These parameters were chosen to balance training efficiency and model performance. Two different learning rates were used: $1e - 3$ for the Feed Forward layer and $1.5e - 4$ for the QLoRA adapter when it was included. The learning rates were selected based on preliminary experiments and common practices in fine-tuning LLMs. The FF layer typically benefits from a higher learning rate due to its role in direct task-specific adaptation, while the QLoRA adapter requires a more conservative rate to ensure stability and effective integration. To further refine the training process, we experimented with a linear learning rate scheduler and gradient clipping. These techniques are known to enhance training stability and performance, especially in the context of large models. we conducted an experiment to determine whether weighting could improve performance, especially in scenarios where the data is imbalanced.

Two configurations of the QLoRA adapter were tested: 64 rank for 16 alpha and 16 rank for 64 alpha. These configurations were chosen because it was demonstrated in the original LoRA paper that minimal rank with high alpha performs better [24]. Conversely, the QLoRA paper demonstrated that high rank with low alpha performs better [25]. Both configurations incorporated a dropout rate of 0.1 to prevent overfitting.

Finally, we trained the model on each different hidden layer of each LLMs to see if there was a relationship between class performance and depth.

3.5. Additional Details

We submitted 9 test results using three distinct strategies to assess model performance: ensemble, high, and low. In the ensemble strategy, we employed a majority vote among the five models derived through cross-validation. For the high and low strategies, we selected the optimal and sub-optimal splits obtained from cross-validation.

The 36 experiments conducted as part of this study were executed on the Grid5000 platform, with an average runtime between 2h00 and 2h30 on a Nvidia A100 GPU (40 GiB). While these values should be interpreted with caution, they provide a general indication of the cost and resources required to perform this type of research.

4. Results

In this section, we first look at the results obtained from our various experiments. All these results should be interpreted with caution. The variance of the results across the 5 splits does not always allow us to conclude that one model performs better than another, as is often the case with Llama2 and Llama3. Furthermore, the small size of the dataset and the absence of a test set also limit the interpretation of the results. Secondly, we look at the results obtained on the test set provided by the Joker shared task's organizers and evaluated independently

4.1. Parameters Results

The results of our parameter-focused experiments are reported in Table 2. A subset of the Llama2 experiments is reported here, with the full set of results being accessible within the GitHub repository[4]. The observations presented below are equally applicable to the other experimented LLMs.

We can see that balancing cross-entropy did not lead to any significant improvements in the model's performance. Despite the theoretical advantages of mitigating class imbalance, our findings demonstrated that the impact on metrics was significantly worse than expected. Conversely, implementing a linear scheduling strategy for the learning rate yielded a notable enhancement in the model's performance. This approach permitted a more gradual adjustment of the learning rate, which in turn facilitated better convergence and reduced overfitting, as evidenced by lower validation loss and higher overall accuracy.

The application of QLoRA demonstrated substantial improvements across all tested setups. QLoRA consistently enhanced performance metrics, indicating its robustness. Notably, configurations utilizing a rank of 16 and an alpha of 64 demonstrated superior results in every experimental setup. This indicates that the combination of reduced parameter dimensionality with a strong scaling factor can effectively capture essential features and nuances in the data, thereby enhancing the model's performances.

4.2. LLMs Results

In our comparative analysis of LLMs (see Table 3), we observed that LLama2 consistently outperformed both Llama3 and Mistral in their best setups. This finding underscores that the performance of LLMs is influenced by factors beyond the mere chronological advancement of the model. Notwithstanding the more recent architectural developments and potential enhancements in Llama3 and Mistral, LLama2's superior performance serves to highlight the importance of specific optimizations and configurations that can play a critical role in achieving better results.

4.3. Classification Results

In the context of our classification task, we observed varying levels of difficulty among different classes. The results are reported in Table 4 and Figure 2. In general, it can be observed that the different LLMs

[4]https://github.com/PierreEpron/joker2024-task2

Table 2
Parameter oriented results for Llama2-7b model. ↑ indicates that higher is better.

	Balanced	Scheluder	Macro F1↑	Weighted F1↑	Accuracy↑	MCC↑
	no	no	0.603 (0.057)	0.672 (0.043)	0.684 (0.034)	0.592 (0.042)
No QLoRA	yes	no	0.608 (0.041)	0.675 (0.030)	0.679 (0.036)	0.588 (0.046)
	no	yes	**0.626 (0.046)**	**0.690 (0.040)**	**0.695 (0.043)**	**0.605 (0.054)**
	no	no	0.630 (0.039)	0.696 (0.028)	0.700 (0.027)	0.612 (0.033)
64r, 16a	yes	no	0.633 (0.036)	0.700 (0.030)	0.705 (0.034)	0.616 (0.044)
	no	yes	**0.632 (0.025)**	**0.700 (0.024)**	**0.706 (0.026)**	**0.618 (0.033)**
	no	no	**0.661 (0.031)**	**0.720 (0.030)**	0.722 (0.031)	0.641 (0.040)
16r, 64a	yes	no	0.644 (0.024)	0.706 (0.022)	0.709 (0.021)	0.623 (0.028)
	no	yes	0.654 (0.032)	**0.720 (0.024)**	**0.731 (0.020)**	**0.653 (0.023)**

Table 3
The optimal results for each LLM. They were obtained using QLora with rank 16 and alpha 64. A linear scheduler was employed with a warmup of 1 epoch and a full training of 10 epochs. ↑ indicates that higher is better.

Model	Macro F1↑	Weighted F1↑	Accuracy↑	MCC↑
llama2-7b	0.654 (0.032)	0.720 (0.024)	**0.731 (0.020)**	**0.653 (0.023)**
llama3-8b	**0.663 (0.022)**	**0.721 (0.016)**	0.724 (0.014)	0.643 (0.018)
mistral-7b	0.652 (0.030)	0.717 (0.024)	0.718 (0.023)	0.636 (0.027)

exhibit comparable ease and difficulty across the various classes. This conclusion is supported by the observation that the confusion matrices are also highly similar.

It is notable that IR (Irony) and SC (Sarcasm) were challenging to differentiate, given their subtle distinctions and overlapping characteristics in textual expressions. Furthermore, EX (Exaggeration) frequently posed confusion, being misclassified as either irony or sarcasm due to their nuanced nature. Although instances of ambiguity between AID (Incongruity) and WS (Wit and Surprise) were less common, they still presented some classification challenges. Interestingly, the overall performance for WS (Wit and Surprise), SD (Self-Deprecating), and AID (Incongruity) was relatively robust. The high accuracy in classifying WS (Wit and Surprise) can be attributed to its status as the majority class, which naturally leads to a more substantial training set and better model performance. In contrast, the strong results for SD (Self-Deprecating) and AID (Incongruity) were somewhat unexpected, particularly for SD (Self-Deprecating), which is significantly underrepresented in the dataset. These findings suggest that while certain classes exhibit inherent complexities leading to misclassification, others (even with fewer examples) can achieve reliable identification when given appropriate model training.

Table 4
The optimal results for each LLM by classes. They were obtained using QLora with rank 16 and alpha 64. A linear scheduler was employed with a warmup of 1 epoch and a full training of 10 epochs. The support values are presented as intervals, as they may vary slightly depending on the split.

Model	IR (support=[42;42])			SC (support=[70;73])			AID (support=[45;49])		
	P	R	F1	P	R	F1	P	R	F1
llama2-7b	**0.557 (0.021)**	0.312 (0.170)	0.380 (0.123)	**0.622 (0.061)**	**0.737 (0.091)**	**0.669 (0.025)**	**0.803 (0.053)**	0.751 (0.093)	0.772 (0.054)
llama3-8b	0.447 (0.069)	0.412 (0.147)	0.423 (0.112)	0.598 (0.029)	0.689 (0.061)	0.639 (0.029)	0.777 (0.045)	**0.790 (0.042)**	**0.783 (0.033)**
mistral-7b	0.469 (0.036)	**0.433 (0.105)**	**0.444 (0.066)**	0.601 (0.063)	0.695 (0.077)	0.639 (0.026)	0.797 (0.055)	0.768 (0.071)	0.779 (0.037)
Model	SD (support=[32;34])			EX (support=[25])			WS (support=[126;128])		
	P	R	F1	P	R	F1	P	R	F1
llama2-7b	0.706 (0.145)	**0.767 (0.103)**	**0.722 (0.051)**	0.516 (0.123)	**0.502 (0.148)**	0.499 (0.112)	0.874 (0.021)	**0.892 (0.039)**	**0.882 (0.021)**
llama3-8b	**0.736 (0.078)**	0.702 (0.085)	0.715 (0.064)	**0.588 (0.027)**	0.501 (0.081)	**0.538 (0.045)**	**0.891 (0.017)**	0.870 (0.026)	0.880 (0.011)
mistral-7b	0.709 (0.057)	0.671 (0.144)	0.682 (0.090)	0.538 (0.049)	0.455 (0.045)	0.491 (0.032)	0.890 (0.038)	0.869 (0.041)	0.878 (0.025)

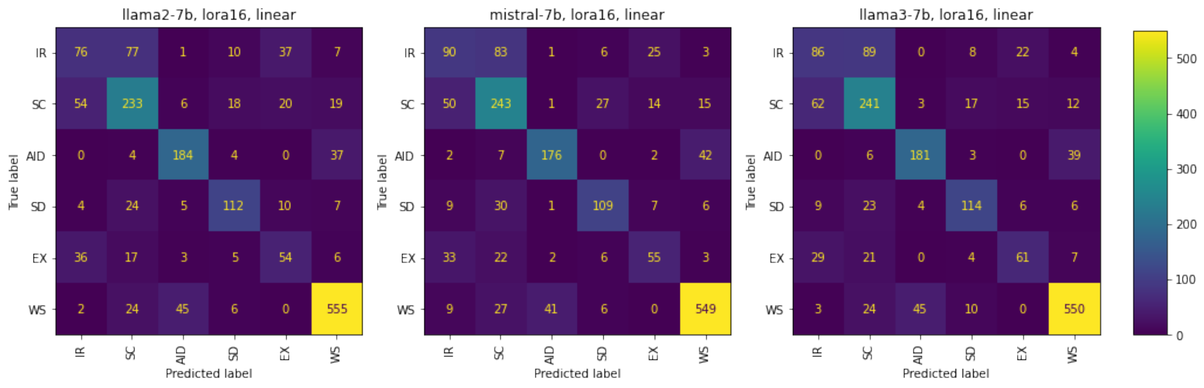

Figure 2: Confusion matrix aggregated over the five splits of the best run for each LLM.

4.4. Qualitative Results

We have identified two common types of errors that are found regardless of the LLMs. The first type consists of confusing WS (Wit and Surprise) with AID (Incongruity) when the text consists of a question followed by an answer. Each text presents a question followed by a clever or unexpected answer, often relying on homophones, similar-sounding words, or humorous reinterpretations of common phrases. The humor is derived from the audience's recognition of the pun or wordplay, resulting in a light-hearted and amusing effect. The format is simple and straightforward, making it easy to deliver and understand, typical of classic joke telling. Here are some examples:

- What do you call a fish wearing a crown? King Cod!
- What do you call a doctor who treats retired soldiers? A sawbone.
- Where do the pancakes live? In an apartment.
- What did the janitor say when he jumped out of the closet? Supplies!
- How does a penguin catch a fish? It just waddles down to the grocery store!

The second type of common mistake is to wrongly predict some texts to be SD (Self-Deprecating) that use the first person. This is perfectly understandable, since most SD examples should use the first person, so the model is biased by this. Some examples below:

- My poo is green, how festive.
- I'm mad at myself for not taking karate sooner.
- My name is Bet. I am a cutter.
- I always pronounce one word wrong. Wrong.

4.5. Hidden Layer Analysis

Although there is no clear association between the hidden index and the score, it is evident that certain classes, such as IR (Irony), are sensitive to the "deepness" of the model. Indeed, the performance of IR on low layers is less optimal for each model. These observations can also be made for SD (Self-Deprecating), EX (Exageration), and SD (Self-Deprecating), although to a lesser extent. Conversely, there are some classes, such as WS (Wit and Surprise) and AID (Incongruity), that appear to be relatively stable regardless of the depth of the model. As it might be expected, these results align with the overall performance of each class.

4.6. Submission results

All results submitted for the shared task CLEF 2024 Joker shared task number 2 are presented in Table 8. The four submissions with the highest scores are ours, and all of our submissions are among the top

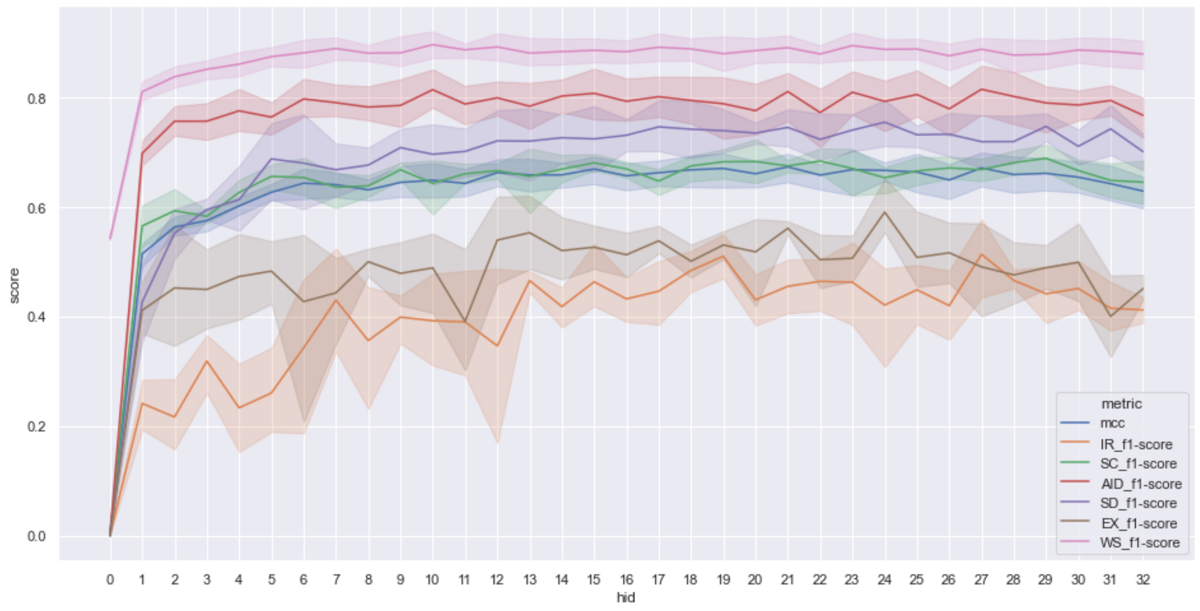

Figure 3: The value of different scores over the hidden layer index for llama2-7b best setup

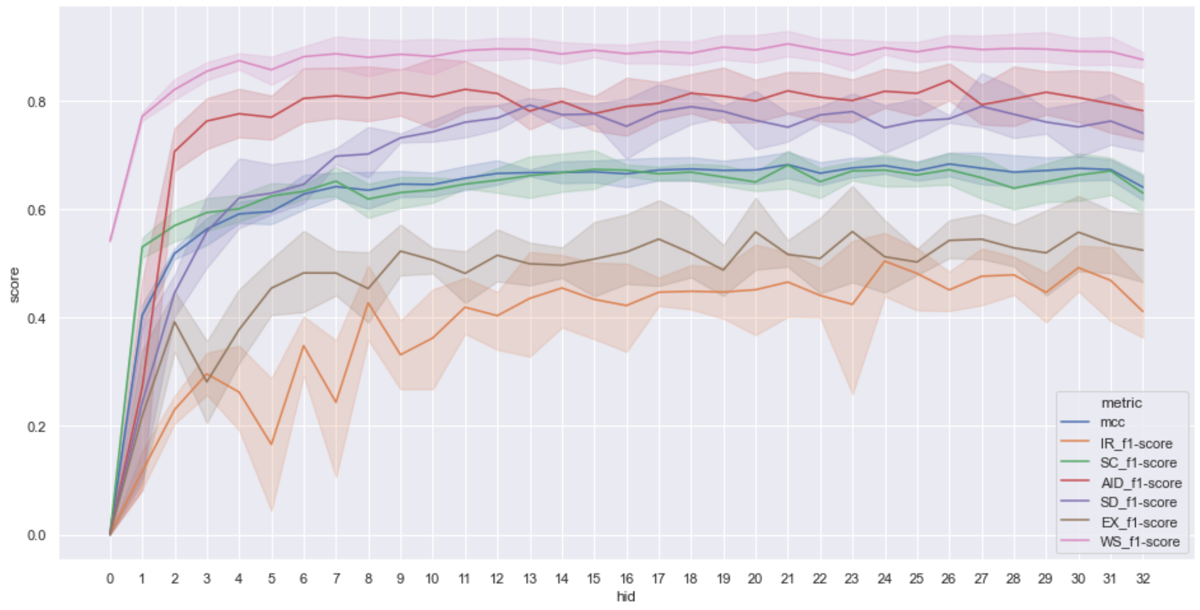

Figure 4: The value of different scores over the hidden layer index for mistral-7b best setup

12. In summary, this indicates that the methodology employed is both effective and consistent. In terms of macro F1-score, our highest-scoring submission achieved a score of 0.70, while our lowest-scoring submission achieved a score of 0.604. The second-best approach, apart from ours, achieved a score of 0.638. It is also noteworthy that other approaches using LLM have been submitted. While we lack the information to properly compare them, it appears that the relatively simple approach we employed is the most effective.

The detailed results of our submissions are presented in Tables 5 and 7. The first observation that can be made is that the performance of the strategies is consistent across LLMs. Specifically, the ensemble strategy consistently outperforms the high strategy, which in turn outperforms the low strategy most of the times. This observation is particularly noteworthy as it suggests that the optimal training set identified through cross-validation is also the most effective when evaluated on the final test set. If

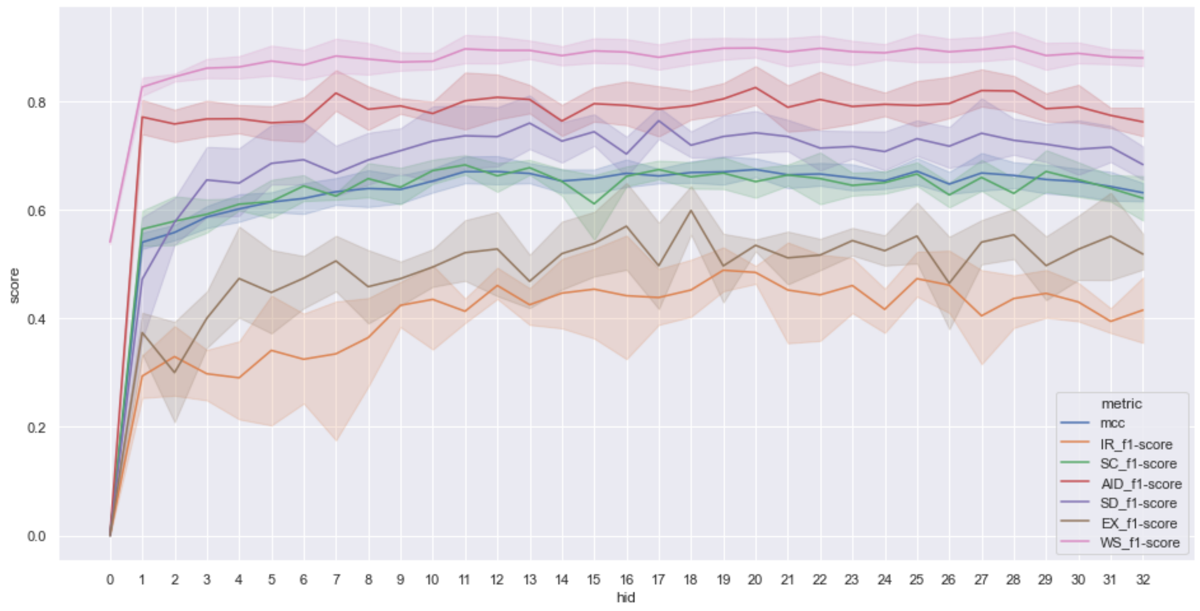

Figure 5: The value of different scores over the hidden layer index for llama3-8b best setup

the low-performing model had outperformed the high-performing model on the real test set, it would indicate that the optimal model identified through cross-validation may not be the optimal overall model.

As only three runs are submitted per model, it is not possible to definitively conclude that this is true for all five splits. Nevertheless, the observed trends offer valuable insights into the performance of the models and the reliability of cross-validation on this dataset.

The distribution of classes in the final test set differs from that of the train set. This is illustrated in Table 6. For example, there are 20% IR (Irony) examples in the test set where there are only 12% in the train set. Conversely, there are only 7% examples WS (Wit and Surprise) compared with 37% in the train set. Table 7 shows that for the AID (Incongruity) class, the F1-score increase of 1.2 compared with the train set. Furthermore, the poor scores obtained on the WS (Wit and Surprise) class are confirmed, 0.538 for the train set versus 0.521 for the test set. On the other hand, the IR (Irony) class results are particularly noteworthy, as they indicate a significant improvement over the train set results (+0.3 F1-score compared with the train set).

Table 5
Our submission results for each LLM. The best overall results are in **bold**. The best results for each LLM are underlined. ↑ indicates that higher is better.

Model	Strategy	Macro↑			Weighted↑			Accuracy↑
		P	R	F	P	R	F	
llama2	ens	0.684	0.672	0.659	0.737	0.738	0.721	0.738
	high	0.632	0.646	0.635	0.711	0.711	0.708	0.711
	low	0.649	0.635	0.617	0.708	0.701	0.683	0.701
mistral	ens	**0.714**	**0.697**	**0.700**	**0.753**	**0.756**	**0.749**	**0.756**
	high	0.669	0.657	0.660	0.719	0.723	0.719	0.723
	low	0.650	0.606	0.604	0.694	0.673	0.661	0.673
llama3	ens	0.675	0.652	0.659	0.724	0.727	0.723	0.727
	high	0.630	0.611	0.614	0.689	0.701	0.691	0.701
	low	0.638	0.626	0.629	0.696	0.699	0.695	0.699

Table 6
Comparison of the distribution of classes between the cleaned training set and the test set.

	Train set		Test set	
	Count	Ratio	Count	Ratio
IR	209	0.12	147	0.20
SC	350	0.21	59	0.08
AID	229	0.13	270	0.37
SD	162	0.10	91	0.13
EX	122	0.07	106	0.15
WS	632	0.37	49	0.07
Total	1704	1	722	1

Table 7
Our submission results for each LLM by classes. The best overall results are in **bold**. The best results for each LLM are underlined.

		IR (support=147)			SC (support=59)			AID (support=270)		
		P	R	F	P	R	F	P	R	F
llama2	ens	0.596	0.782	0.676	0.649	0.814	0.722	0.891	**0.911**	**0.901**
	high	0.669	0.592	0.628	0.524	0.729	0.610	0.885	0.885	0.885
	low	0.553	0.741	0.634	0.570	**0.831**	0.676	0.860	0.889	0.874
mistral	ens	**0.670**	**0.830**	**0.742**	0.733	0.746	**0.739**	0.869	0.881	0.875
	high	0.649	0.769	0.704	0.695	0.695	0.695	0.855	0.852	0.853
	low	0.492	0.816	0.614	0.714	0.508	0.594	0.888	0.793	0.838
llama3	ens	0.592	0.721	0.650	**0.765**	0.661	0.709	**0.903**	0.893	0.898
	high	0.563	0.667	0.611	0.731	0.644	0.685	0.874	0.896	0.885
	low	0.539	0.653	0.591	0.678	0.678	0.678	0.881	0.881	0.881

		SD (support=91)			EX (support=106)			WS (support=49)		
		P	R	F	P	R	F	P	R	F
llama2	ens	0.773	0.824	0.798	**0.643**	0.255	0.365	0.550	0.449	0.494
	high	0.755	0.813	0.783	0.510	**0.491**	0.500	0.450	0.367	0.404
	low	**0.823**	0.714	0.765	0.629	0.208	0.312	0.457	0.429	0.442
mistral	ens	0.763	0.780	0.772	0.639	0.434	0.517	**0.610**	**0.510**	**0.556**
	high	0.708	0.747	0.727	0.581	0.472	**0.521**	0.526	0.408	0.460
	low	0.667	0.835	0.741	0.535	0.217	0.309	0.605	0.469	0.529
llama3	ens	0.762	**0.846**	**0.802**	0.483	0.406	0.441	0.543	0.388	0.452
	high	0.748	**0.846**	0.794	0.448	0.368	0.404	0.414	0.245	0.308
	low	0.774	0.791	0.783	0.494	0.387	0.434	0.462	0.367	0.409

5. Conclusion

In conclusion, the application of LLM embeddings, a simple classification layer, and a cross-validation strategy yielded optimal performance on this task. This outcome suggests the potential utility of these embeddings in complex text classification tasks such as irony and humor categorization.

Our findings demonstrate the importance of considering more nuanced factors beyond the mere recency of the model. LLama2 emerges as the superior model, outperforming LLama3 and Mistral across the majority of configurations. This indicates that factors beyond chronological advancement are influencing the efficacy of the models, thus answering RQ2.

Furthermore, the integration of QLoRA consistently enhanced performance, regardless of the base model. The incorporation of lower ranks and higher alpha values yielded unexpected yet consistent improvements, leading to further interrogation in regards to RQ3. Notably, issues persisted between IR

(Irony) and SC (Sarcasm). Qualitative analyses yielded intriguing insights, particularly regarding the impact of text related to the coronavirus on correctness. This was observed in IR, SC, and EX, which demonstrated sensitivity to such mentions. Fluctuations in correctness were observed, suggesting that different classes exhibited varying degrees of performance.

Further investigations into the different hidden layer variables revealed varying sensitivity among classes to the depth of analysis. In particular, WS (Wit and Surprise) and AID (Incongruity) demonstrated resilience, whereas others displayed sensitivity, partially answering RQ4.

In essence, our study highlights the multifaceted nature of language model performance on complex classification, underscoring the necessity for comprehensive evaluations encompassing both quantitative metrics and qualitative considerations to elucidate underlying mechanisms and optimize efficacy in text classification tasks. It also showed that the general knowledge embedded in LLMs facilitates more accurate classification of irony and humor genre, even though it is still far from being sufficient (RQ1).

6. Acknowledgments

Experiments presented in this paper were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and in cluding CNRS. RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

References

[1] C. Francesconi, C. Bosco, F. Poletto, M. Sanguinetti, Error analysis in a hate speech detection task: The case of haspeede-tw at evalita 2018, in: CEUR WORKSHOP PROCEEDINGS, volume 2481, CEUR-WS, 2019, pp. 1–6.

[2] G. Guibon, L. Ermakova, H. Seffih, A. Firsov, G. L. Noé-Bienvenu, Multilingual fake news detection with satire, in: International Conference on Computational Linguistics and Intelligent Text Processing, Springer, 2019, pp. 392–402.

[3] T. Miller, C. F. Hempelmann, I. Gurevych, Semeval-2017 task 7: Detection and interpretation of english puns, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 2017, pp. 58–68.

[4] O. Weller, K. Seppi, Humor detection: A transformer gets the last laugh, ArXiv abs/1909.00252 (2019).

[5] J. Devlin, M. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR abs/1810.04805 (2018). URL: http://arxiv.org/abs/1810.04805. arXiv:1810.04805.

[6] M. Wang, H. Yang, Y. Qin, S. Sun, Y. Deng, Unified humor detection based on sentence-pair augmentation and transfer learning, in: Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, 2020, pp. 53–59.

[7] Y. Ziser, E. Kravi, D. Carmel, Humor detection in product question answering systems, Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (2020).

[8] A. Reyes, P. Rosso, D. Buscaldi, From humor recognition to irony detection: The figurative language of social media, Data & Knowledge Engineering 74 (2012) 1–12. URL: https://www.sciencedirect.com/science/article/pii/S0169023X12000237. doi:https://doi.org/10.1016/j.datak.2012.02.005.

[9] C. V. Hee, E. Lefever, V. Hoste, Semeval-2018 task 3: Irony detection in english tweets, in: Proceedings of the 12th international workshop on semantic evaluation, 2018, pp. 39–50.

[10] S. Frenda, A. Pedrani, V. Basile, S. Lo, A. T. Cignarella, R. Panizzon, C. S. Marco, B. Scarlini, V. Patti, C. Bosco, D. Bernardi, Epic: Multi-perspective annotation of a corpus of irony, in: Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2023, pp. 13844–13857.

[11] N. K. Baym, The performance of humor in computer-mediated communication, Journal of computer-mediated communication 1 (1995) JCMC123.

[12] J. Morkes, H. K. Kernal, C. Nass, Effects of humor in task-oriented human-computer interaction and computer-mediated communication: A direct test of srct theory, Human-Computer Interaction 14 (1999) 395–435.

[13] H. Touvron, L. Martin, K. R. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. M. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. S. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. M. Kloumann, A. V. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, T. Scialom, Llama 2: Open foundation and fine-tuned chat models, ArXiv abs/2307.09288 (2023).

[14] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de Las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, W. E. Sayed, Mistral 7b, ArXiv abs/2310.06825 (2023).

[15] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, ArXiv abs/2005.14165 (2020).

[16] O. J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, R. Avila, I. Babuschkin, S. Balaji, V. Balcom, P. Baltescu, H. Bao, M. Bavarian, J. Belgum, I. Bello, J. Berdine, G. Bernadett-Shapiro, C. Berner, L. Bogdonoff, O. Boiko, M. Boyd, A.-L. Brakman, G. Brockman, T. Brooks, M. Brundage, K. Button, T. Cai, R. Campbell, A. Cann, B. Carey, C. Carlson, R. Carmichael, B. Chan, C. Chang, F. Chantzis, D. Chen, S. Chen, R. Chen, J. Chen, M. Chen, B. Chess, C. Cho, C. Chu, H. W. Chung, D. Cummings, J. Currier, Y. Dai, C. Decareaux, T. Degry, N. Deutsch, D. Deville, A. Dhar, D. Dohan, S. Dowling, S. Dunning, A. Ecoffet, A. Eleti, T. Eloundou, D. Farhi, L. Fedus, N. Felix, S. P. Fishman, J. Forte, I. Fulford, L. Gao, E. Georges, C. Gibson, V. Goel, T. Gogineni, G. Goh, R. Gontijo-Lopes, J. Gordon, M. Grafstein, S. Gray, R. Greene, J. Gross, S. S. Gu, Y. Guo, C. Hallacy, J. Han, J. Harris, Y. He, M. Heaton, J. Heidecke, C. Hesse, A. Hickey, W. Hickey, P. Hoeschele, B. Houghton, K. Hsu, S. Hu, X. Hu, J. Huizinga, S. Jain, S. Jain, J. Jang, A. Jiang, R. Jiang, H. Jin, D. Jin, S. Jomoto, B. Jonn, H. Jun, T. Kaftan, L. Kaiser, A. Kamali, I. Kanitscheider, N. S. Keskar, T. Khan, L. Kilpatrick, J. W. Kim, C. Kim, Y. Kim, H. Kirchner, J. R. Kiros, M. Knight, D. Kokotajlo, L. Kondraciuk, A. Kondrich, A. Konstantinidis, K. Kosic, G. Krueger, V. Kuo, M. Lampe, I. Lan, T. Lee, J. Leike, J. Leung, D. Levy, C. M. Li, R. Lim, M. Lin, S. Lin, M. Litwin, T. Lopez, R. Lowe, P. Lue, A. A. Makanju, K. Malfacini, S. Manning, T. Markov, Y. Markovski, B. Martin, K. Mayer, A. Mayne, B. McGrew, S. M. McKinney, C. McLeavey, P. McMillan, J. McNeil, D. Medina, A. Mehta, J. Menick, L. Metz, A. Mishchenko, P. Mishkin, V. Monaco, E. Morikawa, D. P. Mossing, T. Mu, M. Murati, O. Murk, D. M'ely, A. Nair, R. Nakano, R. Nayak, A. Neelakantan, R. Ngo, H. Noh, O. Long, C. O'Keefe, J. W. Pachocki, A. Paino, J. Palermo, A. Pantuliano, G. Parascandolo, J. Parish, E. Parparita, A. Passos, M. Pavlov, A. Peng, A. Perelman, F. de Avila Belbute Peres, M. Petrov, H. P. de Oliveira Pinto, M. Pokorny, M. Pokrass, V. H. Pong, T. Powell, A. Power, B. Power, E. Proehl, R. Puri, A. Radford, J. Rae, A. Ramesh, C. Raymond, F. Real, K. Rimbach, C. Ross, B. Rotsted, H. Roussez, N. Ryder, M. D. Saltarelli, T. Sanders, S. Santurkar, G. Sastry, H. Schmidt, D. Schnurr, J. Schulman, D. Selsam, K. Sheppard, T. Sherbakov, J. Shieh, S. Shoker, P. Shyam, S. Sidor, E. Sigler, M. Simens, J. Sitkin, K. Slama, I. Sohl, B. D. Sokolowsky, Y. Song, N. Staudacher, F. P. Such, N. Summers, I. Sutskever, J. Tang, N. A. Tezak, M. Thompson, P. Tillet, A. Tootoonchian, E. Tseng, P. Tuggle, N. Turley, J. Tworek, J. F. C. Uribe, A. Vallone, A. Vijayvergiya, C. Voss, C. L. Wainwright, J. J. Wang, A. Wang, B. Wang, J. Ward, J. Wei, C. Weinmann, A. Welihinda, P. Welinder, J. Weng, L. Weng, M. Wiethoff, D. Willner, C. Winter,

S. Wolrich, H. Wong, L. Workman, S. Wu, J. Wu, M. Wu, K. Xiao, T. Xu, S. Yoo, K. Yu, Q. Yuan, W. Zaremba, R. Zellers, C. Zhang, M. Zhang, S. Zhao, T. Zheng, J. Zhuang, W. Zhuk, B. Zoph, Gpt-4 technical report, arXiv preprint arXiv:2303.08774 (2023).

[17] AI@Meta, Llama 3 model card, 2024. URL: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[18] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. X. Song, J. Steinhardt, Measuring massive multitask language understanding, ArXiv abs/2009.03300 (2020).

[19] P. Clark, I. Cowhey, O. Etzioni, T. Khot, A. Sabharwal, C. Schoenick, O. Tafjord, Think you have solved question answering? try arc, the ai2 reasoning challenge, ArXiv abs/1803.05457 (2018).

[20] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, Y. Choi, Hellaswag: Can a machine really finish your sentence?, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 4791–4800. URL: https://aclanthology.org/P19-1472. doi:10.18653/v1/P19-1472.

[21] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, OpenAI blog (2019).

[22] L. Reynolds, K. McDonell, Prompt programming for large language models: Beyond the few-shot paradigm, Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems (2021).

[23] H. Cho, H. J. Kim, J. Kim, S.-W. Lee, S. goo Lee, K. M. Yoo, T. Kim, Prompt-augmented linear probing: Scaling beyond the limit of few-shot in-context learners, ArXiv abs/2212.10873 (2022).

[24] J. E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, W. Chen, Lora: Low-rank adaptation of large language models, ArXiv abs/2106.09685 (2021).

[25] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, Qlora: Efficient finetuning of quantized llms, ArXiv abs/2305.14314 (2023).

[26] L. Ermakova, T. Miller, A.-G. Bosser, V. M. P. Preciado, G. Sidorov, A. Jatowt, Overview of joker - clef-2024 track on automatic humor analysis, in: L. Goeuriot, P. Mulhem, G. Quénot, D. Schwab, L. Soulier, G. M. D. Nunzio, P. Galuščáková, A. G. S. de Herrera, G. Faggioli, N. Ferro (Eds.), Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fifteenth International Conference of the CLEF Association (CLEF 2024), 2024.

[27] L. Ermakova, A.-G. Bosser, T. Miller, T. Thomas-Young, V. M. P. Preciado, G. Sidorov, A. Jatowt, Clef 2024 joker lab: Automatic humour analysis, in: N. Goharian, N. Tonellotto, Y. He, A. Lipani, G. McDonald, C. Macdonald, I. Ounis (Eds.), Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, Proceedings, Part VI, volume 14613 of *Lecture Notes in Computer Science*, Springer, Cham, 2024, pp. 36–43. doi:10.1007/978-3-031-56072-9_5.

[28] M. Bouazizi, T. O. Ohtsuki, A pattern-based approach for sarcasm detection on twitter, IEEE Access 4 (2016) 5477–5488.

[29] H. Cramér, Mathematical Methods of Statistics (PMS-9), Volume 9, Princeton University Press, Princeton, 1946. URL: https://doi.org/10.1515/9781400883868. doi:doi:10.1515/9781400883868.

[30] B. W. Matthews, Comparison of the predicted and observed secondary structure of t4 phage lysozyme., Biochimica et biophysica acta 405 2 (1975) 442–51.

A. Shared Task results

Table 8
All results submitted for the shared task Joker number 2. All values are the F1-score. Our submission starts with a star: "* ORPAILLEUR_..."

Run ID	Macro	Weighted	SD	WS	EX	IR	SC	AID
* ORPAILLEUR_mistral-7b-ens	**0.700**	**0.749**	0.772	0.556	0.517	**0.742**	0.739	0.875
* ORPAILLEUR_mistral-7b-high	0.660	0.719	0.727	0.460	0.521	0.704	0.695	0.853
* ORPAILLEUR_llama2-7b-ens	0.659	0.721	0.798	0.494	0.365	0.676	0.722	**0.901**
* ORPAILLEUR_llama3-8b-ens	0.659	0.723	**0.802**	0.452	0.441	0.650	0.709	0.898
CYUT_llama3-fine-tuning	0.639	0.695	0.696	0.521	0.457	0.618	0.672	0.868
* ORPAILLEUR_llama2-7b-high	0.635	0.708	0.783	0.404	0.500	0.628	0.610	0.885
* ORPAILLEUR_llama3-8b-low	0.629	0.695	0.783	0.409	0.434	0.591	0.678	0.881
* ORPAILLEUR_llama2-7b-low	0.617	0.683	0.765	0.442	0.312	0.634	0.676	0.874
PunDerstand_DeBERTaSampled	0.616	0.677	0.777	0.500	0.424	0.515	0.600	0.880
* ORPAILLEUR_llama3-8b-high	0.614	0.691	0.794	0.308	0.404	0.611	0.685	0.885
PunDerstand_GuidedAnnotation	0.606	0.668	0.667	0.615	0.222	0.400	**0.900**	0.833
* ORPAILLEUR_mistral-7b-low	0.604	0.661	0.741	0.529	0.309	0.614	0.594	0.838
PunDerstand_DeBERTa	0.603	0.673	0.774	0.476	0.261	0.597	0.624	0.889
DadJokers_bert_base_uncased	0.601	0.669	0.750	0.395	0.402	0.588	0.618	0.851
NLPalma_BERTd	0.594	0.665	0.714	0.395	0.488	0.589	0.543	0.835
CodingRangers_bert_uncased	0.594	0.658	0.756	0.446	0.335	0.594	0.590	0.841
Code Rangers_roberta	0.590	0.659	0.472	**0.697**	**0.727**	0.027	0.737	0.881
Demonteam_BERTM	0.580	0.650	0.719	0.463	0.345	0.569	0.537	0.850
UAms_BERT_ft	0.522	0.602	0.755	0.418	0.054	0.589	0.484	0.832
NLPalma_PREDCNN	0.515	0.587	0.648	0.301	0.396	0.527	0.464	0.753
VayamSolveKurmaha_BERT	0.511	0.580	0.464	0.434	0.250	0.570	0.566	0.785
NaiveNeuron_fastText	0.507	0.584	0.683	0.367	0.411	0.462	0.343	0.776
NaiveNeuron_llama3:70b_rag-uae	0.501	0.534	0.459	0.471	0.409	0.426	0.571	0.671
VayamSolveKurmaha_BERT	0.500	0.571	0.568	0.426	0.256	0.517	0.458	0.777
NaiveNeuron_llama3:70b_rag	0.499	0.538	0.450	0.423	0.392	0.438	0.602	0.687
DadJokers_RandomForest_MLP_Ensemble	0.471	0.534	0.602	0.378	0.155	0.525	0.456	0.711
HumourInsights_Random Forest	0.449	0.516	0.633	0.421	0.197	0.455	0.289	0.702
PunDerstand_GPT4oFewShot	0.418	0.473	0.243	0.275	0.345	0.421	0.583	0.642
UBO_RubyAiYoungTeam	0.400	0.484	0.529	0.254	0.141	0.507	0.293	0.675
team1_Petra_and_Regina_LogisticRegression	0.400	0.484	0.529	0.254	0.141	0.507	0.293	0.675
Dajana&Kathy_Joker_LogisticRegression	0.400	0.484	0.529	0.254	0.141	0.507	0.293	0.675
team1_FRANE_AND_ANDREA_LogisticRegression	0.400	0.484	0.529	0.254	0.141	0.507	0.293	0.675
Tomislav&Rowan_SVM	0.381	0.468	0.374	0.381	0.178	0.460	0.198	0.693
AB&DPV_MLP3000params	0.375	0.442	0.495	0.300	0.109	0.434	0.296	0.618
DadJokers_RandomForest	0.374	0.458	0.566	0.320	0.054	0.470	0.174	0.662
CYUT_GPT-4	0.336	0.324	0.087	0.351	0.275	0.195	0.567	0.539
Tomislav&Rowan_LogisticRegression	0.309	0.410	0.331	0.179	0.094	0.412	0.187	0.651
AB&DPV_DecisionTreeClassifier	0.221	0.282	0.222	0.128	0.149	0.261	0.143	0.424
CYUT_roBERTa-fine-tuning	0.208	0.173	0.000	0.042	0.045	0.562	0.601	0.000
AB&DPV_RandomForestClassifier250	0.194	0.292	0.187	0.091	0.069	0.205	0.061	0.549
Tomislav&Rowan_NaiveBayes	0.189	0.319	0.178	0.000	0.000	0.352	0.000	0.602
AB&DPV_RandomForestClassifier500	0.187	0.286	0.152	0.125	0.052	0.219	0.032	0.544
AB&DPV_GaussianNB	0.173	0.245	0.000	0.206	0.103	0.047	0.169	0.515
AB&DPV_MLP2000	0.097	0.206	0.000	0.036	0.000	0.000	0.000	0.544
AB&DPV_MLP3000	0.097	0.206	0.000	0.036	0.000	0.000	0.000	0.544