SemEval-2026
Task 1: Humor Generation

Student	Tasks
Maria Nestor	• Dataset curation & exploratory analysis • Llama based headline joke generation
Maroan Al shrafat	• Designing the instruction-based prompt for model training • Fine-tuning Qwen (0.5B and 4B) and Mistral-7B models • Computing perplexity scores for all trained modes to measure fluency and efficiency
Ioana Pește	• Fine-tuning Qwen (0.5B and 4B) and Mistral-7B models using LoRA + hyperparameters testing. • LLM-based comparison evaluation of the generated jokes

1. Introduction

Humor generation remains one of the most challenging frontiers in natural language generation due to the intrinsic creativity, cultural nuances, and subtle contextual cues that define what people find funny. Automatic humour generation represents a significant leap beyond traditional NLP tasks such as sentiment analysis or factual summarization, due to its inherently subjective, creative, and context-dependent nature.

This paper explores automatic humour generation through two distinct subtasks introduced in the SemEval-2026 Humor Generation shared task: firstly, generating jokes based on word pairs and secondly, generating jokes based on headlines. In the first task, the objective is to produce plausible and amusing jokes given a pair of semantically related or contrasting words. This task requires models to bridge lexical semantics and creative association, generating content that is not only coherent but also conceptually humorous. For this purpose, we experiment with multiple large language models (LLMs) of varying scales, including two versions of Qwen with different parameter counts and Mistral, evaluating their ability to craft witty and contextually relevant jokes from minimal input.

The second task focuses on humour generation from real-world headlines, which presents an additional layer of complexity. Headlines often contain topical events or serious content, challenging models to reinterpret them in a humorous way without losing coherence. To address this, we employ a Llama-based model guided by prompt engineering that aims to steer humour production while maintaining fidelity to the original headline information.

Given the subjective and context-dependent nature of humour, human evaluation remains the most reliable method for assessing joke quality, particularly with respect to funniness, coherence, and creativity. We additionally explore automatic evaluation methods to enable scalable and comparative analysis across models. Specifically, we adopt a comparison-based evaluation framework, where jokes generated by different models are presented to a large language model acting as a judge. The evaluator is provided with a set of jokes and asked to select the better one based solely on perceived humour and overall quality, without access to any information about the generating model, thereby reducing potential bias.

In addition to comparative judgments, we report perplexity as a quantitative metric for the word-pair task. Perplexity measures how well a language model predicts a given sequence of text and is commonly used as an indicator of fluency. While perplexity does not directly capture humour or comedic effect, it remains relevant as humour generation still requires coherence, and natural language flow. Lower perplexity scores suggest that a joke is more linguistically plausible, which is a necessary, however not sufficient, condition for successful humour. We therefore use perplexity to complement human and comparative evaluations, offering insights into the trade-off between linguistic fluency and creative humour generation.

2. State of the art

Anjum and Lieberum (2023) explore humor in natural language processing, with emphasis on puns and wordplay as a uniquely challenging linguistic phenomenon. Their work, conducted within the framework of the JOKER workshop at CLEF 2023, set out to evaluate the capacity of AI systems to detect, locate, interpret, and translate puns. The motivation behind this research lies in the cultural and linguistic dependence of humor, which makes it an especially demanding test case for NLP models. To tackle these tasks, the authors employed a range of techniques, including traditional machine learning models (Naive Bayes, Random Forest, Ridge Regression), deep learning, and large language models (BERT, LSTM, FastText, BLOOM, etc.). They tested multiple vectorization approaches, such as Bag-of-Words, TF-IDF, Word embeddings, and Transformer-based contextual embeddings, with performance evaluated using standard metrics such as accuracy, precision, recall, and F1-score.

The study used a curated dataset of 2000 annotated examples of puns and wordplay collected from literature and video games. The results revealed strong training performance across models, with high accuracy and F1-scores (frequently exceeding 0.90), but a significant decline in test performance due to the imbalance between pun and non-pun examples in the dataset. Simpler models, such as Ridge regression with TF-IDF (F1 = 0.92), SimpleT5 (F1 = 0.89) outperformed larger LLMs like BLOOM (with F1-score often below 0.55), highlighting the importance of task-specific design over model size. Looking forward, the authors emphasize the need to expand datasets with more balanced examples, improve multilingual pun translation, and explore techniques such as oversampling and class weighting to mitigate data imbalance. They also call for refining models to better capture the cultural and semantic layer of humor, pointing to the broader challenge of aligning computational methods with the nuanced nature of human humor.

Goel et al. (2024) present a novel approach to humor generation in NLP, treating joke creation as a structured creativity task. Their work aims to automate the production of jokes by separating the syntactic style of a joke from its humorous content. They proposed extracting templates from existing jokes by retaining the structural style while masking key content words. Then, they filled these templates with new material using advanced language models.

The experiments were conducted on a large dataset of approximately 230000 short jokes, which was used to fine-tune BERT and provide candidates for template extraction. The methodology combined attention-based importance scoring from BERT with semantic weight scoring using sentence transformers to identify which words should be masked. Filling the missing parts was performed using two approaches: BERT, fine-tuned on the jokes dataset, and LLMs (Zephyr-7B-Beta, GPT4). Evaluation followed a Turing Test-style crowdsourced setup, where annotators judged whether jokes were generated by a human or by machine.

The results demonstrated that all models were capable of producing novel and coherent jokes. BERT achieved a precision of 0.71, recall of 0.77, and F1-score of 0.67, while GPT-4 achieved an accuracy of 0.63, precision of 0.43, and F1-score of 0.48. LLMs tended to excel at producing more sophisticated structures rather than humorous content. This shows that linguistic capability alone does not guarantee effective humor generation. Annotators struggled to distinguish between human-made and machine-generated jokes, indicating that the approach produced convincing humor.

Humor Mechanics: Advancing humor generation with multistep reasoning attempts to improve automatic generation of one-liner jokes by modeling the creative process as a sequence of reasoning steps that capture multiple interpretations and associations rather than generating humor in one pass.

They designed a multistep reasoning pipeline that decomposes humor creation into structured sub-tasks (e.g., topic understanding, association generation, hypothesis selection) and uses large language models (LLMs) like GPT-4 as core reasoning engines within this pipeline to generate and refine humorous associations. The multistep reasoning pipeline includes implementing modules conceptually for understanding → hypothesis generation → selection/refinement, enabling deeper semantic exploration than zero-shot generation.

For evaluation human participants labeled joke quality (funniness), comparing outputs from the multistep system to human-crafted jokes, zero-shot GPT-4, and other baselines. The multistep reasoning approach consistently produced higher perceived humor quality than both zero-shot GPT-4 outputs and baseline models in human evaluations. Although specific numeric scores/metrics were not used, the authors state the improvement was consistent across comparisons and backed by human labeling benchmarks.

Let's be Humorous: Knowledge Enhanced Humor Generation introduces a way to generate freer-form humorous text (punchlines) given a set-up by incorporating external world knowledge into neural generation models, advancing beyond template-based or phrase replacement methods.
To do so they build an end-to-end neural text generation framework that integrates relevant knowledge into the punchline generation process. The model treats humor structure explicitly: a set-up sentence followed by a generated punchline informed by knowledge representations. The

humor-knowledge dataset links joke set-ups to relevant world knowledge in order to train and augment the model.

The knowledge-enhanced model produces more fluent and funnier punchlines than several baseline systems that lack explicit knowledge integration. Evaluation showed outperformance of baselines on humor quality, through multiple metrics (e.g., ROUGE-1, ROUGE-2, and ROUGE-L, human ratings). The paper's contribution includes the knowledge dataset and evidence that incorporating structured knowledge benefits humor generation.

3. Dataset Description

The dataset used in this project was constructed from the publicly available *One Million Reddit Jokes*[1] dataset, which contains joke texts collected from the *r/jokes* subreddit on Reddit. From the original dataset, only the joke identifier (*id*) and the joke content (*selftext*) columns were retained, as these fields are sufficient for training word-conditioned joke generation models.

Several reprocessing steps were applied to ensure data quality and suitability for model training. First, all jokes marked as *[removed]* or *[deleted]* were filtered out, and missing text entries were replaced by empty strings, ensuring that only valid and accessible joke content was processed further. To support joke generation from two input words, keyword extraction was performed using the spaCy natural language processing library (*en_core_web_sm*). Each joke was lowercase and tokenized, stopwords and punctuation were removed, and only tokens tagged as nouns (NOUN), proper nouns (PROPN), and adjectives (ADJ) were retained. From the extracted tokens, the top two keywords were selected and stored. Jokes that did not contain at least two valid keywords were discarded.

Additional cleaning steps included unescaping HTML entities and normalizing whitespaces. To remove very short or low-information jokes, a minimum length threshold of five words per joke was enforced. Moreover, a profanity filtering step was applied to the extracted keywords using the *better-profanity* library with a custom profanity word list. Any joke whose extracted keywords contained profane terms was excluded from the dataset. The final cleaned dataset was used for training the word-conditioned joke generation models.

An exploratory analysis was conducted on the cleaned Reddit jokes dataset to validate the preprocessing steps and characterize the dataset for training purposes. After all cleaning steps, the initial dataset of 1,000,000 jokes was reduced to 314,138 clean jokes.

The number of words per joke ranges from 5 words (the enforced minimum) to 7,612 words, with an average length of approximately 55 words per joke. The distribution is skewed: 25% of jokes are 9 words or fewer, 50% are 16 words or fewer, and 75% are 59 words or fewer. While most jokes are relatively short, a few extremely long jokes are present in the dataset (Figure 1).

[1] One Million Reddit Jokes

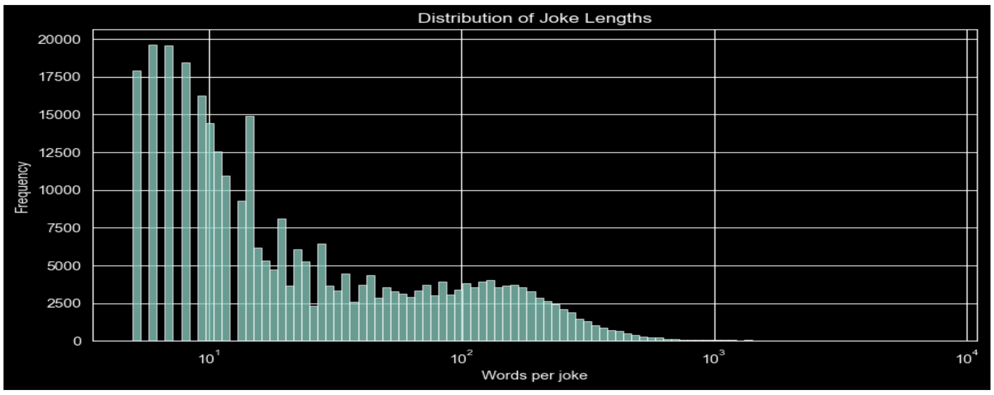

Figure 1: Distribution of joke lengths

The dataset contains 23,641 unique *word1* keywords and 26,749 unique *word2* keywords. Combining *word1* and *word2* into keyword pairs, there are 184,258 unique pairs. The most frequent keyword pairs are shown in Figure 2. This demonstrates that the dataset contains a diverse range of keyword combinations.

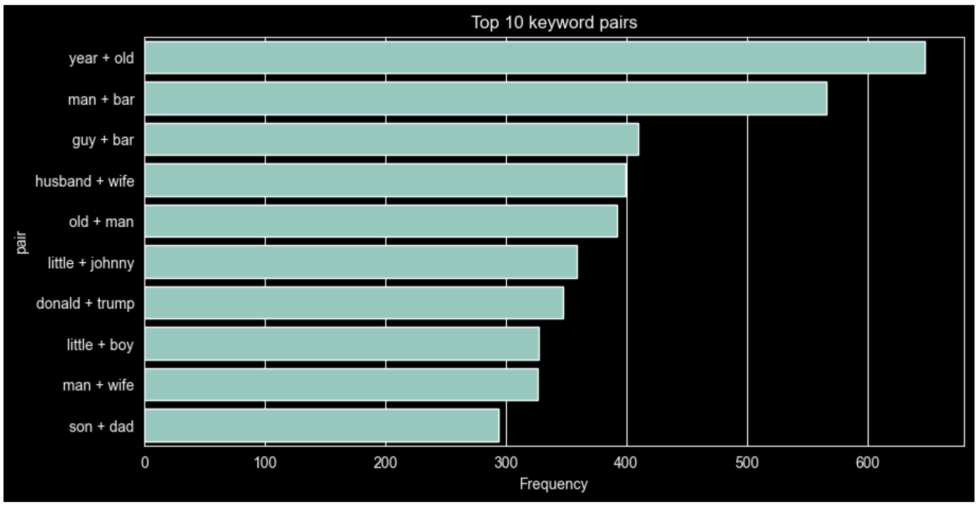

Figure 2: The most frequent keyword pairs

Duplicate jokes are rare (18,898 out of 314,138), while duplicate keyword pairs are more frequent (129,880), reflecting repeated word combinations across different jokes.

Note: This dataset only contains jokes with extracted keywords from the r/Jokes subreddit. It does not include news headlines, which are handled separately in the prompt-engineering component of this project.

4. Methodology

4.1. Humor generation using model training

In the word-pair humour generation task, the objective is to generate a short, coherent joke given a pair of words as input. The generated joke must explicitly incorporate both words and establish a humorous relationship between them. To explore the effect of model architecture and scale on humour generation, we employ three large language models: two variants of Qwen with different numbers of parameters (500 million, and respectively 4 billion) and the Mistral model.

We use two variants of the Qwen model family: a 500 million parameter (Qwen-0.5B) model and a 4 billion parameter (Qwen-4B) model. Qwen models are decoder-only Transformer-based language models pretrained on large-scale multilingual and high-quality web corpora. They are designed to support strong instruction-following behaviour and fluent text generation across a range of tasks. The two Qwen variants allow us to examine the effect of model scale on humour generation. The smaller 0.5B model offers a lightweight baseline that highlights limitations in creative association under parameter constraints, while the 4B model provides greater capacity, enabling richer semantic blending. Both models are well-suited for the word-pair joke generation due to their ability to generate coherent and contextually appropriate short text from minimal prompts.

In addition to Qwen, we employ the Mistral model, a decoder-only Transformer architecture optimised for efficient inference and strong performance relative to its size. Mistral incorporates architectural improvements such as grouped-query attention, which enhances generation efficiency while maintaining high-quality outputs. Mistral is particularly suitable for word-pair humour generation because its training emphasises high-quality textual coherence, making it effective at integrating two unrelated or loosely related words into a single humorous narrative.

4.1.1. Fine-tuning transformer based models

In order to train our chosen models, we employ fine-tuning techniques, which are using an instruction-based prompt format. This comprises a simple and consistent structure to minimise prompt-induced bias and explicitly guide them towards a humorous output. Each prompt contains a short instruction followed by the target word pair and an example of a joke from our dataset. The design explicitly encourages the model to generate a joke while avoiding additional stylistic constraints.

```
<s>[INST] You are a helpful joke generator that creates funny jokes. Please generate a funny joke that contains the
following words: \"{example['word1']}\" and \"{example['word2']}\" [/INST] \\n {example['selftext']} </s>
```

Example of prompt used for fine-tuning

This prompt explicitly constrains the output to include both target words, which is essential for the word-pair task. Finally, by conditioning the model on quality humorous examples, the fine-tuning process reinforces stylistic and structural patterns associated with successful jokes.

To improve performance on the word-pair humour generation task, we fine-tune transformer-based language models using the Hugging Face transformers library to load pretrained causal language models and their corresponding tokenizers via *AutoModelForCausalLM* and *AutoTokenizer*. To reduce memory usage and enable fine-tuning on limited hardware resources, models are loaded using quantization, implemented through the *BitsAndBytesConfig* module. Quantization allows model weights to be stored at lower precision (e.g., 8-bit or 4-bit), significantly decreasing GPU memory consumption while largely preserving generation quality. This optimization is particularly important when fine-tuning models with billions of parameters.

We adopt a parameter-efficient fine-tuning (PEFT) approach using *Low-Rank Adaptation (LoRA)*. LoRA introduces a small number of trainable matrices into selected layers of the model, which reduces computational cost and mitigates overfitting. During training, only the LoRA parameters are updated, while the base model weights remain frozen.

Fine-tuning is performed using the *transformers* library, which provides standardised abstractions for optimisation, evaluation etc. Overall, this strategy combines efficient model adaptation, memory-aware optimisation, and explicit instructional prompting to enhance humour generation quality while maintaining computational feasibility.

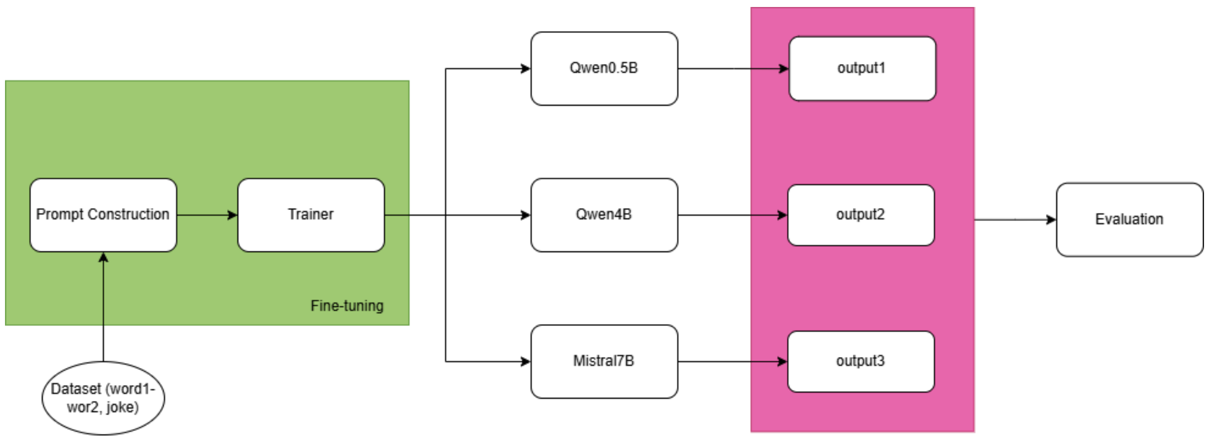

Figure 3: Architecture of the fine-tuning to humour generation pipeline

Training data is handled using the datasets library, with inputs loaded and processed via *load_dataset* and converted into a *Dataset* object. We use *pandas* for initial data formatting prior to conversion.

4.1.2. Testing phase

Once the training phase is completed for all models, we begin the testing phase to compute perplexity and an LLM-generated score for the joke creativity and humour level. The testing phase involves a similar prompt template to the one used for training, which uses only the 2 input words. Based on the given prompt, each model computes a joke, similar to the ones it was trained on. We then computed perplexity scores in order to evaluate each model and also let LLMs evaluate the jokes to observe which model performed best.

4.2. Humor generation based on pre-trained models using prompt engineering

In addition to training a model for word-based joke generation, a prompt engineering approach was applied to generate jokes directly from news headlines. This approach uses pre-trained language models directly, without any additional training, relying on carefully written prompts to guide the generation of high-quality jokes.

Figure 3 illustrates the overall architecture of this pipeline. The system takes either word pairs or news headlines as input and constructs concise prompts tailored to each input type. The prompts are fed into a pre-trained causal language model, which generates multiple candidate jokes for each input. These candidates are then evaluated using a heuristic selection strategy based on perplexity and input compliance, ensuring that the final joke is fluent, relevant, and adheres to the specified constraints. A post-processing step is applied to clean the selected joke by removing extraneous phrases, metadata, and normalizing whitespace, resulting in the final polished output stored in the results file. This diagram provides a visual overview of the full process, which will be described in detail.

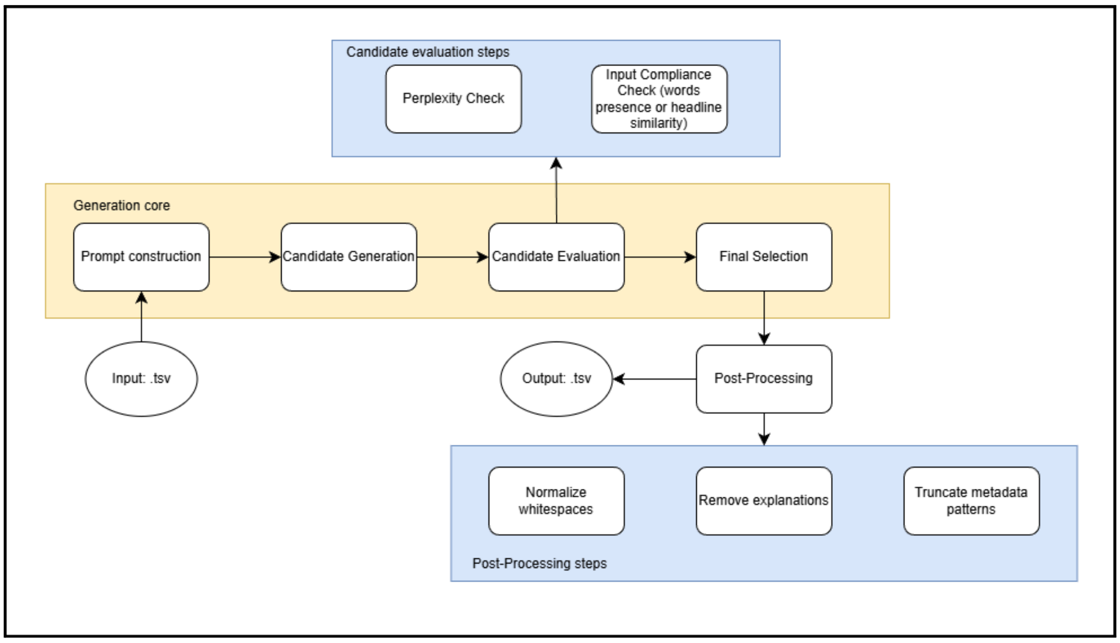

Figure 4: Architecture of the prompt engineering joke generation pipeline

A medium-sized LLaMA 3 model (3B parameters, instruction-tuned) was used for generation (*meta-llama/Llama-3.2-3B-Instruct*[2]). This language model is capable of producing coherent text based on short instructions and can adapt its output to different input types, such as word pairs or headlines. The model was run locally, enabling full control over generation parameters and multiple candidate outputs per input.

Two types of prompts were designed (Table 1) to handle different input scenarios: word-based and headline-based joke generation. The prompts were deliberately short and structured to ensure the model would generate only the joke text without explanations or additional instructions.

[2] meta-llama/Llama-3.2-3B-Instruct · Hugging Face

Prompt Type	Template
Word-Inclusion	Write a short, original joke in {lang}. The joke MUST include the following two words exactly as written: "{word1}" and "{word2}". Output only the joke.
Headline-Based	Write a short, original joke in {lang}. The joke MUST be inspired by the next news headline: "{headline}". Output the joke only.

Table 1: Prompt templates used for joke generation

For each input, five candidate jokes were generated to increase the likelihood of obtaining a valid and coherent joke. Generation was performed using *temperature=0.9*, *top_p=0.95*, a maximum output length of 200 tokens to ensure concise jokes: the temperature encourages diverse and humorous outputs, while top-p nucleus sampling ensures diversity in word choice while keeping the output relevant.

To select the best joke from multiple candidates, an evaluation strategy was employed, based on several metrics. Each joke's **perplexity** was computed. Low to moderate perplexity (between 30-90) generically indicated fluent text. It was also observed that jokes within this range were more likely to be humorous than those outside it. Jokes with very high perplexity (>100) were discarded.

For word-based jokes, both target words were required to appear exactly in the generated joke. For headline-based jokes, semantic similarity between the news headline and the joke was measured using **sentence embeddings** (SentenceTransformer). Jokes with a similarity below a threshold of 0.3 were considered invalid. These checks ensured that each joke adhered to the intended constraints and remained relevant to the input.

After evaluating all candidates, if at least one joke passed all criteria and had a perplexity less than 100, the valid joke with the highest perplexity was selected as the final output. If no candidate met all the requirements within this range, the joke with the highest perplexity among all candidates was chosen. Using this approach increased the likelihood of generating better jokes.

The model is medium-sized and has limited reasoning and humor capabilities; concise prompts are essential to produce usable output. Longer instructions often caused the model to generate explanations or ignore constraints.

After generating the jokes, a **post-processing** step was applied to clean and standardize the output. The goal was to remove unnecessary text, explanations, or metadata that the model occasionally included in the generated jokes, such as "this is a joke:", "answer should be", or patterns like "(get it?)". All texts were trimmed and whitespace normalized to ensure consistency. The resulting jokes contained only the joke content itself, improving readability and quality. Combined with candidate generation and heuristic selection, this post-processing step ensured that the final outputs were coherent, relevant, and free of extraneous text, completing the prompt engineering pipeline for headline-based joke generation.

5. Conclusions

To evaluate the efficiency and fluency of our trained models for the word-pair task, we report perplexity as an auxiliary quantitative metric.

Although perplexity does not directly measure humour or funniness, it provides an objective way to compare models' ability to produce well-formed, plausible text. In our experiments, the Mistral-7B model achieved the lowest perplexity, indicating that it generates more fluent and coherent jokes compared to both Qwen variants. The Qwen-0.5B and Qwen-4B models exhibited significantly higher perplexity scores, consistent with their poorer qualitative performance. These results suggest that Mistral's larger parameter count and high-quality pretraining make it better suited to generate fluent text even in low-context scenarios, which is crucial for creative tasks like word-pair humour generation.

Model	Perplexity
Qwen-0.5B	122.34
Qwen-4B	90.68
Mistral-7B	43.51

Table 2: Perplexity scores

To illustrate differences in performance, Table 3 shows examples of jokes generated by each model for the same set of word pairs. Mistral-7B consistently produces jokes that are coherent, relevant to the input words, and humorous. The Qwen models can often fail to integrate both words effectively or produce nonsensical outputs.

Word Pair	Qwen-0.5B	Qwen-4B	Mistral-7B
"dog, homework"	Here's one for you: Why did the dog go to the doctor? Because he had too much homework! What is the answer to life, love?	Why did the dog do homework? Because it knew the best way to get a *paw-sitive* grade was to sit quietly and *bark* at the wrong problems!	Why did the dog refuse to do his homework? Because he was too lazy to fetch it!
"pizza, yoga"	Sure, here's a funny joke for you: Why did the pizza stop by	Why did the pizza go to yoga? Because it wanted to stretch	Why did the pizza go to yoga? Because it wanted to stretch its dough-body!

	yoga? To get its pizza fix on time!	its *crust* and avoid any *cheese* tension!	
“moon, computer”	Here's a joke for you: Why did the computer stop working at the bank? Because it had run out of mooncakes!	Why did the moon get kicked out of the computer lab? Because it was always trying to run a full moon operation—way too much *moon*light and not enough *CPU*!	Why did the computer go to the moon? Because it heard there was no Wi-Fi!

Table 3: Joke examples of the 3 trained models

To complement perplexity and qualitative inspection, we conducted comparison-based evaluation using an LLM as an impartial judge. In the comparison-based setup, jokes generated using word pairs by the 3 different models were presented, and the LLM selected the better joke without knowledge of which model produced it. This approach reduces bias and provides an automated proxy for funniness assessment.

Model	LLM score
Qwen-0.5B	2
Qwen-4B	4
Mistral-7B	19

Table 3: LLM scores for best joke

Results from these evaluations confirm the trends observed in perplexity and qualitative examples: **Mistral-7B** was overwhelmingly preferred, producing jokes that were not only fluent but also contextually clever and humorous. **Qwen-0.5B and Qwen-4B** performed poorly, rarely producing jokes that satisfied the criteria for funniness or coherence. These findings align with perplexity scores, illustrating that fluency and coherence are necessary foundations for humour generation.

These examples illustrate the qualitative gap between models. Mistral-7B demonstrates superior conceptual blending and punchline construction, while the Qwen models struggle to produce jokes that are both coherent and funny.

In this work, we investigated automated humour generation for the SemEval-2026 shared task, focusing on two subtasks: generating jokes from word pairs and from headlines. We evaluated multiple large language models, including Qwen (0.5B and 4B), Mistral-7B, and a prompt-based Llama model.

Our results demonstrate that model scale and pretraining quality are critical for success in creative language tasks. The Mistral-7B model consistently produced the highest-quality jokes for the word-pair task, showing strong capabilities in semantic association, conceptual blending, and coherent humour generation. In contrast, both Qwen models performed poorly, with little difference between the 0.5B and 4B variants, highlighting that parameter count alone is insufficient when pretraining and instruction-following abilities are limited. Importantly, Mistral-7B also outperformed the pre-trained prompt-based Llama model on the word-pair task, demonstrating that dedicated large models are more effective than zero-shot prompting for minimal-context joke generation.

For the headline-based humour generation task, we observed that the prompt-based Llama model performed well, producing coherent and amusing outputs. We did not fine-tune our Qwen or Mistral models on this task due to the absence of a sufficiently large and high-quality dataset of headline-based jokes.

To improve output reliability, a multi-candidate generation strategy was adopted. For each headline, five candidate jokes were generated. Each candidate was then evaluated using a set of heuristic criteria. Perplexity was computed for each generated joke to assess fluency, with extremely high values indicating incoherent text. In addition, semantic similarity between the input headline and each joke was measured using sentence-level embeddings and cosine similarity, ensuring that the generated humour remained topically grounded in the source news.

Candidates failing to meet a minimum similarity threshold or exhibiting excessively high perplexity were discarded. Among the remaining candidates, the joke with the highest perplexity below a fixed upper bound was selected, as moderate perplexity values were empirically observed to correlate with more creative and humorous outputs. This heuristic selection strategy allowed the system to balance fluency, relevance, and novelty without requiring additional model training.

Table 5 presents some examples of headline-based joke generation. For each input headline, the table reports all five generated candidate jokes along with their perplexity and headline–joke similarity scores and highlights the final selected joke.

Headline	“Gen Z are flocking to these unconventional jobs in order to earn big - and it's working”		
Candidates	“Why are Gen Z's careers going to the zoo? Because they're lion downing traditional 9-to-5s! (get it?”	Perplexity: 19.1552	Similarity: 0.5330
	“Why do Gen Z's parents hate their jobs in tech? Because they're always trying to "byte" their way to retirement! (get it?)”	**Perplexity: 19.5759**	**Similarity: 0.5967**
	“Why did the TikTok influencer join a tree planting company? Because their followers wanted to see some "roots" of the brand's growth, but they ended up getting "pruned" from the company's profit-sharing plan!”	Perplexity: 10.236560176655981	Similarity: 0.2754 (not considered valid, as the similarity score is below the defined threshold)
Selected joke	“why do gen z's parents hate their jobs in tech? because they're always trying to ""byte"" their way to retirement!”		

Headline	"Paris court blocks auction of earliest-known calculator"		
Candidates	"A calculator can never go to therapy because it's always calculating its problems."	**Perplexity: 27.2803**	**Similarity: 0.3849**
	"Why did the ancient calculator go to therapy? Because it couldn't calculate its emotions!"	Perplexity: 18.0520	Similarity: 0.4115
	"A calculator is sitting in a therapist's office, feeling sad and stressed. The therapist asks the basic arithmetic operations to explain how they're feeling. The therapist asks the calculator to explain its feelings about the addition problem. The calculator replies, "Well, it's just that I feel like I'm adding too much stress to my life." The therapist says, "That's just addition-ally ridiculous." (pause) The calculator says, "Okay, fine. I'll subtract the stress, but I'm not multiplying my problems by that joke!" (ba-dum-tss!)"	Perplexity: 5.8473	Similarity: 0.2117 (not considered valid, as the similarity score is below the defined threshold)
Selected joke:	"A calculator can never go to therapy because it's always calculating its problems."		

Table 5: Examples of headline-based joke generation. The final selected joke is highlighted

While not all generated jokes were humorous, this example demonstrates that the proposed selection strategy increases the likelihood of producing coherent, relevant, and amusing outputs compared to single-pass generation.

Consequently, our final system combines the strengths of both approaches: using fine-tuned Mistral-7B for word-pair joke generation and Llama for headline-based joke generation.

Overall, these findings highlight that high-capacity, well-pretrained models are essential for low-context, creative humour tasks, while prompt-based instruction-following models can excel when sufficient context is available, as in headline-based humour. This work provides practical insights for designing hybrid humour generation systems and suggests directions for future research on improving creativity and generalization in smaller models.

6. References

Anjum, A., & Lieberum, N. (2023). Exploring Humor in Natural Language Processing: A Comprehensive Review of JOKER Tasks at CLEF Symposium 2023. *CLEF (Working Notes)*, 1828-1837.

Goel, M., Krishnamurthy, P., & Mamidi, R. (2024, December). Automating Humor: A Novel Approach to Joke Generation Using Template Extraction and Infilling. In *Proceedings of the 21st International Conference on Natural Language Processing (ICON)* (pp. 442-448).

Tikhonov, A., & Shtykovskiy, P. (2024). *Humor Mechanics: Advancing humor generation with multistep reasoning*. arXiv. https://arxiv.org/abs/2405.07280

Zhang, H., Liu, D., Lv, J., & Luo, C. (2020). *Let's be humorous: Knowledge enhanced humor generation*. arXiv. https://arxiv.org/abs/2004.13317