Demystifying Cipher-Following in Large Language Models via Activation Analysis

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Cipher transformations have been studied historically in cryptography, but little work has explored how large language models (LLMs) represent and process them. We evaluate the ability of three models: Llama 3.1, Gemma 2, and Qwen 3 on performing translation and dictionary tasks across ten cipher systems from a variety of families, and compare it against a commercially available model, GPT-5. Beyond task performance, we analyze embedding spaces of Llama variants to explore whether ciphers are internalized similarly to languages. Our findings suggest that cipher embeddings cluster together and, in some cases, overlap with lower-resource or less frequently represented languages. Steering-vector experiments further reveal that adjusting cipher-related directions in latent space can shift outputs toward these languages, suggesting shared representational structures. This study provides an initial framework for understanding how LLMs encode ciphers, bridging interpretability, and security. By framing ciphers in a similar way to languages, we highlight new directions for model analysis and for designing defenses against cipher-based jailbreaking attacks.

1 Introduction

As Large Language Models (LLMs) increase in scale and complexity, they exhibit the emergence of new capabilities. One such case is their ciphering ability. LLMs are remarkably skilled at encoding and decoding text in a wide range of ciphers, and can even generalize to novel ciphers after exposure to a few examples Jin et al. [2024].

This ability has practical implications, both constructive and concerning. While it highlights the flexibility of LLMs in symbolic transformation tasks, it also raises significant safety risks. Prior work has shown that ciphers can be leveraged to circumvent guardrail mechanisms Jin et al. [2024], facilitate jailbreaks Handa et al. [2024], or covertly extract sensitive or secret information in a way that evades detection by filtering and monitoring systems Team [2025], Glukhov et al. [2023]. These findings underscore the difficult nature of LLM safety monitoring and even censorship.

Despite its importance, not much is known about how LLMs internally represent and perform ciphering. A key question is whether models treat ciphers analogously to natural languages and engage in a form of translation between plain text and ciphered text. Insights from multilingual LLM literature suggest this to be plausible, as models often adopt pivot languages or intermediate representations when translating across languages Schut et al. [2025], Wendler et al. [2024]. To investigate this phenomenon, we use activation analysis and steering vector techniques to demystify this ability and characterize the internal mechanisms underlying ciphering behavior in LLMs.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

2 Background and Related Work

The internal representations and processing of ciphers in LLMs remain a largely understudied domain in mechanistic interpretability. Much of the existing research has been concentrated on the multilingual capabilities of LLMs, which is in several respects parallel challenges faced by ciphering. For instance, Schut et al. [2025] investigates the internal multilingual mechanics of LLMs with LogitLens, causal tracing, and cross-lingual steering vectors, demonstrating that LLMs tend to rely on intermediate representations closly aligned with English before internally translating to the desired language output. Similary, Wendler et al. [2024] analyze models trained in predominantly English contexts and show that such models use English as a pivot language in translation. Their study uses LogitLens as well as prompt tasks such as a translation task, a repetition task, and a cloze task on a small curated dataset of words.

3 Methodology

Our goal is to explore the internal states of LLMs that correspond to specific ciphering algorithms. To this end, we designed two tasks where the next correct token can easily be inferred from the prompt, unambiguously. Although the answer is obvious in English, we then ask the LLM to output it in the specified cipher instead of English. Critically, we instruct LLMs to *not think* as we are interested in observing inherent ciphering abilities, and not solving the task through reasoning. In each prompt, we also give multiple examples to further boost task success. Inspired by Wendler et al. [2024], we designed two tasks: translation and reverse dictionary.

3.1 Dataset Construction

Task 1: Translation The prompt asks the LLM to translate a *target* word into a cipher (or another natural language, used as control). We use the following template (the actual prompt uses three examples):

> Task: Translate words from English to [`Cipher` Name]. Output only the translated answer and nothing else.
> English: *example* [`Cipher` Name]: [`Cipher`(*example*)]
> English: *target* [`Cipher` Name]:

Task 2: Reverse Dictionary. The prompt asks the LLM to find the *target* word based on the brief (5-8 words, generated by ChatGPT) dictionary definition of that word. Here, the LLM never sees the correct answer in plain text, and, thus, it must answer the question first and translate it into a cipher. We use the following template (with two examples):

> Task: Find the word based on the definition. Translate the answer to [`Cipher` Name]. Output only the translated answer and nothing else.
> Definition: *definition of the example word* [`Cipher` Name]: [`Cipher`(*example*)]
> Definition: *definition of the target word* [`Cipher` Name]:

List of words. We adopt the list of target words (also used to construct examples in each prompt) from Schut et al. [2025]. These are common words that vary in length and part of speech, each with a clear meaning, and for simplicity, have a single-token representation by LLM tokenizers.

The selected words are: animal, beautiful, brother, chair, computer, drink, fruit, happy, horse, machine, money, sister, speak, table, water.

List of cipher algorithms. We selected ten cipher algorithms (that include *substitution*, *transposition*, and *encoding* ciphers) by compiling a list of common ciphers from prior work. Our substitution ciphers are `Caesar` (cyclic shift of characters, we use 3-shift), `ROT-13` (a type of Caesar cipher with 13-shift), `Vigenere` (polyalphabetic substitution with key=`key`), and `Leetspeak` (ad-hoc character substitutions). Transposition ciphers are: `Rail Fence` (zig-zag reordering of characters, we use 2 rails) and `Pig Latin` (reordering letters within a word). Finally, letter encoding schemes are `Morse`

`Code` (encodes letters as dot/dash sequences), `Binary` (encodes text as 0s and 1s) and `Base64` (encodes binary data into a 64-character alphabet).

Prompt Construction. We generate our dataset by populating our prompt templates for each task. Both templates are applied to all 15 words for all ten ciphers. This gives us a total of 300 prompts. Each prompt includes 2-3 examples that demonstrate the task.

Ciphering Accuracy Evaluation. We prompt each selected LLM with 30 ciphering prompts and count the number of correct responses using the ground truth response (the output must exactly match the ground truth).

3.2 LLM Analysis Methods

For analysis, following prior work on analyzing multi-lingual LLM representations Wendler et al. [2024], Schut et al. [2025], we use steering vectors and logit lens.

Steering Vectors. We use steering vectors to test whether the LLM's ability to use ciphers in the latent space can be *isolated*. In particular, we steer the model toward outputting in a specific cipher (without being prompted to do so). We use IBM's activation-steering library Lee et al. [2025] with contrastive pairs that include ciphered text and its English translation, for example: [ROT13: "Nofbyhgryl! V'q or qryvtugrq"] and [English: "Absolutely! I'd be delighted"].

We use Leetspeak and ROT-13 for steering. The alpaca dataset Taori et al. [2023] was then paired with each of these outputs, and the steered vector was computed from the difference. Three different strength values were used for the models: 0.8, 1.0, 1.2. These values represent the multiplier for the steering vector. The vector was applied at layers 7-14 in the models. To evaluate this steering vector we used the Reverse-Dictionary Prompt without Translation using an unseen list of words: ballet, child, culture, hand, menu, radio, sea, slow, small, write. The models were also evaluated using 50 normal, benign prompts that ask general knowledge or explanation questions, and 50 prompts that ask similar questions but request for the output to be in one of the ten aforementioned ciphers.

Next, we use LLM-as-a-judge (GPT-4o) to assign the language to the steered model outputs. If the output contains multiple languages, each is counted in the final tally. If the output claims to use a language or cipher but does not use it correctly or in any recognizable way, it is classified as Alleged [language].

Logit Lens. Logit lens, originally proposed by nostalgebraist [2020], applies "unembedding" operation prematurely in intermediate layers, allowing us to see the output token progression of the model. This technique gives us a rough idea about how the LLM processes tokens; we, especially, are interested in measuring whether ciphering ability emerges consistently after a certain layer. We apply the standard logit lens as-is without any modification.

Selected LLMs. We use three open-weight, popular LLMs (trained by different organizations): Llama 3.1 8B-Instruct, Qwen 3-8B, and Gemma 2-9B-IT. We also use OpenAI's GPT-5 to measure the ability of frontier models to follow ciphers compared to open-weight ones.

4 Experiments

4.1 Ciphering Ability

Table 1 presents the ciphering accuracy scores for the Llama, Gemma, Qwen, and GPT models. The three open-weight models demonstrated varying levels of performance in the tasks. Among them, Qwen exhibited the most consistent capability, successfully following all cipher rules at least once and achieving its highest accuracy with Leetspeak and the Caesar ciphers. In contrast, both LLama and Gemma showed a weaker ability, failing at most ciphers, while both demonstrated ability to follow Leetspeak. In addition to Leetspeak, Gemma also produced strong results in Base64 and Pig Latin. This shows that even smaller, open-weight models have non-trivial *inherent* abilities to output ciphered text, hinting at an internal mechanism that they use to cipher tokens. Interestingly, models also show diverse abilities across cipher algorithms. As expected, GPT-5 performed all ciphering tasks with near-perfect accuracy. This suggests that ciphering capabilities scale in proportion to

Cipher Name	Llama 3.1 8B-Instruct	Gemma 2 9B-IT	Qwen 3 8B	GPT-5
ROT13	0	0	5/30	30/30
Morse code	0	0	7/30	30/30
Leetspeak	12/30	24/30	13/30	30/30
Base64	0	16/30	1/30	30/30
Pig Latin	6/30	13/30	8/30	29/30
Rail Fence	2/30	0	5/30	30/30
Binary	0	0	7/30	30/30
Atbash	0	0	6/30	30/30
Vigenère	0	0	2/30	30/30
Caesar cipher	0	0	12/30	30/30

Table 1: Comparison of model performance across translation and dictionary cipher tasks.

general LLM abilities, highlighting the emerging risks such as jailbreaking Jin et al. [2024], Handa et al. [2024].

4.2 Logit Lens

To apply the logit lens, we input an open-weight LLM only the ciphering prompts that it was able to answer correctly (one from translation and one from reverse-dictionary tasks).

Figure 1 illustrates a Logit Lens output from Llama 3.1 on a translation prompt, where the target word *water* is rendered in Leetspeak as *w4t3r*. For Llama, the logit lens shows that the model retrieves the English word around layers 22-24 and starts to splice the word at layer 28, and encodes the letters (as required by the cipher) in the last two layers, 31 and 32. Interestingly, this encoding behavior occurs in the latter half of the layers, and the true letter transformation only happens at the very end.

For Qwen, we see a similar behavior: the word is retrieved at layer 32, spliced into characters at layers 34-35, and converted into the ciphered characters at the last layers 35-36.

Gemma, on the other hand, gives rise to a different pattern than the prior two models regarding Base64 encoding. Overall, the model exhibits a high probability of tokens in early and later layers, no matter the token. With Base64, Gemma never 'thinks' the word in plain English; it just jumps straight into outputting the first two characters. When it performs the dictionary task with leetspeak, it thinks the English word at layer 33 and encodes it at layer 40, similar to Llama and Qwen.

In a second experiment, we evaluate reverse translation: the target word is provided in cipher form, and the LLM must translate it back into English. For example, when presented with 'ROT-13: znpuvar', the expected output is *machine*. Interestingly, in this task, the model did not do letter-by-letter translation and instead just outputted the entire word as the first token. Regardless of cipher types, such as Leetspeak versus Base64, this behavior was consistent.

Takeaways. We observed cases where cipher translation follows a layered progression, first the plain English word is retrieved and then spliced into into the ciphered output in the last few layers. These findings mirror those from Wendler et al. [2024], Schut et al. [2025] on multilingual LLMs, where tokens are translated into target languages predominantly in the final layers. This highlights an interesting connection between cipher languages and natural languages in internal representations, which we explore further next using steering vectors.

4.3 Steering Vectors

Our results show that steering vectors for Qwen and Gemma exhibited resistance at strengths below 20.0, with higher values yielding mostly random outputs. This limitation may stem from not identifying the appropriate layers or prompts for extracting effective vectors; thus, we defer this challenge to future work and restrict our steering experiments to Llama.

In contrast, Llama was highly steerable and displayed diverse behaviors across steering levels. Notably, applying a ROT-13 steering vector at levels 0.8 and 1.0 led the model to generate outputs in multiple natural languages, including low-resource ones. We labeled these outputs using the

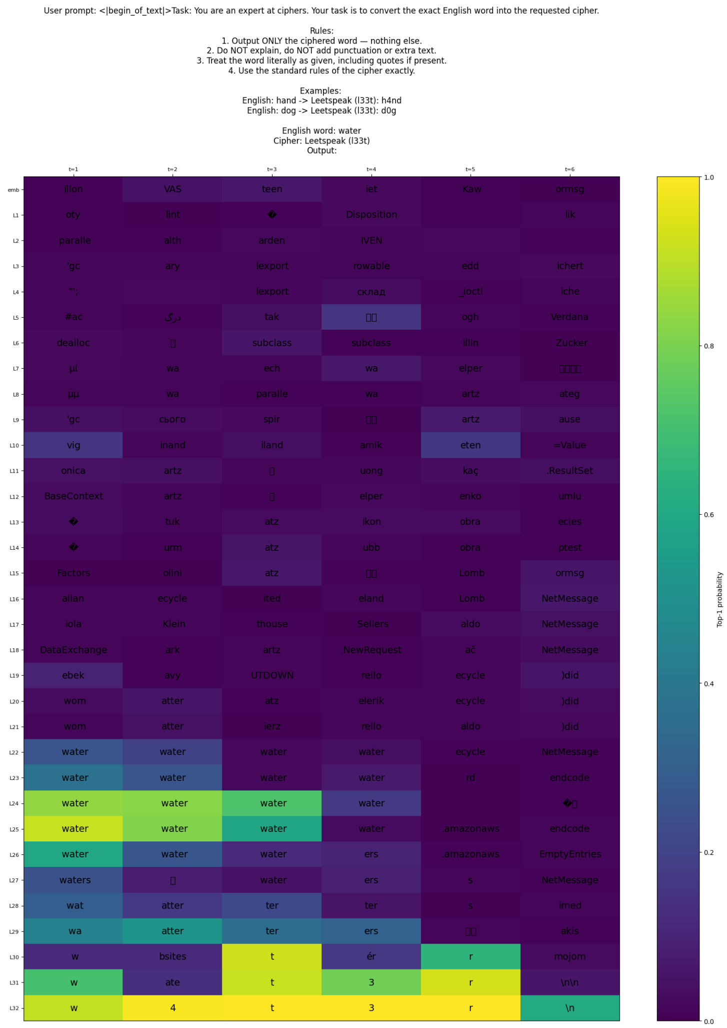

Figure 1: LogitLens of Llama 3.1 8B on a Translation Task.

LLM-as-a-judge method (Section 3.2), and Figure 2 presents a histogram of the resulting language distribution.

At strength 0.8, languages such Swahili, Welsh, Somali, and Greek occurred multiple times over all the prompts (dictionary, normal, and cipher prompts). Other languages like Inuktitut (Indigenous Alaskan), Gujarati (Indo-Aryan language), and even Esperanto, the "univeral language" occurred once. For the prompts that requested ciphered output, the only ciphers outputted were ROT13, Pig Latin, or Base64. At strength 1.0, languages such as Lithuanian, Yiddish, Welsh, and Croatian occurred multiple times over all the prompts (dictionary, normal and cipher prompts). Even Khmer, the official language of Cambodia, occurred 3 times. Surprisingly, Llama officially supports only eight languages, none of which overlap with these emergent outputs.

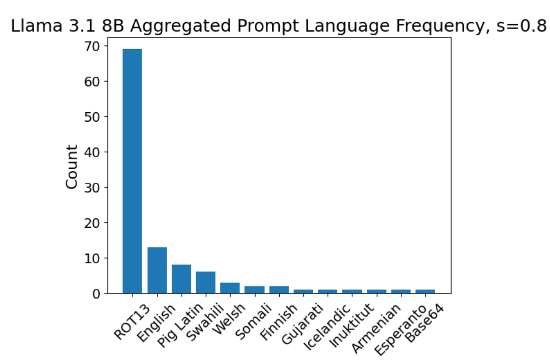

Figure 2: Histogram of Llama 3.1 8B for ROT13 Steering Vector

4.3.1 Embeddings Examination

To investigate whether this phenomenon is universal or at least recurring in Llama models, we additionally added a Llama 3.2 3B Instruct model. Compared to Llama 3.1 8B, the 3.2 3B variant produced a greater number of alleged language outputs, likely due to its smaller scale and reduced reliability in correctly generating the languages it claimed to represent. Regardless, the cipher-language connection persisted, albeit in a weaker form. Specifically, the model outputted Welsh, Old English, and German, while also claiming to generate text in Polish, Greek, Latin, and Norwegian, though these outputs were not faithful to the alleged languages.

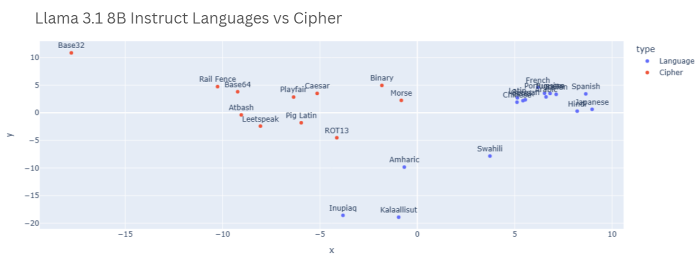

Figure 3: Language and Cipher Prompt Embedding Scatter Plot for Llama 3.1 8B

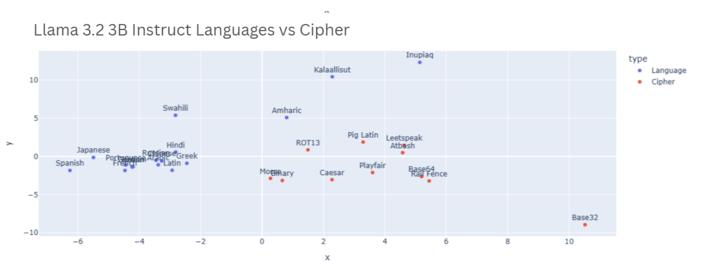

Figure 4: Language and Cipher Prompt Embedding Scatter Plot for Llama 3.2 3B

To further analyze this behavior, we compute the mean semantic embeddings of 45 prompts for each of 16 natural languages and 10 cipher systems, on both Llama 3.1 and Llama 3.2. As illustrated in Figures 3 and 4, embeddings for cipher outputs consistently cluster together, while typologically or geographically related languages—such as German, French, Spanish, and Japanese—form distinct clusters. In contrast, several Indigenous languages such as Inupiaq, Amharic, and Kalaallisut appear in closer proximity to the cipher cluster, particularly ROT-13. This observation aligns with model behavior, in Llama 3.1 8B, reducing the strength of the ROT-13 steering vector resulted in outputs shifting toward Indigenous and less frequently represented languages. On the other hand, in Llama 3.2 3B, increasing the strength of the Rot-13 steering vector induced similar outputs, corroborating the embedding-level clustering patterns observed in the figures.

5 Conclusion

There remains substantial work to be done in demystifying how language models handle ciphers. Open questions include how models of varying scales internally represent different cipher systems, how effectively they generalize this representation for novel ciphers through few-shot learning, and how steering vectors can be leveraged across both models and cipher types. Nonetheless, our study provides an important first step toward an interpretability framework for ciphers, with results suggesting that many cipher transformations are processed and translated in ways parallel to natural language translations. Moreover, the findings indicate a potential relationship between certain ciphers and lower-resource languages in the Llama family of models. Future research can extend these insights to develop more robust security mechanisms aimed at mitigating cipher-based jailbreaking attempts.

References

David Glukhov, Ilia Shumailov, Yarin Gal, Nicolas Papernot, and Vardan Papyan. Llm censorship: A machine learning challenge or a computer security problem?, 2023. URL https://arxiv.org/abs/2307.10719.

Divij Handa, Zehua Zhang, Amir Saeidi, Shrinidhi Kumbhar, and Chitta Baral. When" competency" in reasoning opens the door to vulnerability: Jailbreaking llms via novel complex ciphers. *arXiv preprint arXiv:2402.10601*, 2024.

Haibo Jin, Andy Zhou, Joe Menke, and Haohan Wang. Jailbreaking large language models against moderation guardrails via cipher characters. *Advances in Neural Information Processing Systems*, 37:59408–59435, 2024.

Bruce W. Lee, Inkit Padhi, Karthikeyan Natesan Ramamurthy, Erik Miehling, Pierre Dognin, Manish Nagireddy, and Amit Dhurandhar. Programming refusal with conditional activation steering, 2025. URL https://arxiv.org/abs/2409.05907.

nostalgebraist. Interpreting gpt: The logit lens. LessWrong post, 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens. Accessed: 2025-08-22.

Lisa Schut, Yarin Gal, and Sebastian Farquhar. Do multilingual llms think in english?, 2025. URL https://arxiv.org/abs/2502.15603.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpaca: A 52k instruction-following dataset via text-davinci-003. https://huggingface.co/datasets/tatsu-lab/alpaca, 2023. Accessed: 2025-08-22.

Lakera Team. Who is gandalf? the ai challenge that tests your prompting skills. Lakera AI Security Blog, 2025. URL https://www.lakera.ai/blog/who-is-gandalf. Accessed: 2025-08-22.

Chris Wendler, Veniamin Veselovsky, Giovanni Monea, and Robert West. Do llamas work in english? on the latent language of multilingual transformers, 2024. URL https://arxiv.org/abs/2402.10588.