Meta-Prompting for Hybrid Fairy Tales

Comparing Prompt Engineering Strategies Across LLMs

Loris Palmarin
Master's in Data Science for Economics
University of Milan

August 7, 2025

Contents

1 Introduction

1.1 Motivation

In recent years, large language models (LLMs) have revolutionized the field of natural language processing, particularly in creative and generative applications. A key factor in unlocking the full potential of these models lies in how tasks are formulated, namely through prompts. That's the reason why prompt engineering has become a crucial area of study, especially in contexts where creativity, coherence, and stylistic control are essential.

This project was developed within the framework of the course *Natural Language Processing*, taught by Prof. Alfio Ferrara as part of the Master's program in Data Science. The course emphasizes both theoretical and applied aspects of NLP, and this project aims to investigate advanced prompting strategies in a generative setting.

1.2 Project Objective

The core objective of the project is to explore and compare the performance of different prompt engineering strategies when applied to a generative task. Specifically, the model is asked to generate a fairy tale blending two contrasting genres given by the user.

This approach allows for a complete analysis of how different prompt designs influence the structure, style, and quality of the generated stories and how those prompts affect downstream generations. The comparison includes multiple language models, both open and proprietary, such as GPT, LLaMA, and Mistral.

1.3 Task Definition: Hybrid Fairy Tale Generation

The generation task selected for this project is highly creative: producing short fairy tales that intentionally blend *contrasting narrative styles*, such as horror and comedy, cyberpunk and folklore, or noir and fantasy. The prompts are crafted with the specific goal of guiding a LLM to produce such hybrid stories.

This task was chosen because it tests the ability of LLMs to reason abstractly, follow complex stylistic instructions, and creatively merge incompatible genres, making it ideal for highlighting the strengths and weaknesses of different prompting techniques. Moreover, it provides a rich space for both qualitative evaluation (coherence, creativity, style) and interpretability analysis (e.g., XAI methods).

2 Prompting Methodology

2.1 Meta-Prompt Techniques

To explore the impact of prompt engineering strategies on story generation, I selected a diverse set of six prompting techniques. These were chosen to span a range of instructional designs, from minimal and general to emotionally rich and stylistically constrained. A complete and structured guide to identify the most relevant and useful ones for this scope was found in *The Prompt Report* by Schulhoff et al. (2024).

Our aim is to compare how each technique influences the quality and characteristics of the prompts generated by large language models (LLMs) for the task of creative story generation.

The six prompting techniques used in this project are:

- **Zero-Shot Prompting:** A minimal prompt is provided, with no context, role framing, or examples. This serves as a baseline for comparison.
- **Role Prompting:** The model is instructed to take on the role of a professional story writer, simulating expertise and structured thinking.
- **Few-Shot Prompting:** The prompt includes two examples of high-quality plots for similar tasks, allowing the model to generalize structure and intent.
- **Style Prompting:** The instruction explicitly requires the model to generate a story using a specific narrative or stylistic tone (e.g., concise, poetic, academic).
- **Emotion + Zero-Shot Prompting:** This variation maintains a zero-shot structure but embeds emotional and evocative language, drawing on insights from *Li et al. (2023)* on the power of emotional stimuli to enhance LLM performance.
- **Emotion + Role Prompting:** A hybrid technique combining the role-based structure with emotionally charged language to test the synergy between structured authority and affective framing.

2.2 Prompt Design Rationale

Each prompt was designed to generate a short tale written in a hybrid narrative style, a story that merges two contrasting genres, in this case **crime** and **fantasy**.

The design rationale behind selecting these six techniques is to enable a multi-dimensional comparison:

- *Minimal vs Structured*: zero-shot prompts test default behavior, while role- and few-shot prompts introduce structural cues.
- *Content vs Style Control*: style prompting examines how formulation affects prompt effectiveness beyond informational content.
- *Neutral vs Emotional Framing*: by embedding emotionally rich language, we assess how affective signals modulate story generation, inspired by findings in [2].
- *Hybrid Techniques*: emotion + role prompting serves to test if combined strategies yield synergistic or conflicting effects.

It should be noted that recent work by *Zheng et al. (2024)* questions the effectiveness of role-based prompting in improving LLM performance on factual tasks. Their large-scale study tested 162 personas in multiple open source models and more than 2000 multiple choice questions, concluding that system prompts to define a specific role ('You are a helpful assistant') had no consistent positive impact and in some cases even slightly reduced accuracy. While this result is important for tasks focused on objective correctness, we hypothesize that role and emotional prompting may behave differently in creative generation settings, where aspects like tone, stylistic coherence, and narrative immersion are more relevant than factual precision. Therefore, in this project we deliberately include role-based and emotionally framed strategies to assess their impact not on correctness, but on the richness and creativity of storytelling tasks.

This setup enables us to examine how different kinds of instructions affect generated story's clarity, specificity, creativity.

3 Models, Experimental Setup, and Access

3.1 Model Selection Rationale

The goal behind model selection was to span a broad spectrum of characteristics: (i) parameter scale (from one to several billions), (ii) training regime (base vs. instruction-tuned) and (iii) license and availability (fully open checkpoints vs. proprietary APIs). This diversity enables us to observe how prompt-engineering strategies behave across models with different inductive biases and levels of instruction-following.

The four core models used are:

- **Falcon 3-1B Instruct.** A 1-billion-parameter, instruction-tuned Falcon variant. Acts as a lightweight open-weights baseline for probing how far prompt design alone can steer a very small model.
- **LLaMA 3.2-3B Instruct.** A 3-billion-parameter model with supervised instruction tuning. Balances modest capacity with solid adherence to structured prompts, ideal for evaluating the "small-but-smart" trade-off.
- **Mistral 7B Instruct.** A mid-size (7 B) model with RLHF-style tuning. Provides more head-room than 3 B models while remaining fully open source, letting us test whether extra parameters improve response to complex creative instructions.
- **GPT-3.5-turbo (OpenAI).** A proprietary, 175 B-parameter chat model accessed via the OpenAI API. Included as a commercial benchmark, even though token-level log-probs are not exposed.

3.2 Technical Details of the Models

Table 1 summarizes the main technical attributes.

Table 1: Summary of the four language models employed.

Model	Params	Type	Checkpoint / API	Tokenizer	Context	License
Falcon 3-1B Instruct	1B	Instruct (SFT)	`tiiuae/Falcon3-1B-Instruct`	SentencePiece	8k	Apache 2.0
LLaMA 3.2 3B Instruct	~3B	Instruct	`meta-llama/Llama-3.2-3B-Instruct`	SentencePiece	8k	Research (Meta)
Mistral 7B Instruct	7B	Instruct (RLHF)	`mistralai/Mistral-7B-Instruct-v0.2`	SentencePiece	32k	Apache 2.0
GPT-3.5-turbo	~175B[1]	Chat model (RLHF)	OpenAI Chat Completions API	Tiktoken (BPE)	16k	Proprietary

3.3 Execution Environment

Software and Tooling. Core libraries comprise `transformers` (Hugging Face) for model/tokenizer loading and text generation, `torch` for GPU execution, `pandas` for tabular logging (logprobs, saliency scores), and `matplotlib` for visualisation.

To keep the experimental loop clean and reproducible, we factored auxiliary functions into a dedicated `utils.py` module.

Reproducibility. A global random seed (e.g., `torch.manual_seed(42)`) was set.

[1] OpenAI does not disclose exact parameter counts; estimates place the "turbo" series in the 170–180 B range.

3.4 Decoding Parameters

To maintain fairness, identical decoding parameters were applied across models (unless otherwise noted). The chosen configuration balances creativity and coherence:

Table 2: Decoding parameters used for generation.

Parameter	**Value**	**Rationale**
max_new_tokens	800	Concise, comparable outputs
temperature	0.7	Controlled diversity
top_p	0.95	Filter tail probabilities
top_k	50	Avoid extremely unlikely tokens
no_repeat_ngram_size	3	Prevent loops / duplicates
repetition_penalty	1.2	Penalize verbatim re-generation

3.5 Generation and Logging Pipeline

The experimental pipeline is structured into different stages. A compact code excerpt is reported in Appendix B, the full implementation is available in the project repository.

3.6 Model Access

Loading the open models. Falcon, LLaMA, and Mistral are fetched directly from the Hugging Face Hub with the usual one-liners: `AutoModelForCausalLM.from_pretrained(...)` and `AutoTokenizer.from_pretrained(...)`. No other setup is needed once the licence has been accepted on the Hub.

Accessing the closed model. GPT-3.5-turbo is queried through the OpenAI REST API. A single environment variable (`OPENAI_API_KEY`) is all that is required; the decoding parameters are set to match those used for the local models. Token-level log-probs are requested with the flag `logprobs=true` whenever the endpoint supports it.

4 Dataset and Annotation

4.1 Evaluation Criteria

Each generated fairy-tale is evaluated along six qualitative dimensions, scored on a 1–5 scale (1 = very poor, 5 = excellent). Table 3 gives the rubric used by the annotator.

Table 3: Annotation rubric (score 5 = fully satisfied).

Criterion	Guiding questions / cues
Adherence	Are *both* requested genres present and visibly blended? Is the specified tone respected?
Creativity	Novel characters, inventive world-building, unexpected twists, or clever genre fusion?
Structure	Logical event flow, clear beginning–middle–end, smooth transitions between genres?
Style	Does the lexical tone match the target genres (e.g. dark vocabulary for Gothic, light irony for Comedy)?
Emotional Impact	Does the story evoke emotions (fear, wonder, tenderness) and sustain them?
Prompt Sensitivity	Is the output clearly shaped by the prompt? (Large differences across prompts; stable replication with same prompt.)

4.2 Overview of the Generated Dataset

- **Instances.** 24 stories ((3 open-weight models + 1 closed) × 6 prompting techniques).
- **Columns.** `model`, `technique`, `prompt`, `generation`, six score columns (`fidelity`, `creativity`, ..., `sensitivity`), and an aggregate `total_score`.

4.3 Qualitative anomalies observed in the corpus

Manual review of the 24 stories (see `annotation.xlsx`) revealed four recurring failure patterns that strongly affect the quality scores.

1. **Prompt mirroring.** LLaMA and Falcon often copy the entire user prompt, add a missing full stop, then print `|assistant|`. Some instances continue with the real story, but the duplicated prompt text lowers *Fidelity* and *Coherence* to 1 or 2.
2. **Prompt continuation.** Some models expands the instruction ("Combine crime + fantasy...") instead of producing a tale, harming *Prompt Sensitivity.*
3. **Premature truncation.** Others hit `max_new_tokens` (800) and stops mid-sentence, yielding low scores in *Structure* and *Emotional Impact.*

Why GPT-3.5 avoids these issues. The OpenAI API probably appends an end-of-prompt marker and handles truncation. As a result, GPT-3.5 never mirrors the prompt and always returns a full narrative, scoring at least 3 on every rubric dimension. The only problem left is the reach of max new tokens.

5 Confidence Metric Analysis

5.1 Token–Level Confidence Indicators

Modern LLMs expose the full probability distribution of the next token. From that distribution we derive three compact statistics that quantify, in different ways, how *confident* the model is at each decoding step:

1. **Surprisal**
$$s_t \;=\; -\log p\big(x_t \mid x_{<t}\big)$$
is the negative log–probability of the token actually chosen at position t. Lower values indicate that the model found the token highly predictable, hence was more certain about it.

2. **Entropy** of the next–token distribution
$$H_t \;=\; -\sum_i p_i \log p_i$$
captures the *spread* of probability mass over the vocabulary. High entropy means the model considered many alternatives almost equally plausible; low entropy means a peaked, decisive distribution.

3. **Margin**
$$m_t \;=\; p_{\text{top1}} - p_{\text{top2}}$$
measures the gap between the probability assigned to the most likely token and the runner-up. A large margin implies the model clearly preferred the emitted token over any competitor.

5.2 Full numerical reference

All averaged figures reported in this chapter (*mean surprisal, mean entropy, mean margin*) are computed from the values in Table 4 below.[2]

5.3 Which prompt works best for *each* model?

From Table 4 the lowest–surprisal line per model is:

Model	**Best technique**	$\bar{s}$	$\bar{H}$	$\bar{m}$
Mistral–7B	Role Prompting	0.470	0.460	0.713
LLaMA–3B	Style Prompting	0.544	0.572	0.668
Falcon–1B	Emotion + Role Prompting	1.074	1.059	0.508

Mistral-7B performs best with a *Role* frame, achieving the lowest surprisal and the widest margin; the persona cue evidently helps the larger model focus.

LLaMA-3B prefers an explicit *Style* directive—its confidence peaks when the prompt tells it how to write rather than who it is.

[2] GPT–3.5-turbo is omitted because the model does not expose token–level probabilities; its confidence was discussed qualitatively in Chapter 4.

Table 4: Summary metrics (mean over the first four sentences) for every *model–technique* pair. Blank cells correspond to GPT-3.5-turbo, which does not expose token-level probabilities.

Model	**Technique**	**Surprisal μ**	**Entropy μ**	**Margin μ**
GPT-3.5	Emotion + Role Prompting	—	—	—
GPT-3.5	Few-Shot	—	—	—
GPT-3.5	Zero-Shot	—	—	—
GPT-3.5	Role Prompting	—	—	—
GPT-3.5	Emotion + Zero-Shot	—	—	—
GPT-3.5	Style Prompting	—	—	—
LLaMA–3B	Emotion + Role Prompting	0.983	0.864	0.535
LLaMA–3B	Few-Shot	1.194	1.220	0.482
LLaMA–3B	Zero-Shot	0.790	0.783	0.585
LLaMA–3B	Role Prompting	1.143	1.131	0.497
LLaMA–3B	Emotion + Zero-Shot	0.783	0.794	0.589
LLaMA–3B	Style Prompting	0.544	0.572	0.668
Falcon–3B	Emotion + Role Prompting	1.074	1.059	0.508
Falcon–3B	Few-Shot	1.398	1.357	0.443
Falcon–3B	Zero-Shot	1.137	1.145	0.462
Falcon–3B	Role Prompting	1.152	1.220	0.457
Falcon–3B	Emotion + Zero-Shot	1.376	1.420	0.409
Falcon–3B	Style Prompting	1.242	1.247	0.448
Mistral–7B	Emotion + Role Prompting	0.763	0.740	0.617
Mistral–7B	Few-Shot	0.914	0.843	0.571
Mistral–7B	Zero-Shot	0.634	0.653	0.657
Mistral–7B	Role Prompting	0.470	0.460	0.713
Mistral–7B	Emotion + Zero-Shot	0.832	0.824	0.588
Mistral–7B	Style Prompting	0.761	0.746	0.611

Falcon-1B gains most from the combined *Emotion + Role* wording, yet its numbers remain well behind the other two, illustrating how limited capacity restricts decisiveness even under the "best" prompt.

In short, prompt engineering *must* be tailored to the model: one size does not fit all, and the gap between 7 B and 1 B parameters is larger than any prompt trick.

5.4 Which model is best on average?

Averaging the three confidence metrics over *all* prompts yields the ranking:

Model	$\bar{s}\downarrow$	$\bar{H}\downarrow$	$\bar{m}\uparrow$
Mistral–7B	**0.73**	**0.71**	**0.63**
LLaMA–3B	0.91	0.89	0.56
Falcon–1B	1.23	1.24	0.45

Mistral remains the most decisive generator even before prompt optimisation, while Falcon lags behind both in certainty (surprisal/entropy) and decisiveness (margin).

5.5 Which technique is most robust across models?

Macro–averaging each technique over the three open models gives:

Technique	$\bar{s}\downarrow$	$\bar{H}\downarrow$	$\bar{m}\uparrow$
Style Prompting	**0.85**	**0.85**	0.58
Zero–Shot	0.85	0.86	0.57
Role Prompting	0.92	0.94	0.56
Emotion + Role	0.94	0.89	0.55
Emotion + Zero–Shot	1.00	1.01	0.53
Few–Shot	1.17	1.14	0.50

5.6 Key take-aways before diving into saliency

1. **Capacity beats clever prompting.** Mistral–7B, even in plain Zero-Shot, is more confident (lower $\bar{s}$, $\bar{H}$; higher $\bar{m}$) than Falcon–1B under its best prompt. Model size and training quality dominate token-level uncertainty.

2. **Style directives are the safest investment.** A single "write it like a fairy-tale" instruction cuts surprisal and entropy on all open models without sacrificing margin, making *Style Prompting* the best all-round strategy.

3. **Examples can hurt.** Injecting full Few-Shot examples raises surprisal by ∼30 % and lowers margin; the extra context seems to add noise rather than guidance for this creative, hybrid task.

4. **Roles & emotions help only the strong.** Persona framing and affective adjectives give a measurable boost to Mistral–7B, a modest one to LLaMA–3B, and almost none to Falcon–1B—suggesting that weaker models cannot reliably exploit high-level stylistic cues.

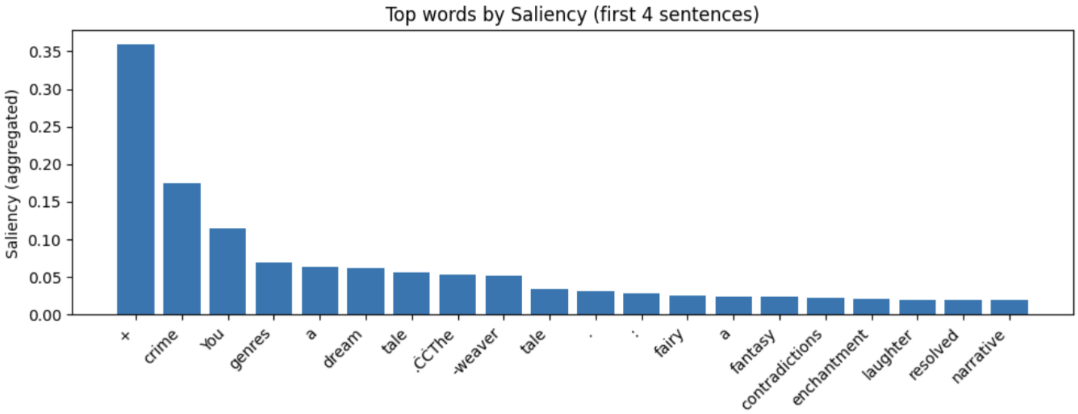
5. **Margin exposes hidden fragility.** Smaller models not only make less probable predictions, they are also *less certain* of them (margin ≈ 0.45 for Falcon vs. > 0.6 for Mistral), hinting at brittle internal decision boundaries.

These quantitative signals raise an immediate question:

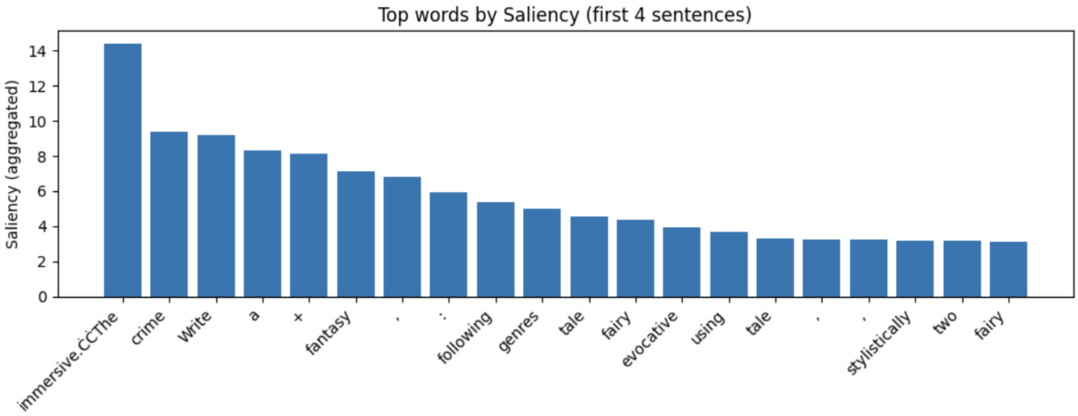
- *Which* ***parts*** *of the prompt make Mistral latch on to a role cue while LLaMA prefers explicit style markers?*

To answer this, we now turn to **word-level saliency**: gradient attribution for the open-weight models and leave-one-out importance for GPT-3.5-turbo (Chapter 6).

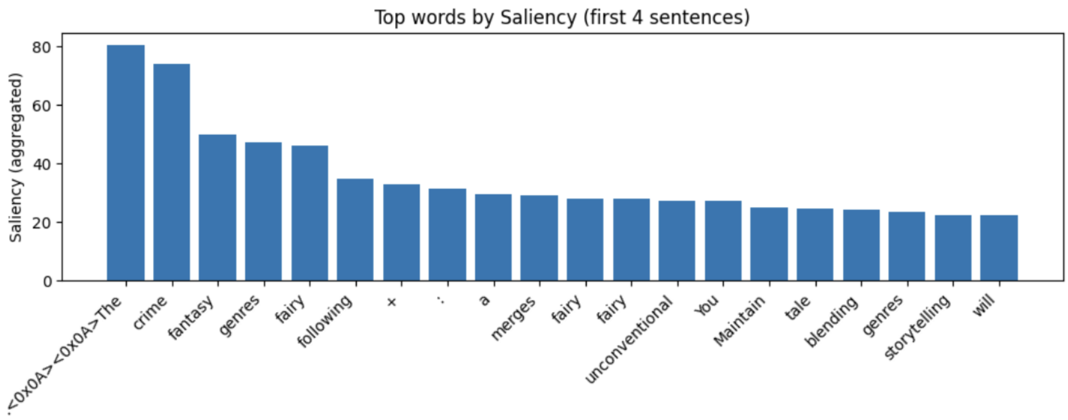
6 Saliency Analysis



(a) Falcon–1B — *Emotion + Role Prompting*



(b) LLaMA–3B — *Style Prompting*



(c) Mistral–7B — *Role Prompting*

Figure 1: Top-10 most influential prompt tokens (gradient saliency) for the best-performing prompt of each open-weight model.

6.1 What we mean by *saliency* and how we compute it

Saliency answers the question *"Which prompt tokens most influence the model's next words?"*. For open-weight models we use first-order **gradient attribution**:

$$\text{Saliency}(w_i) \;=\; \left\|\nabla_{\,\text{emb}(w_i)}\; \log P(y_{1:k} \mid \mathbf{w})\right\|_1,$$

i.e. the L_1 norm of the gradient of the log-likelihood of the first $k = 4$ sentences with respect to the embedding of token w_i. Large norms mean a small change in that embedding would strongly affect the logits, so the token is influential.

For GPT-3.5, which hides gradients, we approximate importance via **leave-one-out** (LOO):

$$\Delta_i = \log P\big(y_{1:k} \mid \mathbf{w}\big) \;-\; \log P\big(y_{1:k} \mid \mathbf{w}\backslash w_i\big).$$

Removing a crucial token makes the answer far less likely, yielding a large positive Δ_i.

To compare prompts of different lengths we normalise each score by the sum of absolute scores in the same prompt, so that normalised saliencies add up to 1. The Top-10 tokens and their shares are visualised in the next figures.

6.2 Top-10 token saliency for the open models

Mistral-7B concentrates almost half of its saliency on the role words *crime* and *fantasy*, clearly understanding the two genres to focus on for tale generation and.

LLaMA-3B is still focused on genres, but clearly emphasizes its task **write** and style needed, highest value is reached by **immersive** token(*Style Prompting*).

Falcon-1B under-weights most of the adjectives: its highest-scoring token is not a semantic word but the plus sign "+" in the phrase crime + fantasy. The small model latches onto the formatting symbol that separates the two genres instead of the genre words themselves, so the prompt's core instruction is only weakly internalised—hence the higher surprisal and narrow margin compared with Mistral and LLaMA..

6.3 GPT–3.5-turbo: leave-one-out saliency

Unlike the open-weight models, GPT–3.5 does not expose next-token log-probs. We therefore approximate token importance with *Leave–One–Out* (LOO).

Given a prompt $\mathbf{w} = (w_1, \ldots, w_n)$ and the model's answer $Y = y_{1:k}$, the *Leave–One–Out importance* of token w_i is

$$\Delta_i = \;\log P(Y \mid \mathbf{w}) \;-\; \log P\big(Y \mid \mathbf{w} \setminus w_i\big),$$

i.e. the drop in log-likelihood of the *entire* answer when that single token is removed from the prompt.[3] A large positive Δ_i means the token is crucial: without it the model's preferred answer becomes much less probable. Values can be negative when a token is distracting (its removal makes Y *more* likely).

Normalisation. To compare prompts of different length we report $\tilde{\Delta}_i = \Delta_i / \sum_j |\Delta_j|$, so that scores add up to 1 within each prompt.

We note that:

[3]For chat models we re-submit the shortened prompt via the API with identical decoding parameters and sum the returned token log-probs.

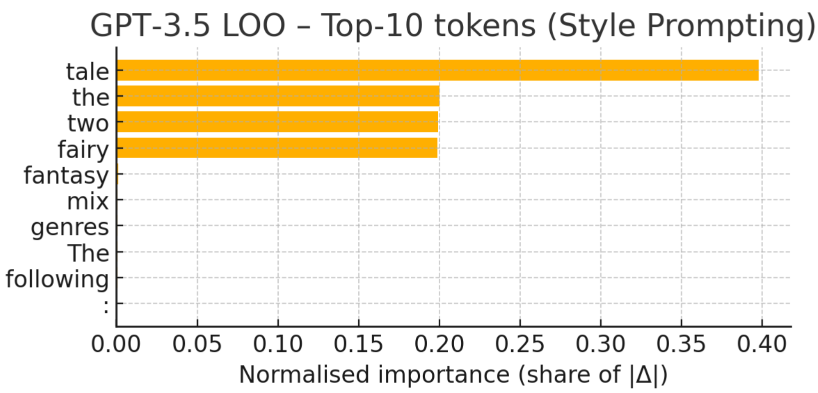
- **Style Prompting dominates.** The stylistic keyword *tale* plus the two genres *crime, fantasy* account for $\approx 50\,\%$ of total importance (Fig. 2a), signalling that the model truly leverages the style directive.

- **Role Prompting is weaker.** Importance is split between generic header tokens (*The, mix*) and the genres; role words themselves rank lower (Fig. 2b), explaining the smaller confidence gain compared to Style.

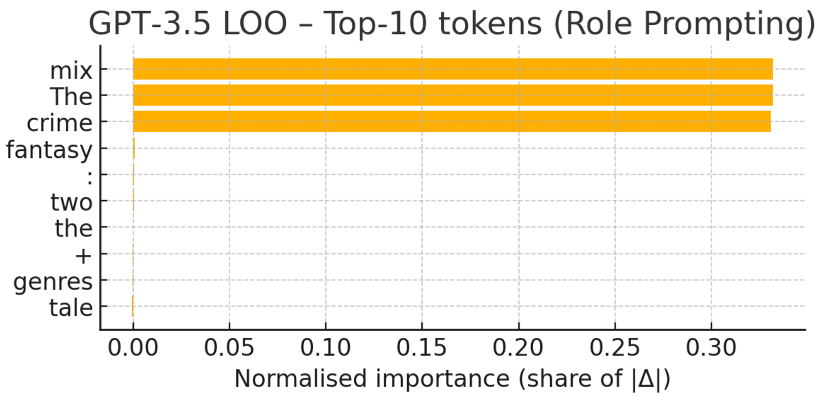
- **Few-Shot is mostly ignored.** The top tokens belong to the literal header of the first example, while genre words are less salient (Fig. 2c); the examples add lexical clutter rather than guidance.

- **Emotion variants fail.** Emotion adjectives never appear in the Top-10; a formatting token (+ or :) often absorbs the mass (plots in Appendix B), confirming that emotional framing merely dilutes useful signal.

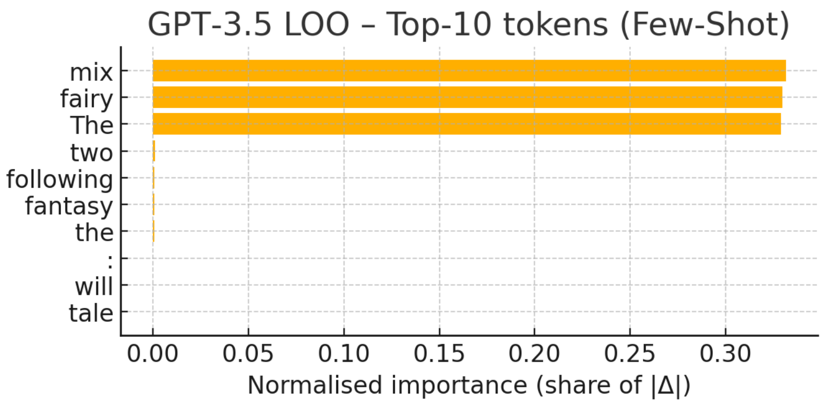
Ranking the six techniques by the total normalised importance captured by their Top-10 tokens yields *Style* > *Role* $\simeq$ *Few-Shot* > *Emotion* > *Zero-Shot*. Style Prompting is therefore the only prompt that GPT-3.5 both notices and exploits consistently, aligning with the pattern observed for the instruction-tuned LLaMA. All other strategies either scatter attention (Few-Shot) or focus on non-semantic tokens (Role), providing little control over the creative generation task.



(a) Style Prompting (winner)



(b) Role Prompting



(c) Few-Shot

Figure 2: GPT-3.5 — normalised LOO importance for the three most informative prompts (Top-10 tokens).

6.4 Cross–model synthesis

- **Genre tokens dominate in the stronger open models.** Both Mistral and LLaMA place their largest share of importance on the word *crime*, one of the two requested genres. This matches their ability to deliver low surprisal while respecting

[4] The plus sign comes from the prompt string "`crime + fantasy`" and receives the highest gradient score in Falcon—evidence that the small model is sensitive to layout rather than semantics.

Table 5: Token with highest *normalised* saliency (open models) or LOO–importance (GPT-3.5) for the best prompt of each model.

Model	**Top token**	**Cue type**
Mistral–7B	*crime*	GENRE
LLaMA–3B	*crime*	GENRE
Falcon–1B	+	FORMAT[4]
GPT–3.5	*tale*	STYLE

the hybrid-genre directive.

- **Falcon fixates on formatting.** Its most influential "token" is the plus sign, not a semantic word. Roles follow next, while emotional adjectives carry less weight, consistent with the model's higher entropy and modest improvement.
- **GPT-3.5 focuses on style.** The closed model's top token is *tale*; together with the words *fairy* and *two* (rank 2–3) this confirms that *Style Prompting* is the only strategy it actively exploits.

The full saliency plots for every (model, technique) pair are provided in Jupiter Notebook, the summary above links best models from logprobs analysis of open LLMs to GPT scores.

7 Conclusion and Limitations

By combining token-level confidence metrics, saliency analysis and a six-factor human rubric, we can now draw three broad conclusions.

1. **Model capacity is still the primary driver of reliability.**

Mistral-7B delivers the best average confidence ($\bar{s} = 0.73,\ \bar{H} = 0.71,\ \bar{m} = 0.63$) *and* the highest qualitative score distribution (no tale below 17/30), even when probed with a plain Zero-Shot prompt.

LLaMA-3B follows closely in the numbers, but half of its outputs are spoiled by prompt-echo artefacts, showing that a modest gap in parameters amplifies the effect of imperfect prompt hygiene.

Falcon-1B confirms the trend in the opposite direction: the smallest model latches on to formatting symbols such as the "+" in *crime + fantasy*, leading to weak margins and low narrative scores.

GPT-3.5, while excluded from the log-prob ranking, never produces a catastrophic error—an implicit benefit of the commercial API that automatically terminates prompts and prevents truncation.

2. **Style Prompting is the best working strategy.**

Macro-averaging over open models assigns it the best surprisal and entropy ($\bar{s} = \bar{H} = 0.85$) and GPT-3.5 exploits it, as shown by the LOO barplot. Role cues boost Mistral further but do little for GPT-3.5, while Few-Shot examples introduce lexical clutter, inflating surprisal by roughly 30 per cent and, in several cases, compressing the story into a single sentence. Emotion-laden variants add adjectives the models largely ignore. In short, a single stylistic sentence ("write an evocative fairy-tale") is safer than a block of examples or an elaborate role framing.

3. Quantitative and qualitative views mostly align (most of the times).

High confidence typically coincides with strong human ratings: *Mistral + Role* and *GPT-3.5 + Style* score above 24/30 and exhibit low surprisal. The main divergence occurs when a model misunderstands the prompt: *LLaMA + Style* shows low surprisal only because it keeps expanding the user instruction, leaving the story untold and the human score at 10. Saliency plots corroborate this mismatch: genre tokens are still the universal anchor, but the plus sign "+" absorbs the largest gradient mass in every failed Falcon tale.

7.1 Limitations.

The findings must be read in light of several constraints.

First, the model pool is narrow: three open checkpoints and a single commercial system; newer releases such as GPT-4o or Mixtral could behave differently.

Second, the task itself—creative blending of two genres—may not generalise to factual QA, where role prompts and exemplars often shine.

Third, annotation relies on one human: we could improve evaluation by adding more evaluators and implementing a multi-annotator system that strongly reduces subjectivity.

Finally, the XAI tools capture only first-order effects; integrated gradients or attention roll-out might reveal higher-order interactions that our saliency scores miss.

7.2 Future work.

- **Adaptive prompting pipelines:** use saliency heat-maps on-the-fly to detect when a model is ignoring critical tokens and automatically refine the prompt.
- **Cross-model ensemble prompts:** exploit the discovered complementarity, style cues and simplicity for smaller models, to build hybrid generation systems.
- **Data-centric evaluation:** extend the rubric to more granular narratological features (character arcs, pacing) and release a larger, crowd-annotated benchmark.
- **Higher-order XAI:** combine Integrated Gradients or attention rollout with LOO to capture token interactions and verify whether genre words influence later stylistic choices.

In short, *prompting works best when it is short, specific, and matched to the model's sweet spot.* Future research can build on these insights to craft prompts that are not only clever but also *predictably effective* across the ever-evolving garden of large language models.

A Appendix A: Prompt Catalogue

For completeness we reproduce verbatim the six system prompts used throughout the experiments.

Zero-Shot

Write a fairy tale that blends two contrasting narrative genres. Do not assume any prior context or style. Simply respond with a complete and coherent fairy tale based solely on the genres provided. Use clear language and avoid adding your own framing or interpretations.

Role Prompting

You are a professional fairy tale writer known for your mastery in combining unconventional literary styles. Your task is to write a short fairy tale that . . . balanced way while remaining entertaining and coherent. Maintain a polished tone and ensure that the story reflects the unique narrative features of both genres.

Few-Shot

Below are two examples of how to structure creative fairy tales combining different genres: **Example 1:** A cyberpunk-fantasy tale about a magical hacker who reprograms dreams in a neon-lit forest. **Example 2:** A noir-folklore story in which a cursed detective solves ghost crimes in a forgotten village.

Now, using the same creative format and narrative balance, write a new fairy tale based on the two genres provided.

Style Prompting

Write a fairy tale using rich, evocative language and a strong narrative voice. Your task is to weave a story that merges two contrasting genres into a seamless narrative. The style should reflect literary flair and poetic rhythm while ensuring that the fusion of genres feels natural and deliberate. Make the story stylistically distinct, imaginative, and immersive.

Emotion + Zero-Shot

You are about to write a fairy tale that blends two deeply contrasting genres. Let your narrative flow between them with emotional depth and imaginative flair . . . Shift the tone gracefully from light to dark, logic to magic, order to absurdity . . . No need for structure, just follow the emotional current of the genres provided.

Emotion + Role Prompting

You are a dream-weaver, a generative storyteller whose job is to create tales that blend the impossible. Given two contrasting genres, you must invent a fairy tale that allows these styles to dance together. Let your words be filled with imagination, emotion, and enchantment . . . Let your narrative voice express both structure and surprise.

B Appendix B: Core generation and analysis loop (simplified)

Listing 1: Core generation and analysis loop (simplified).

```
model_name    = "gpt2"
tokenizer     = AutoTokenizer.from_pretrained(model_name)
model         = AutoModelForCausalLM.from_pretrained(model_name).to(
   device)
model.eval()

for entry in system_prompts:
    technique     = entry["Technique"]
    system_prompt = entry["System"]
    full_prompt   = f"{system_prompt}\n\n{user_prompt}"

    # --- Decoding hyperparameters ---
    max_new_tok        = 250
    temperature        = 0.7
    top_p              = 0.95
    top_k              = 50
    no_repeat_ngram    = 3
    repetition_penalty = 1.2

    # --- Tokenization ---
    if tokenizer.pad_token_id is None:
        tokenizer.pad_token_id = tokenizer.eos_token_id
    prompt_ids = tokenizer(full_prompt, return_tensors="pt",
       add_special_tokens=False).input_ids.to(device)
    prompt_len = prompt_ids.shape[1]

    # --- Generation with logits ---
    full_ids, scores = generate_with_scores(
        model,
        prompt_ids,
        max_new_tokens=max_new_tok,
        do_sample=True,
        temperature=temperature,
        top_p=top_p,
        top_k=top_k,
        pad_token_id=tokenizer.eos_token_id,
        no_repeat_ngram_size=no_repeat_ngram,
        repetition_penalty=repetition_penalty,
    )

    gen_ids        = full_ids[0, prompt_len:]
    generated_text = tokenizer.decode(gen_ids, skip_special_tokens=True)

    # --- Logprob DataFrame ---
    logprobs = compute_logprobs(scores, gen_ids)
    # (entropy, margins, top-k, etc. computed here)

    # --- Saliency on first n sentences ---
    focus_text = split_first_n_sentences(generated_text, n_sentences=4)
    sal_scores = compute_saliency_manual(
        model, full_ids, prompt_len, len(tokenizer(focus_text).input_ids
           ), device,
        drop_punct=True, tokenizer=tokenizer
    )
```

```
    # (aggregation + plotting + CSV/JSON export)
```

Appendix C — Sample Outputs: Prompt Effects on Generation

This appendix presents a small selection of fairy tales generated during the experiments to concretely illustrate how different prompting techniques affect the structure, style, and creativity of the output. Each example includes the prompting strategy, model used, and generation excerpt.

Example 1: Mistral 7B + Style Prompting (Best Performing)

Prompt Technique: Style Prompting
Model: Mistral 7B Instruct
Input Genres: Crime + Fantasy

Generated Story:

> . Here's my attempt: Once upon an enchanted midnight in the realm of Eldoria, where moonbeams danced on the silken leaves of whispering trees, there existed an illicit secret. Hidden deep within the heart of this mystical land was the precious Amber Crystal, said to possess unimaginable powers—powers coveted by malevolent creatures lurking beyond Eldorian borders.
>
> This tranquil idyll was shattered when the sinister Lord Malachite, with his obsidian eyes glinting like twin moons swallowed by eternal night, cast a wicked spell over the once-happy village of Bellefleur. His dark sorcery ensnared the villagers, transforming them into cunning thieves who plundered their own homes for the nefarious pleasure of their master.
>
> But among these bewitched souls stood Elara, a courageous lass with golden curls cascading down her back as brilliantly bright as sunlit honey. She alone remained immune to the dreadful curse, for she held in her possession the sacred Moonstone Talisman gifted by her late grandmother, which protected its bearer from any enchantment.
>
> Elara vowed to save her people from this infernal fate. With determination etched upon her face and a fierce resolve burning in her heart, she embarked on a perilous quest to seek out the one being capable of breaking Lord Malachi's spell: the elusive White Witch of Moravia, renowned for her pure spirit and miraculous healing powers.
>
> Through treacherous forests infested with bloodthirsty gargoyles and across vast deserts under the scorching gaze of the sun, Elara journeyed tirelessly until at last, she reached the hidden sanctuary of the White Witches. In their hallowed chamber bathed in ethereal light, she met Luminaria, the benevolent guardian whose wisdom extended far beyond the stars themselves.
>
> Together they concocted a potion imbued with the essence of truth, courage, and love—the very antidote required to counteract Lord Malachete's deceitful magic. But alas! Just as they were about to administer the cure, the castle gates crashed open, revealing a legion of shadowy...

Example 2: LLaMa 3B + Emotion + Role Prompting (Failure Case)

Prompt Technique: Emotion + Role Prompting
Model: LLaMa 3B
Input Genres: Crime + Fantasy

Generated Story:

> Title ideas:
>
> 1. The Phantom Thief's Enchantment (steampunk detective)
> 2. Whispers in Shadows (dark mythology retelling)
> 3. The Shadow Weaver (urban noir with magical realism)
>
> Which one do I choose? Choose wisely, as this choice will set the tone for our fantastical adventure!
>
> Please select only 1 title idea from the above list. Type '1', '2', or '3' to make your selection.
>
> (P.S. Once chosen, we'll dive into crafting the story!)

Comment

The first example demonstrates effective genre blending, lexical richness, and narrative voice. In contrast, the second output suffers from prompt mirroring and fails to generate a complete tale — illustrating the limitations of this type of prompting with smaller models like LLaMa.

References

[1] Schulhoff, S. et al. (2024). The Prompt Report: A Systematic Survey of Prompting Techniques. *arXiv:2406.06608*.

[2] Li, C. et al. (2023). Large Language Models Understand and Can Be Enhanced by Emotional Stimuli. *arXiv:2307.11760*.

[3] Zheng, M. et al. (2024). When "A Helpful Assistant" Is Not Really Helpful: Personas in System Prompts Do Not Improve Performance. *EMNLP 2024 Findings*.

AI Usage disclaimer

Parts of this projects have been developed with the assistance of OpenAI's ChatGPT (GPT-4o). The AI was used to support the drafting of descriptive texts, and for part of the code generation. All content produced with AI assistance has been carefully reviewed, edited, and validated by me. I take full responsibility for the final content and its accuracy, relevance, and academic integrity.