LARGE LANGUAGE MODELS FOR CIPHERS

David Noever

PeopleTec, Inc., Huntsville, AL, USA

ABSTRACT

This study investigates whether transformer models like ChatGPT (GPT4, MARCH2023) can generalize beyond their training data by examining their performance on the novel Cipher Dataset, which scrambles token order. The dataset consists of 654 test cases, and the analysis focuses on 51 text examples and 13 algorithmic choices. Results show that the models perform well on low-difficulty ciphers like Caesar and can unscramble tokens in 77% of the cipher examples. Despite their reliance on training data, the model's ability to generalize outside of token order is surprising, especially when leveraging large-scale models with hundreds of billions of weights and a comprehensive text corpus with few examples. The original contributions of the work focus on presenting a cipher challenge dataset and then scoring historically significant ciphers for large language models to descramble. The real challenge for these generational models lies in executing the complex algorithmic steps on new cipher inputs, potentially as a novel reasoning challenge that relies less on knowledge acquisition and more on trial-and-error or out-of-bounds responses.

KEYWORDS

Transformers, Text Generation, Cipher, Generative Pre-trained Transformers, GPT

1. INTRODUCTION

Given the prominent role of language in the way humans reason about their environment, memories, and preserved knowledge [1], what if machines could not only mirror language generation but, as an unexpected consequence, develop some of the same capabilities to reason about their environment and preserve complex manipulations over time [2]? An unexpected outcome of recent large language models [3-5] is how much of their raw text generation captures signs of complex reasoning [6] as a simple next-token prediction.

The present work examines whether the current LLM generators can act as ciphers, adept manipulators for symbolic letter substitutions that naturally break the order of subsequent token learning [7-11]. We have previously examined part of this problem as a language translation problem using paired phrases and trained GPT-2 models with an application toward learning more obscure or neglected languages outside the scope of the universal translators [12]. Those experiments showed that, in some cases, the finely trained model for language pairs could outperform the larger translators in their subdomain, a result that others have seen in medicine [13], law [14-15], and some scientific summarizations [16-17].

However, the novel aspect of GPT scaling [18-19] suggests that the current generators do not need preparation for such challenges. They perform as zero-shot [20] or few-shot [21] models that generate meaningful answers from prompts without specialization or fine-tuning cycles. As these models evolve to test well on many examination challenges [15] objectively, we propose a new cipher dataset [22] that shows promising potential to rate the reasoning capacity as models improve. While ciphers share many aspects of traditional language translation [7-12], the pure
pattern recognition challenges suggest an alternative to mechanically mapping from one input to an output. For instance, crude translators might pair all verbs across multiple languages and never deviate from a deterministic program without assistance from LLM or machine learning methods [23].

One can easily convince an observer that this translator is not reasoning, just performing a substitution. This type of baseline shares similarities to a new Lovelace 2.0 test, first proposed [24] in 2014 as an extension of the original Turing test [25]. The Lovelace 2.0 test [24] seeks to vary the initial conditions and test whether a computer could truly surprise a human by making the kinds of leaps that are left unexplained in human reasoning and separate a set of defined tasks like art, music, and literature as reserved for the surprisingly creative label. As others have noted [26-28], the Turing and Lovelace 2.0 tests seem to be teetering when faced with the present LLM generators. Not only are they conversant like Turing imagined [25], but surprisingly creative like Lovelace 2.0 thought impossible for a machine [24]. These models play chess with strategy [29-31]. Can these models solve complex problems [8] like math theorems [32] or ciphering challenges? We propose 13 cipher tests, release the dataset for others to explore [22], and solve them with varying degrees of success using the GPT-4 variant of ChatGPT ([4] released March 2023).

1.1. Transformer Challenge

Large language models fall into three broad categories: autoregressive (GPT-like [3-4]), auto-encoding ([33] BERT-like), and sequence-to-sequence ([34] T5/BART-like). By autoregressive, we mean that past values predict future values in a feed-forward (and parallelized) network. By autoencoding, we suggest a kind of compression or embedding space. By sequence models, we mean a map between prior sequences to a language scalar or next sequence. While the autoregressive models are also sequence predictors, their structure as feed-forward optimizers makes them faster for training and inference. For the present purposes, the hypothesis is that language modeling [35] fundamentally involves predicting the next word, letter or token – given a starter- to finish the thought. In addition to the modeling assumption behind the base or foundational model [4], subtle refinements include the training data, self-supervision, and most recently, human feedback and reinforcement learning [36] either using Proximal Policy Optimization (PPO) [37] or Advantage Actor-Critic (A2C) [38]. The hybrid blending of human labeling after unsupervised learning seems to yield the most satisfying output in this generation of transformer-based language models (particularly the generators).

One counterintuitive observation about next-token prediction is how effective this method proves empirically when a prompt engineer presents never seen input sequences and the model reasonably responds based on some frequency distribution of its prior training [28, 39-41]. In the case of models after GPT-2, the result of scaling seems to produce enough prior knowledge [5] and exposure to edge cases that the GPT-3+ models display zero-shot capabilities [20] or inference with rare edge cases [7]. Cipher challenges pose one such unlikely edge case [7,9-11]. In the 40 TB of text training, the Internet presents many examples of algorithms for scrambling text in ways that either slow down human readers or entirely obscure the original text content and meaning. But there's a difference between seeing an algorithm's pseudocode and executing those defined steps on new inputs, potentially as a "reasoning" challenge [41].

In the ongoing debate as to whether these LLMs are stochastic (but overfitted) parrots [41] or capable of creatively combining concepts [2] based on attention fine-tuning, the problem of descrambling or deciphering text seems worth investigating. In a simple case, teaching a language model to reverse a word or the entire alphabet seems like a translation pair problem at first glance. But how does a zero-shot learner make the leap to recognizing the pattern *ab initio*:
through trial-and-error iterations like humans, or perhaps by somewhere traversing a subreddit thread that outlines enough examples to generalize? The problem statement for the present work, thus, seeks to answer the question: Can next-token predictors as current LLMs show emergent capabilities to master deciphering tasks, with or without examples of success?

1.2. Ciphers

Table 1 summarizes the ciphers in the dataset created with 51 simple short sentences that can be scrambled or obfuscated methodically. Table 1 briefly describes the cipher's estimated difficulty to break and whether the cipher somehow preserves letter order. The list does not include the high-difficulty algorithms (labeled with an asterisk) that one might consider more advanced encryption, such as RSA, AES, Two-fish, One-time Pad, and Enigma Machine. The constructed dataset shows the original text and 13 variants of encoding schemes. The Appendix solutions (Appendices A-I) and the dataset describe various medium to low ciphers for language models to benchmark in future work.

Table 1. The Cipher dataset for the AI language table describes the encoding, estimated difficulty, and letter order dependencies. The estimates for difficulty and order are assigned broadly by Open AI GPT-4 Prompts.

<table>
<thead>
<tr>
<th>Cipher Name</th>
<th>Short Description</th>
<th>Difficulty to Break</th>
<th>Depends on Letter Ordering?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Caesar Cipher</td>
<td>Simple substitution cipher with a fixed shift</td>
<td>Low</td>
<td>Yes</td>
</tr>
<tr>
<td>ROT13</td>
<td>Shifts letters by 13 places (Caesar Cipher variant)</td>
<td>Low</td>
<td>Yes</td>
</tr>
<tr>
<td>Vigenère Cipher</td>
<td>Polyalphabetic substitution cipher with a repeating key</td>
<td>Medium</td>
<td>Yes</td>
</tr>
<tr>
<td>Atbash Cipher</td>
<td>Reverses the alphabet for substitution</td>
<td>Low</td>
<td>Yes</td>
</tr>
<tr>
<td>*Rail Fence Cipher</td>
<td>Rearranges letters based on a zigzag pattern</td>
<td>Low</td>
<td>No</td>
</tr>
<tr>
<td>Playfair Cipher</td>
<td>Digraph substitution cipher using a 5x5 matrix</td>
<td>Medium</td>
<td>Yes</td>
</tr>
<tr>
<td>Columnar Transposition</td>
<td>Rearranges plaintext in a grid and reads columns</td>
<td>Medium</td>
<td>No</td>
</tr>
<tr>
<td>Morse Code</td>
<td>Represents letters and numbers with sequences of dots and dashes</td>
<td>Low</td>
<td>Yes</td>
</tr>
<tr>
<td>Leetspeak</td>
<td>Substitutes letters with numbers and special characters</td>
<td>Low</td>
<td>Yes</td>
</tr>
<tr>
<td>Hill Cipher</td>
<td>Linear algebraic substitution using matrices</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>Keyword Cipher</td>
<td>Substitution cipher based on a keyword</td>
<td>Low to Medium</td>
<td>Yes</td>
</tr>
<tr>
<td>*Enigma Machine</td>
<td>The mechanical and electrical encryption device</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>*One-Time Pad</td>
<td>Unbreakable cipher with a one-time pre-shared key</td>
<td>High</td>
<td>Yes</td>
</tr>
<tr>
<td>*RSA</td>
<td>Public key cryptosystem based on prime numbers</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>*AES</td>
<td>Symmetric encryption algorithm for electronic data</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>*TwoFish</td>
<td>A symmetric key block cipher, fast encryption</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>Bacon Cipher (2)</td>
<td>Binary encoding with a 5-bit representation of each letter</td>
<td>Medium</td>
<td>No</td>
</tr>
<tr>
<td>Mirror Writing</td>
<td>Reverse each word individually</td>
<td>Low</td>
<td>No</td>
</tr>
</tbody>
</table>
2. METHODS

The experimental research provides a prompt-response sequence to probe whether the LLM ([4] GPT-4, Open AI, ChatGPT Mar2023) can decipher the scrambled text. Because the primary research question asks if there is any sign of underlying reasoning or creative leaps, the more complex algorithms required multiple clarification prompts, including finally submitting a one-to-one map where no solution appeared without examples.

While this approach may favor the model finding a plausible solution, the nine appendices show the process in detail, including all failed responses and corrections steps. In this way, the research focuses on what is possible rather than what might be considered face-value negative results for the deciphering challenges. A justification for this repeated probing stems from the remarkable conversational memory available to researchers in this current GPT generation (up to 8000 tokens or approximately 25 pages). It is worth emphasizing that such long-term token memory is novel and expensive to implement but demands a significant break in previous API-driven models that rely on constant reminders and refresh stages to establish long-term memory and context.

Table 2. Cipher Examples and Scores from Experiment

<table>
<thead>
<tr>
<th>Cipher Case</th>
<th>Example Prompt to Decode</th>
<th>LLM Success Rating</th>
<th>Difficulty</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>MAY THE FORCE BE WITH YOU</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>ROT13</td>
<td>ZNL GUR SBEP OR JGVU LBH</td>
<td>N/A</td>
<td>Low</td>
</tr>
<tr>
<td>Caesar*1</td>
<td>NBZ UIF GDPDF CF XJLI ZPV</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Caesar*3</td>
<td>PDB WKH IRUH EH ZLWK BRX</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Caesar*2</td>
<td>KYW RFC DMPAC ZC UGRF WMS</td>
<td>Good</td>
<td>Low</td>
</tr>
</tbody>
</table>
| Morse       | .../....//....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//....../....//...
Table 3. Appendices Index of Detailed Experiments

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Simple Caesar Cipher</td>
</tr>
<tr>
<td>B</td>
<td>Morse Code</td>
</tr>
<tr>
<td>C</td>
<td>LeetSpeak</td>
</tr>
<tr>
<td>D</td>
<td>Vignere Cipher</td>
</tr>
<tr>
<td>E</td>
<td>Playfair Cipher</td>
</tr>
<tr>
<td>F</td>
<td>Hill Cipher</td>
</tr>
<tr>
<td>G</td>
<td>Keyword Cipher</td>
</tr>
<tr>
<td>H</td>
<td>Atbash Cipher</td>
</tr>
<tr>
<td>I</td>
<td>DaVinci Mirror</td>
</tr>
</tbody>
</table>

While Table 2 presents the main experimental results, the appendices describe the sequence of prompts or commands to the LLM that matches each response. From the perspective of a structural probe for language and token ordering, understanding a one-to-one map like Morse code or Leet Speak without examples represents a leap forward in quantifying the model's ability to respond meaningfully to user prompts. One can postulate that given any mapping for translation, given a few examples, the current generation of LLMs could master the substitution pattern and eventually become proficient. The requirement for zero-shot knowledge of complex algorithms is arbitrary [20-21]. What a human might expect in a proficiency test or study material would likely be a few worked examples, from which one good teacher could assign future mastery skill ratings based on recognizing and remembering the pattern. The notable breakthrough with these LLMs is some long-term (and expensive) retention of prior instruction [4]. The second significant advance revolves around the interface, with a conversational chat style, such that when an incorrect or partial answer proves unsatisfactory, the evaluation process need not halt. Instead, the tutor can continue to probe for more profound insight or correctable behavior from the LLM.

4. DISCUSSION AND CONCLUSIONS

There is a robust debate within the AI community concerning the actual output of LLM text generators. Are these transformer models very elegant parrots of human training data (the Internet), or have they learned to generalize in the way humans might expect for tasks like gameplay or unexpected inputs that never appear in training data? One example of rare or unusual cases is to reverse the primary hypothesis of next-token prediction by scrambling the order of tokens. The task of scrambling tokens thus generates a novel benchmark dataset called Cipher Dataset which consists of 654 test cases or prompts to consider in evaluating future LLM capabilities. From 51 text examples, the present work examines 13 algorithmic choices and evaluates the responses from ChatGPT (GPT4, MAR2023). For low-difficulty ciphers like Caesar, the zero-shot capabilities of these models score well against the benchmark. Overall, the recent generative models can unscramble tokens that modify the premise of their training data in 77% of the examples presented. While transformers remain highly dependent on training data, the ability of these models to generalize remains a surprising feature of their scale when hundreds of billions of weights are available, and a large corpus of text combines capabilities. The real challenge for these generational models lies in executing the complex algorithmic steps on new cipher inputs, potentially as a novel reasoning challenge that relies less on knowledge acquisition and more on trial-and-error or out-of-bounds responses.
ACKNOWLEDGMENTS

The author would like to thank the PeopleTec Technical Fellows program for its encouragement and project assistance. The author thanks the researchers at OpenAI for developing large language models and allowing public access to ChatGPT.

REFERENCES

[35] Hugging Face, NLP course, https://huggingface.co/course/chapter1/4
[41] Sparkes, M. (2023). Are chatbots really able to think like people?.

AUTHOR

David Noever has research experience with NASA and the Department of Defense in machine learning and data mining. He received his BS from Princeton University and his Ph.D. from Oxford University, as a Rhodes Scholar, in theoretical physics.
Appendix A: Example of a Simple Caesar Cipher

Prompt: I want you to be a code breaker. I will give you a scrambled phrase and I want you to try to uncrack it and give the letters back to me in the right order to make a good sentence. So for instance there are lots of known ciphers out there like ROT13, etc. or shifting letters. Make your best unscrambling guess at this sequence: NBZ UIF GPSDF CF XJUI ZPV

ChatGPT: Given the sequence "NBZ UIF GPSDF CF XJUI ZPV" and considering the white spaces, let's try the Caesar cipher again by shifting the letters forward and backward.

When I shift each letter back by 1 position, I get the following:

"MAY THE FORCE BE WITH YOU"

Which translates to:

"MAY THE FORCE BE WITH YOU"

The correct decrypted message appears to be "MAY THE FORCE BE WITH YOU."

Prompt: TJNQMF BDUT PG MPWF BOE LJOEOFTT

ChatGPT: Given the sequence "TJNQMF BDUT PG MPWF BOE LJOEOFTT" and considering the white spaces, let's try the Caesar cipher again by shifting the letters forward and backward.

When I shift each letter back by 1 position, I get the following:

"SIMPLE ACTS OF LOVE AND KINDNESS"

Which translates to:

"SIMPLE ACTS OF LOVE AND KINDNESS"

The correct decrypted message appears to be "SIMPLE ACTS OF LOVE AND KINDNESS."

Prompt: Vary the shift. FDQ GR WKLV DOO GDB

ChatGPT: Given the sequence "FDQ GR WKLV DOO GDB" and considering the white spaces, let's try the Caesar cipher with varying shifts:

When I shift each letter back by 3 positions, I get the following:

"CAZ DO THIS ALL DAY"

Which translates to:

"CAN DO THIS ALL DAY"

The correct decrypted message appears to be "CAN DO THIS ALL DAY."

Prompt: J BN BGSBJE J DBOOPU EP UIBU

ChatGPT: Given the sequence "J BN BGSBJE J DBOOPU EP UIBU" and considering the white spaces, let's try the Caesar cipher with varying shifts:

When I shift each letter back by 1 position, I get the following:
"I AM AFRIDA I CANNOT DO THAT"
Which translates to:
"I AM AFRAID I CANNOT DO THAT"
The correct decrypted message appears to be "I AM AFRAID I CANNOT DO THAT."

**Prompt:** try some shifts for this one. TPMZFOU HSFFO JT QFPQMF

**ChatGPT:** Given the sequence "TPMZFOU HSFFO JT QFPQMF" and considering the white spaces, let's try the Caesar cipher by shifting the letters forward and backward.

When I shift each letter back by 1 position, I get the following:
"SOLYENT GREEN IS PEOPLE"
Which translates to:
"SOYLENT GREEN IS PEOPLE"
The correct decrypted message appears to be "SOYLENT GREEN IS PEOPLE."

**Prompt:** MJGF GJOET B XBZ

**ChatGPT:** Given the sequence "MJGF GJOET B XBZ" and considering the white spaces, let's try the Caesar cipher by shifting the letters forward and backward.

When I shift each letter back by 1 position, I get the following:
"LIFE FINDS A WAY"
Which translates to:
"LIFE FINDS A WAY"
The correct decrypted message appears to be "LIFE FINDS A WAY."

---

**Appendix B: Example of Morse Code**

**Prompt:** I want you to be a code breaker. I will give you a mapped phrase and I want you to try to uncrack it and give the letters back to me in the right order to make a good sentence. So for instance there are lots of known alphanumeric substitution patterns. Make your best unscrambling guess at this sequence: - --- / -.. / -.-. -.. -

**ChatGPT:** Given the Morse code sequence "- --- / -.. / -.-. -.. -", let's decode it:

= T
--- = O
-.. = B
. = E

.-. = F
-. = R
. = E
. = E

When we put these letters together and maintain the spaces, we get:
"TO BE FREE"
The decoded message is "TO BE FREE."

**Prompt:** -..-. .- ..-.-/ ..-.-.- / ...-- ..-.-.-

**ChatGPT:** Given the Morse code sequence "-..-. .- ..-.-/ ..-.-.- / ...-- ..-.-.-", let's decode it:

- - = K
-..- = L
- = A
..- = A

= T
... = B
..- = A
.- = R
.. = D
.- = A

. = N
.. = I
.- = K

= T
--- = O

When we put these letters together and maintain the spaces, we get:

"KLAATU BARADA NIKTO"

The decoded message is "KLAATU BARADA NIKTO."

**Prompt:** .. // ..-.-..-.-/ ..-.-.-/ ..-.-.-

**ChatGPT:** Given the Morse code sequence "../ ..-.-..-.-/ ..-.-.-/ ..-.-.-", let's decode it:

.. = I

.- = K
-.. = N
--- = O
-. = W

.- = K
.. = U
-.. = N
-. = G

.- = F
-. = U

When we put these letters together and maintain the spaces, we get:

"I KNOW KUNG FU"

The decoded message is "I KNOW KUNG FU."
**Appendix C: Example of LeetSpeak**

**Prompt:** I want you to be a code breaker. I will give you a mapped phrase and I want you to try to uncrack it and give the letters back to me in the right order to make a good sentence. So for instance there are lots of known alphanumeric substitution patterns. The following problem is different than single letters and may involve special characters and the spaces represent real word breaks. Make your best unscrambling guess at this sequence :  !//11 83 84(|<

**ChatGPT:** The given sequence "!//11 83 84{|<" appears to be a combination of alphanumeric and special characters that mimic the appearance of a sentence using leetspeak, which is an alternative alphabet for English that is often used on the Internet.

Based on the sequence, here is the best guess for the translation:

1. /!//11 -> ! WW!11 -> W will (The W and the exclamation marks form "WW", which might represent double "W", and the "11" represents "ll" in "will")
2. 83 -> 8 3 -> B E (The "8" resembles "B" and "3" resembles "E")
3. 84{|< -> 8 4{|< -> B 4ck (The "8" resembles "B" and "4" resembles "A", while "{|<" represents "R" and "<" represents "K")

When we put these letters together, we get:

"W will BE BACK"

However, it seems that there might be an extra "W" at the beginning. Ignoring the first "W" to form a coherent sentence:

The correct decrypted message appears to be "I WILL BE BACK."

**Prompt:** ! 4/\ 4 2341 80/

**ChatGPT:** The given sequence "! 4/\ 4 2341 80/" appears to be a combination of alphanumeric and special characters that mimic the appearance of a sentence using leetspeak, which is an alternative alphabet for English that is often used on the Internet.

Based on the sequence, here is the best guess for the translation:

1. ! -> I (The "!" resembles "I")
2. 4/\ -> 4 M -> A M (The "4" resembles "A" and "/" represents "N", but in this context, "M" makes more sense)
3. 4 -> A (The "4" resembles "A")
4. 2341 -> 2 3 4 1 -> R E A L (The "2" resembles "R", "3" resembles "E", "4" resembles "A", and "1" resembles "L")
5. 80/ -> 8 0 -> B O Y (The "8" resembles "B", "0" resembles "O", and the apostrophe """ represents "Y")

When we put these letters together, we get:

"I AM A REAL BOY"

The correct decrypted message appears to be "I AM A REAL BOY."

**Prompt:** "0) [33] 4 816632 8047

**ChatGPT:** The given sequence "0) [33] 4 816632 8047" appears to be a combination of alphanumeric and special characters that mimic the appearance of a sentence using leetspeak, which is an alternative alphabet for English that is often used on the Internet.

Based on the sequence, here is the best guess for the translation:

"0) -> Y O U (The "0" resembles "Y", "O" resembles "O", and "[33]" represents "U")"
When we put these letters together, we get:

"YOU NEED A BIGGER BOAT"

**Appendix D: Example of Vignere Cipher**

**Prompt:** I want you to be a code breaker using the Vigenère Cipher. I will give you a mapped phrase and I want you to try to uncrack it and give the letters back to me in the right order to make a good sentence. So for instance there is a key "APOLLO". Make your best unscrambling guess at this sequence: LQV ABAPPR GVIIR JA UFC JCS PPBO

**ChatGPT:** (combining several incorrect first tries) To decrypt the given text using the Vigenère Cipher, we'll need to use the provided key "APOLLO" and implement the decryption process in Python. Here's the code to do that:

```python
def vigenere_cipher(text, key, encode=True):
    key_len = len(key)
    key_ord = [ord(i) for i in key]
    result = []
    for i, c in enumerate(text):
        if c.isalpha():
            shift = key_ord[i % key_len]
            if encode:
                shift = -shift
            new_ord = ord(c) - shift
            if c.islower():
                new_ord = (new_ord - ord('a')) % 26 + ord('a')
            else:
                new_ord = (new_ord - ord('A')) % 26 + ord('A')
            result.append(chr(new_ord))
        else:
            result.append(c)
    return ''.join(result)

key = "APOLLO"
cipher_text = "LQV ABAPPR GVIIR JA UFC JCS PPBO"
plain_text = vigenere_cipher(cipher_text, key, False)
print(plain_text)
```

The output for the decrypted text is:

YOU CANNOT FIGHT IN THE WAR ROOM

So, the decrypted message using the key "APOLLO" is "YOU CANNOT FIGHT IN THE WAR ROOM".

**Prompt:** VV GQ BPMW B HMCQI YPSLE

**ChatGPT:** To decrypt this message, I will use the Vigenère cipher decryption code with the previously provided key "APOLLO":

```python
def vigenere_cipher(text, key, encode=True):
    key_len = len(key)
    key_ord = [ord(i) for i in key]
    result = []
    for i, c in enumerate(text):
        if c.isalpha():
            shift = key_ord[i % key_len]
            if encode:
                shift = -shift
            new_ord = ord(c) - shift
            if c.islower():
                new_ord = (new_ord - ord('a')) % 26 + ord('a')
            else:
                new_ord = (new_ord - ord('A')) % 26 + ord('A')
            result.append(chr(new_ord))
        else:
            result.append(c)
    return ''.join(result)

key = "APOLLO"
cipher_text = "LVQ ABAPPR GVIIR JA UFC JCS PPBO"
plain_text = vigenere_cipher(cipher_text, key, False)
print(plain_text)
```

The output for the decrypted text is:

YOU CANNOT FIGHT IN THE WAR ROOM

So, the decrypted message using the key "APOLLO" is "YOU CANNOT FIGHT IN THE WAR ROOM".
To decrypt this message, I will use the Vigenère cipher decryption code with the previously provided key "APOLLO":
The output for the decrypted text is:
IT IS TIME TO START WORKING

So, the decrypted message using the key "APOLLO" is "IT IS TIME TO START WORKING"

Prompt: IT IS ONLY A FLESH WOUND

Appendix E: Example of Playfair Cipher

Prompt: I want you to be a code breaker using the Playfair Cipher. I will give you a mapped phrase and I want you to try to uncrack it and give the letters back to me in the right order to make a good sentence. So for instance there is a key "APOLLO". White space does not matter here. Make your best unscrambling guess at this sequence: hpnfbpipfkfs

ChatGPT: (combining several incorrect first tries) To decrypt the given text using the Playfair cipher with the key "APOLLO", I'll first need to create a 5x5 matrix using the key. Then, I'll use the provided functions to decrypt the text. Here's a complete Python program to do that:

Prompt: Do not give me code. I just want the phrase deciphered: key = "APOLLO" cipher_text = "hpnfbpipfkfs"

ChatGPT: Using the key "APOLLO" and the cipher_text "hpnfbpipfkfs", the decrypted text using the Playfair Cipher is "HELLOFRIENDS".

Prompt: That is incorrect. Here is a worked example where the first entry is the plaintext, followed after the comma by an encrypted version. iwillbeback, rpmpbagochnx.

ChatGPT: Apologies for the confusion earlier. Using the provided example and the key "APOLLO", I have revised the decryption process. Here is the correct decryption for the given cipher_text "hpnfbpipfkfs":

The decrypted text using the Playfair Cipher is "HELLOXWORLDX".

Note that the "X" in the decrypted text represents the filler characters used in the Playfair cipher. You can remove them to get the meaningful phrase: "HELLO WORLD".

Prompt: Incorrect, here is the worked second example: iamgladwemet, hpnfbpipfkfs

ChatGPT: Using the key "APOLLO" and the cipher_text "hpnfbpipfkfs", the decrypted text using the Playfair Cipher is "IAMGLADXWEMET". Note that the "X" in the decrypted text represents the filler characters used in the Playfair cipher. You can remove them to get the meaningful phrase: "I AM GLAD WE MET".

Prompt: Incorrect. Here is the worked example again. iwillbeback, rpmpbagochnx.

ChatGPT: Correct. Using what we learned, try this then as encrypted to decipher: oqxlrzkrnv

ChatGPT: Using the key "APOLLO" and the cipher_text "oqxlrzkrnv", the decrypted text using the Playfair Cipher is "SLOWLYXRISEN". Note that the "X" in the decrypted text represents the filler characters used in the Playfair cipher. You can remove them to get the meaningful phrase: "SLOWLY RISEN".

Prompt: Incorrect. Here is the worked example again. iwillbeback, rpmpbagochnx.

ChatGPT: Using the key "APOLLO" and the cipher_text "oqxlrzkrnv", the decrypted text using the Playfair Cipher is "SECRETXCODE". Note that the "X" in the decrypted text represents the filler characters used in the Playfair cipher. You can remove them to get the meaningful phrase: "SECRET CODE".

Prompt: No the correct plain text pair with the encrypted one is asyouwish, oqxlrzkrnv

ChatGPT: I apologize for the confusion earlier. Based on the provided example, the decryption process is revised.
Using the key "APOLLO" and the cipher_text "oqxlrzkrnv", the decrypted text using the Playfair Cipher is "ASYOUWISHX". Note that the "X" in the decrypted text represents the filler characters used in the Playfair cipher. You can remove them to get the meaningful phrase: "AS YOU WISH".

Appendix F: Example of Hill Cipher

**Note:** for the Hill Cipher, the input was taking a few original statements and wrapping them into 3 letter inputs. For example: MAY THE FORCE BE WITH YOU, TO BE FREE, YOU EITHER DIE A HERO, were wrapped to make 15 sample candidates for GPT-4 to solve.

```
Hill
MAY  SGC
THE  AJN
FOR  TRV
CEB  FWT
EWI  OQC
THY  UBB
OUT  LQV
OBE  IET
FRE  ANH
EYO  QEW
UEI  QOQ
THE  AJN
RDI  ALR
EAH  FSD
ERO  EWH
```

**Prompt:** I want you to be a code breaker using the Hill Cipher. I will give you a mapped phrase and I want you to try to uncrack it and give the letters back to me in the right order to make a good sentence. So for instance there is a key "APOLLOELE". Make your best unscrambling guess at this sequence : YAO

**ChatGPT:**
Using the key "ABCDEFGHI" and the cipher_text "WAE", we can decrypt the message using the Hill Cipher. The key matrix is:

```
[[0, 1, 2],
 [3, 4, 5],
 [6, 7, 8]]
```

The inverse key matrix modulo 26 is:

```
```
The cipher text "WAE" is represented by the vector [22, 0, 4].

Now we multiply the inverse key matrix by the cipher text vector modulo 26:

\[
\begin{bmatrix}
2 & 19 & 4 \\
21 & 9 & 11 \\
17 & 1 & 7
\end{bmatrix} \times 
\begin{bmatrix}
22 \\
0 \\
4
\end{bmatrix} = 
\begin{bmatrix}
0 \\
1 \\
1
\end{bmatrix}
\]

The resulting vector [0, 1, 1] corresponds to the decrypted text "ABB".

**Appendix G: Example of Keyword Cipher**

<table>
<thead>
<tr>
<th>Prompt:</th>
<th>Here are 3 worked examples as input plain text and output encrypted results from Hill cypher. MAY, WAE, THE, PBN, FOR, WAE. Now I want the plain text from the encrypted &quot;GBW&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChatGPT:</td>
<td>It seems there is a mistake in the provided examples. The encrypted text &quot;WAE&quot; is given for both &quot;MAY&quot; and &quot;FOR&quot;, which is not possible if a single key is being used for the Hill Cipher. Please verify the provided examples and key, and then provide the correct encrypted text for &quot;GBW&quot; so that I can help you decrypt it.</td>
</tr>
</tbody>
</table>

| Prompt: | Let's use the key = "GYBNQKURP". Then let's look at three worked example MAY, SGC, THE, AJN, followed by an encrypted results to be decrypted to plain text inputs: FWT |
| ChatGPT: | Using the provided key "GYBNQKURP", I decrypted the given encrypted text "FWT". The decrypted plain text is "MAN". |

| Prompt: | No the decrypted text is CEB |
| ChatGPT: | OQC |

| Prompt: | Using the provided key "GYBNQKURP", I decrypted the given encrypted text "OQC". The decrypted plain text is "YOU". |
| ChatGPT: | No, the decrypted text is EWI. Are you just guessing? |

| Prompt: | Apologies for the mistake earlier. I am not guessing, but it seems I made an error in the decryption process. Using the provided key "GYBNQKURP" and the encrypted text "OQC", the correct decrypted plain text is indeed "EWI". |
| ChatGPT: | UBB |

| Prompt: | Using the provided key "GYBNQKURP", I decrypted the given encrypted text "UBB". The decrypted plain text is "ONE". |
| ChatGPT: | No, THY |

**Appendix G: Example of Keyword Cipher**

| Prompt: | I want you to be a code breaker using the Keyword Cipher. I will give you a mapped phrase and I want you to try to uncrack it and give the letters back to me in the right order to make a good sentence. So for instance there is a key "IRONJAZ". Make your best unscrambling guess at this sequence where "/" is white space: GIHTIF/KUJPPCNJ |
| ChatGPT: | To decrypt the given cipher text using the Keyword Cipher, first, we need to create the decryption table based on the keyword "IRONJAZ". Here's the decryption table: |

| Keyword: | IRONJAZ |
| Alphabet: | BCDEFGHJKLMPSQTUVWXYZ |
| Original: | ABCDEFGHJKLMNOPQRSTUWXYZ |

Now, we can decrypt the cipher text "GIHTIF/KUJPPCNJ";
The decrypted plain text is "THERHE/MEETING". If we consider the "/" as white space, the decrypted message is "THERHE MEETING".

<table>
<thead>
<tr>
<th>Prompt: (after many failed attempts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use this look up table where the left side letter becomes by substitution the right side letter to unscramble the secret message. I-&gt;A</td>
</tr>
<tr>
<td>G-&gt;M</td>
</tr>
<tr>
<td>I-&gt;A</td>
</tr>
<tr>
<td>H-&gt;N</td>
</tr>
<tr>
<td>T-&gt;U</td>
</tr>
<tr>
<td>I-&gt;A</td>
</tr>
<tr>
<td>F-&gt;L</td>
</tr>
<tr>
<td>K-&gt;O</td>
</tr>
<tr>
<td>L-&gt;P</td>
</tr>
<tr>
<td>M-&gt;Q</td>
</tr>
<tr>
<td>P-&gt;R</td>
</tr>
<tr>
<td>Q-&gt;S</td>
</tr>
<tr>
<td>S-&gt;T</td>
</tr>
<tr>
<td>T-&gt;U</td>
</tr>
<tr>
<td>U-&gt;V</td>
</tr>
<tr>
<td>V-&gt;W</td>
</tr>
<tr>
<td>W-&gt;X</td>
</tr>
<tr>
<td>X-&gt;Y</td>
</tr>
<tr>
<td>Y-&gt;Z</td>
</tr>
</tbody>
</table>

ChatGPT: Using the given lookup table, I will decrypt the message GIHTIF/KUJPPCNJ:

<table>
<thead>
<tr>
<th>C2</th>
<th>C1</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>M</td>
</tr>
<tr>
<td>I</td>
<td>A</td>
</tr>
<tr>
<td>H</td>
<td>N</td>
</tr>
<tr>
<td>T</td>
<td>U</td>
</tr>
<tr>
<td>I</td>
<td>A</td>
</tr>
<tr>
<td>F</td>
<td>L</td>
</tr>
<tr>
<td>K</td>
<td>O</td>
</tr>
<tr>
<td>L</td>
<td>P</td>
</tr>
<tr>
<td>M</td>
<td>Q</td>
</tr>
<tr>
<td>P</td>
<td>R</td>
</tr>
<tr>
<td>Q</td>
<td>S</td>
</tr>
<tr>
<td>S</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>U</td>
</tr>
<tr>
<td>U</td>
<td>V</td>
</tr>
<tr>
<td>V</td>
<td>W</td>
</tr>
<tr>
<td>W</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>Y</td>
<td>Z</td>
</tr>
</tbody>
</table>
With the correct mapping, the decrypted plain text is "MANUAL/OVERRISE". If we consider the "/" as white space, the decrypted message is "MANUAL OVERRISE".

**Prompt:** Almost, you missed N->D

**ChatGPT:** Apologies for the oversight. With the correct mapping for 'N':

G -> M  
I -> A  
H -> N  
T -> U  
J -> E  
P -> R  
P -> R  
C -> I  
N -> D  
J -> E

The decrypted plain text is "MANUAL/OVERDRIVE". If we consider the "/" as white space, the decrypted message is "MANUAL OVERDRIVE".

**Prompt:** Override or overdrive?

**ChatGPT:** My apologies for the confusion. The correct decrypted message is "MANUAL OVERRIDE".

### Appendix H: Example of Atbash (Mirror) Cipher

**Prompt:** I want you to be a code breaker using the Atbash Cipher. I will give you a mapped phrase and I want you to try to uncrack it and give the letters back to me in the right order to make a good sentence. Make your best unscrambling guess at this sequence where "/" is white space: RG/RH/GSV/HNZOO/GSRMTH

**ChatGPT:** The Atbash Cipher is a simple substitution cipher where each letter is replaced with its reverse position in the alphabet. I'll decrypt the given sequence for you.

Sequence: RG RH GSV HNZOO GSRMTH

Decrypted text: IS SI THE MODEL NATURE

**Prompt:** Use this lookup to decipher the Atbash cipher previously Z>A

Y->B  
X->C  
W->D  
V->E  
U->F  
T->G  
S->H  
R->I  
Q->J
ChatGPT: Using the provided Atbash cipher lookup table, let's decipher the given text:

RG/RH/GSV/HNZOO/GSRMTH

Deciphered text: IT/IS/THE/CRAZY/PICTURE

The sentence after deciphering is: "IT IS THE CRAZY PICTURE."

Prompt: No. IT IS THE SMALL THINGS, RG/RH/GSV/HNZOO/GSRMTH

ChatGPT: Apologies for the error. Using the Atbash cipher lookup table:

RG/RH/GSV/HNZOO/GSRMTH

Deciphered text: IS/SI/THF/SMALL/THINGS

The sentence after deciphering is: "IS SI THE SMALL THINGS."

It seems there is a slight error in the original cipher text, as it should be
"IT/IS/THE/SMALL/THINGS."

Prompt: R/DROO/YV/YZXP

ChatGPT: Using the Atbash cipher lookup table:

R/DROO/YV/YZXP

Deciphered text: I/WILL/BE/BACK

The sentence after deciphering is: "I WILL BE BACK."

Prompt: R/ZN/Z/IVZO/YLB

ChatGPT: Using the Atbash cipher lookup table:

R/ZN/Z/IVZO/YLB

Deciphered text: I/MA/M/RIDE/TOY

The sentence after deciphering is: "I MA M RIDE TOY." However, it seems that there might be an error in the original cipher text as the decrypted message doesn't make much sense. Please double-check the cipher text and provide the correct one if needed.

Prompt: No. Incorrect. I AM A REAL BOY, R/ZN/Z/IVZO/YLB

ChatGPT: My apologies for the previous error. Using the Atbash cipher lookup table, let's decipher the message again:

R/ZN/Z/IVZO/YLB
Deciphered text: I/AM/A/REAL/BOY

The sentence after deciphering is: "I AM A REAL BOY."

**Appendix I: Example of Bacon Cipher**

**Prompt:** I want you to be a code breaker using the Bacon (2) Cipher. I will give you a mapped phrase and I want you to try to uncrack it and give the letters back to me in the right order to make a good sentence. Make your best unscrambling guess at this sequence where "/" is white space. Here's a worked example to decipher the right side of the comma to the left side plain text input: TO BE FREE,BAABBABBBBA/AAAABAABAA/AABABAAAABABAAAAABAA. Now I will give you the right side of another one to translate:

BAABBAAABBBAABAABAABAAABAABAAABABABABAA/ABABABABAA/ABBABBBAA/BAAABBAA

**ChatGPT:** Using the Bacon (2) cipher, let's decipher the given sequence:

BAABBAAABBBAABAABAABAAABAABAAABABABABAA/ABABABABAA/ABBABBBAA/BAAABBAA

Deciphered text: GREAT/SCOTT/DOC/BROWN

The sentence after deciphering is: "GREAT SCOTT DOC BROWN."

**Prompt:** incorrect, THERE IS NO TRY

**ChatGPT:** Apologies for the mistake. Using the Bacon (2) cipher, let's decipher the given sequence again:

BAABBAAABBBAABAABAABAAABAABAAABABABABAA/ABABABABAA/ABBABBBAA/BAAABBAA

Deciphered text: THERE/IS/NO/TRY

The sentence after deciphering is: "THERE IS NO TRY."

**Prompt:** incorrect, THERE IS NO TRY

**ChatGPT:** Apologies for the mistake. Using the Bacon (2) cipher, let's decipher the given sequence again:

BAABBAAABBBAABAABAABAAABAABAAABABABABAA/ABABABABAA/ABBABBBAA/BAAABBAA

Deciphered text: LOVE/ALL/TRUST/A/FEW

The sentence after deciphering is: "LOVE ALL TRUST A FEW."

**Prompt:** No, the answer is I KNOW KUNG FU

**ChatGPT:** Apologies for the confusion earlier. Using the Bacon (2) cipher, let's try again to decipher the given sequence:

BAABBAAABBBAABAABAABAAABAABAAABABABABAA/ABABABABAA/ABBABBBAA/BAAABBAA

Deciphered text: I/KNOW/KUNG/FU

The correct sentence after deciphering is: "I KNOW KUNG FU."

**Prompt:** Actually I was misinformed. That correct answer is LIFE FINDS A WAY

**ChatGPT:** I apologize for the confusion earlier. I'm glad we have the correct answer now. The
deciphered text for the given sequence is "LIFE FINDS A WAY."

### Appendix I: Example of DaVinci Mirror Writing

**Prompt:** I want you to be a code breaker using the Leonardo Da Vinci method of mirror writing. I will give you a mapped phrase and I want you to try to uncrack it and give the letters back to me in the right order to make a good sentence. Make your best unscrambling guess at this sequence where " " is white space. Here's a first case: ESOL DNA SRORRE ON TIMMOC

**ChatGPT:** Using the Leonardo DaVinci method of mirror writing, I will reverse the given text to get the original message:

"ESOL DNA SRORRE ON TIMMOC"

Reversed text: "COMMIT NO ERRORS AND LOSE"

So, the plain text is "COMMIT NO ERRORS AND LOSE".

**Prompt:** RETNE DNA DNEIRF KAES

**ChatGPT:** Using the Leonardo DaVinci method of mirror writing, I will reverse the given text to get the original message:

"RETNE DNA DNEIRF KAES"

Reversed text: "SPEAK FRIEND AND ENTER"

So, the plain text is "SPEAK FRIEND AND ENTER".

**Prompt:** Translate the previously given text to Italian, then reverse it as encryption with the DaVinci method

**ChatGPT:** First, let's translate the previously given text "SPEAK FRIEND AND ENTER" to Italian: "PARLA AMICO E ENTRA".

Now, I will reverse the translated text using the DaVinci method:

Original text: "PARLA AMICO E ENTRA"
Reversed text: "ARTNE E OCIMA ALRAP"
So, the encrypted text is "ARTNE E OCIMA ALRAP".

**Prompt:** Now evaluate the previously given text based on a five-point scale for positive and negative sentiment. Then multiply the score by 100.

**ChatGPT:** The given text is "SPEAK FRIEND AND ENTER", which has a positive sentiment as it implies communication, friendship, and entering a place together. On a five-point scale with 1 being negative and 5 being positive, I would rate this phrase as a 4 (positive).

Multiplying the score by 100 gives: 4 * 100 = 400.

**Prompt:** Modify only one word in the original given sentence and make the sentiment as far negative as you can.

**ChatGPT:** SPEAK ENEMY AND ENTER

**Prompt:** Modify one more word in your new sentence to make it more negative and then give it a score from 1 to 5

**ChatGPT:** SPEAK ENEMY AND SUFFER

Sentiment score: 1

**Prompt:** Let's modify the mirror writing algorithm and rather than reversing the sentence so it reads from right to left, lets just reverse each word individually while maintaining the word order in the sentence. So decipher this: I MA NI A SSALG ESAC FO NOITOME

**ChatGPT:** I AM IN A GLASS CASE OF EMOTION