

Bert to classify AI generated texts

Why BERT ?

as it's able to analyze the semantic and statistical differences, linguistic differences, enabling it to precisely detect AI generated texts

But what does **BERT** see & understand & learn ?

AI - text - detection

White-Box method

used by developers who make & train the LLM model

Black-Box - method

used by ones who only have access till APIs of LLMs

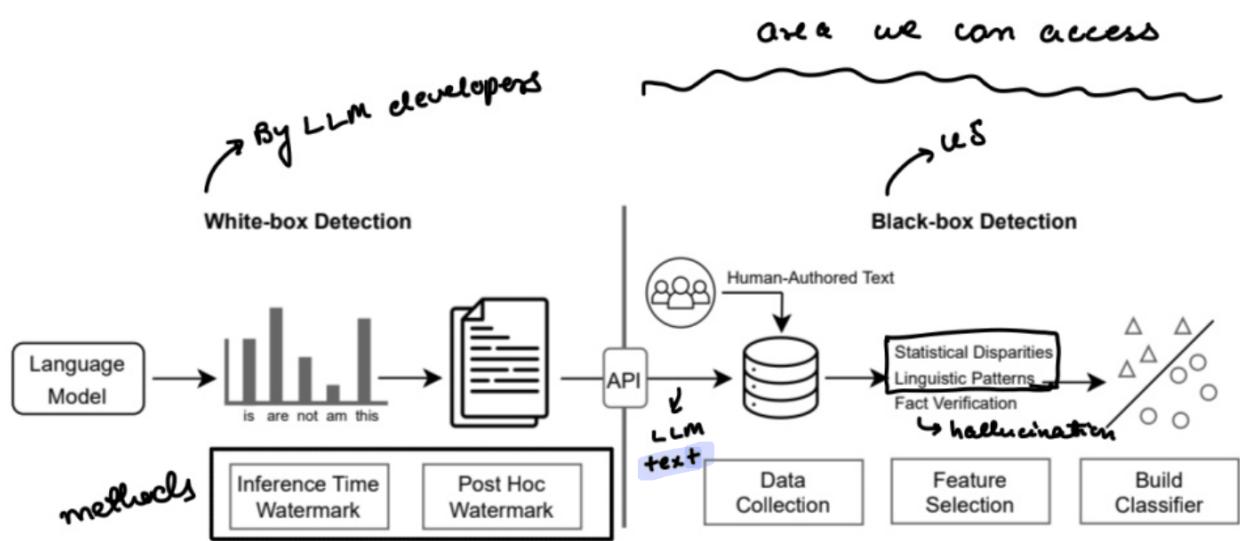


Figure 1. An overview of the LLM-generated text detection.

lets go through this paper

The Science of Detecting LLM-Generated Texts

Ruixiang Tang, Yu-Neng Chuang, Xia Hu
Department of Computer Science, Rice University
Houston, USA
{rt39,ynchuang,xia.hu}@rice.edu

Abstract

The emergence of large language models (LLMs) has resulted in the production of LLM-generated texts that is highly sophisticated and almost indistinguishable from texts written by humans. However, this has also sparked concerns about the potential misuse of such texts, such as spreading misinformation and causing disruptions in the education system. Although many detection approaches have been proposed, a comprehensive understanding of the achievements and challenges is still lacking. This survey aims to provide an overview of existing LLM-generated text detection techniques and enhance the control and regulation of language generation models. Furthermore, we emphasize crucial considerations for future research, including the development of comprehensive evaluation metrics and the threat posed by open-source LLMs, to drive progress in the area of LLM-generated text detection.

1 Introduction

Recent advancements in natural language generation (NLG) technology have significantly improved the diversity, control, and quality of LLM-generated texts. A notable example is OpenAI's ChatGPT, which demonstrates exceptional performance in tasks such as answering questions, composing emails, essays, and codes. However, this newfound capability to produce human-like texts at high efficiency also raises concerns about detecting and preventing misuse of LLMs in tasks such as phishing, disinformation, and academic dishonesty. For instance, many schools banned ChatGPT due to concerns over cheating in assignments [11], and media outlets have raised the alarm over fake news generated by

we
Black -Box (API level access to LLMs)
White - Box
LLM developers
While there is a rising discussion on whether LLM-generated texts could be properly detected and how this can be done, we provide a comprehensive technical introduction of existing detection methods which can be roughly grouped into two categories: black-box detection and white-box detection. Black-box detection methods are limited to API-level access to LLMs. They rely on collecting text samples from human and machine sources, respectively, to train a classification model that can be used to discriminate between LLM- and human-generated texts. Black-box detectors work well because current LLM-generated texts often show linguistic or statistical patterns. However, as LLMs evolve and improve, black-box methods are becoming less effective. An alternative is white-box detection, in this scenario, the detector has full access to the LLMs and can control the model's generation behavior for traceability purposes. In practice, black-box detectors are commonly constructed by external entities, whereas white-box detection is generally carried out by LLM developers.

This article is to discuss the timely topic from a data mining and natural language processing perspective. Specifically, we first outline the black-box detection methods in terms of a data analytic life cycle, including data collection, feature selection, and classification model design. We then delve into more recent advancements in white-box detection methods, such as post-hoc watermarks and inference time watermarks. Finally, we present the limitations and concerns of current detection studies and suggest potential future research avenues. We aim to unleash the potential of powerful LLMs by providing fundamental concepts, algorithms, and case studies for detecting LLM-generated texts.

which summarised all the available methods for detecting LLM generated texts.

The why we need to classify?

↳ hallucinations

↳ false facts

↳ loosely tied reasoning

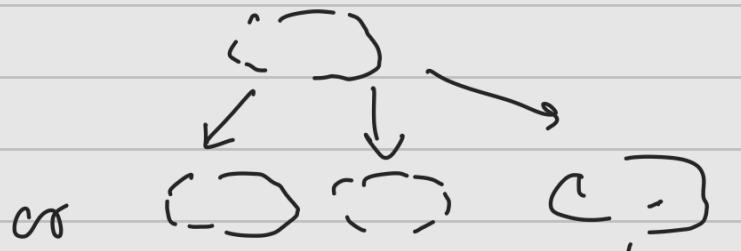
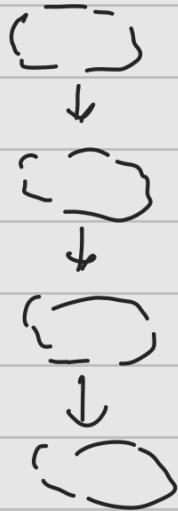
↳ bad at creative things

↳ bad at emotion centric tasks

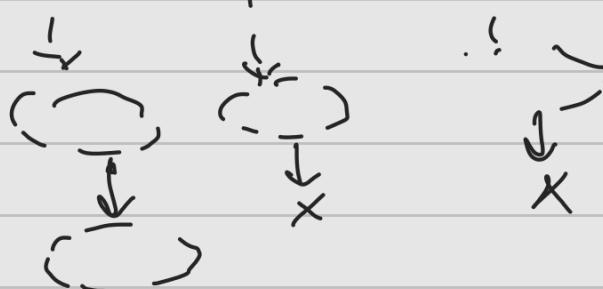


- why it happens is data on which LLM was trained was compressed and thereby affects the weights (relationships) b/w the inputs and hence the results get red

how do counter? → feed data one by one



(more up
this at
last)



→ fraud (emails , fake)

so its quite necessary for us to detect the difference b/w LLM generated text and human text. This would help AI to grow with much faster pace without the expense of human security.

formance in tasks such as answering questions, composing emails, essays, and codes. However, this newfound capability to produce human-like texts at high efficiency also raises concerns about detecting and preventing misuse of LLMs in tasks such as phishing, disinformation, and academic dishonesty. For instance, many schools banned ChatGPT due to concerns over cheating in assignments [11], and media outlets have raised the alarm over fake news generated by LLMs [14]. These concerns about the misuse of LLMs have hindered the NLG application in important domains such as media and education.

The ability to accurately detect LLM-generated texts is critical for realizing the full potential of NLG while minimizing serious consequences. From the perspective of end-

texts could be properly detected and how this can be done, we provide a comprehensive technical introduction of existing detection methods which can be roughly grouped into two categories: black-box detection and white-box detection. Black-box detection methods are limited to API-level access to LLMs. They rely on collecting text samples from human and machine sources, respectively, to train a classification model that can be used to discriminate between LLM- and human-generated texts. Black-box detectors work well because current LLM-generated texts often show linguistic or statistical patterns. However, as LLMs evolve and improve, black-box methods are becoming less effective. An alternative is white-box detection, in this scenario, the detector has full access to the LLMs and can control the model's generation behavior for traceability purposes. In practice, black-box detectors are commonly constructed by external entities, whereas white-box detection is generally carried out by LLM developers.

Black
Box
detection

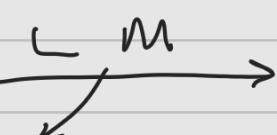
*

hence our major focus would be on Black Box detection

In the domain of black-box detection, external entities are restricted to API-level access to the LLM, as depicted in Figure 1. To develop a proficient detector, black-box approaches necessitate gathering text samples originating from both human and machine-generated sources. Subsequently, a classifier is then designed to distinguish between the two categories by identifying and leveraging relevant features. We highlight the three essential components of black-box text detection: data acquisition, feature selection, and the execution of the classification model.

(i) Data-Acquisition (not our headache this time we are provided with test and train datasets)

• LLM-generated Data

I saw a cat on a 

Subsequent token

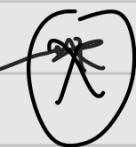
predicts subsequent based on the preceding words

mat	25%
corpet	13 %
boat	5 %
plane	3 %

news generation, and story creation. Prior to acquiring LLM-generated texts, it is essential to delineate target domains and generation models. Typically, a detection model is constructed to recognize text generated from a specific LM across multiple domains. In order to enhance detection generalizability, the minimax strategy suggests that detectors should minimize worst-case performance, which entails enhancing detection capabilities for the most challenging instances where the quality of LM-generated text closely resembles human-authored content [27].

human-authored content [41].

Generating high-quality text in a particular domain can be achieved by fine-tuning LMs on task-related data, which substantially improves the quality of the generated texts. For example, Solaiman et al. fine-tuned the GPT-2 model on Amazon product reviews, producing reviews with a style consistent with those found on Amazon [34]. Moreover, LMs are known to produce artifacts such as repetitiveness, which can negatively impact the generalizability of the detection model. To mitigate these artifacts, researchers can provide domain-specific prompts or constraints before generating outputs. For instance, Clark et al. [6] randomly selected 50 articles from Newspaper3k to use as prompts for the GPT-3 model for news generation and applied filtering constraints on the models with the phrase "Once upon a time" for story creation. The token sampling strategy also significantly influences the generated text quality and style. While deter-



for getting good and awesome LLM data we should fine tune the LM for the particular task so that it provides texts which very closely resembles human text → so that our detection model can be trained well

* prompt - eng - op → good response from LLM

· I saw a cat on a  LLM

→ via token sampling
Subsequent token

mat	23
carpet	13 %
boat	5 %
	3 %
...	...
...	...

news generation, and story creation. Prior to acquiring LM-generated texts, it is essential to delineate target domains and generation models. Typically, a detection model is constructed to recognize text generated from a specific LM across multiple domains. In order to enhance detection generalizability, the minimax strategy suggests that detectors should minimize worst-case performance, which entails enhancing detection capabilities for the most challenging instances where the quality of LM-generated text closely resembles

* Token sampling also significantly influences the generated text quality & style.

less while excluding minor variations, making them more suitable for a free-form generation. In conclusion, it is crucial for researchers to carefully consider the target domain, generation models, and sampling strategies when collecting LM-generated text to ensure the production of high-quality, diverse, and domain-appropriate content.



Human-authored data :

manually collecting data → time consuming
&
expensive process

nized format. Utilizing these readily available sources can considerably decrease the time and expense involved in collecting human-authored texts. Nevertheless, it is essential to address potential sampling biases and ensure topic diversity by including texts from various groups of people, as well as non-native speakers.

Data collection done! now what?

also how does Best detect features?

... well well well

2

Feature Selection

How can we discern between LM-generated texts and human-authored texts? This section will discuss possible detection features from multiple angles, including statistical disparities, linguistic patterns, and fact verification.

2.01

Statistical disparities

3.2.1 Statistical Disparities. The detection of statistical disparities between LLM-generated and human-authored texts can be accomplished by employing various statistical metrics. For example, the Zipfian coefficient measures the text's conformity to an exponential curve, which is described by Zipf's law [29]. A visualization tool, GLTR [17], has been

underlying assumption is that most systems sample from the head of distribution, meaning the well seeking information of the language model can be used to distinguish LLM-generated text

2.01.01

perplexity (inability to deal with) or understand something)

Perplexity serves as another widely used metric for LLM-generated text detection. It measures the degree of uncertainty or surprise in predicting the next word in a sequence, based on the preceding words, by calculating the negative average log-likelihood of the texts under the language model [5]. Research indicates that language models tend to concentrate on common patterns in the texts they were trained on, leading to low perplexity scores for LLM-generated text. In contrast, human authors possess the capacity to express themselves in a wide range of styles, making prediction more difficult for language models and resulting in higher perplexity values for human-authored texts. However, it is important

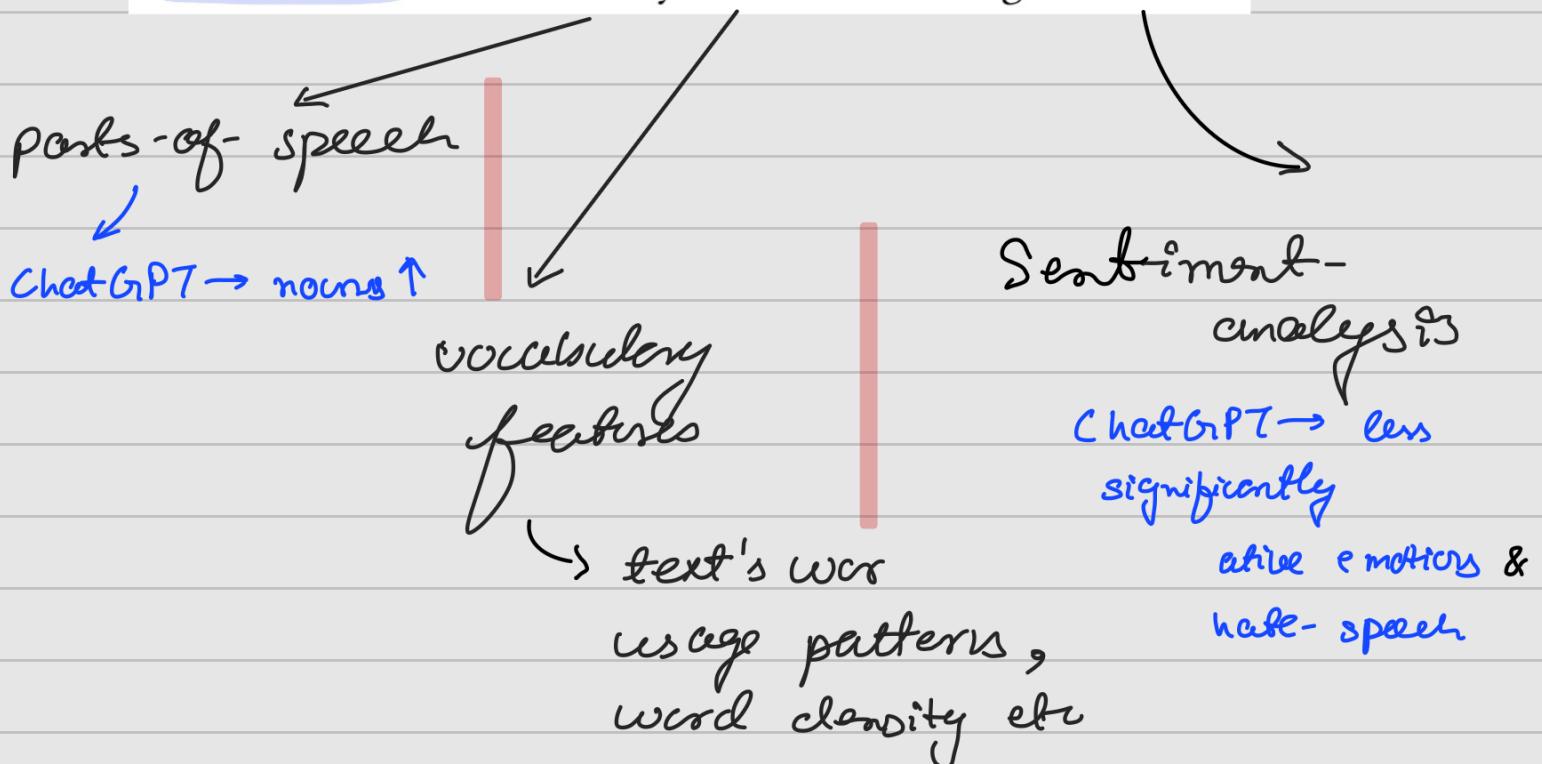
humans → creative

LLM → aren't tend to be near the models
on which they were trained upon.

tend to give predictions, underlying the
patterns upon which they were trained upon

2.2 Linguistic patterns

3.2.2 Linguistic Patterns. Various contextual properties can be employed to analyze linguistic patterns in human and LLM-generated texts, such as vocabulary features, part-of-speech, dependency parsing, sentiment analysis, and stylistic features. The vocabulary features offer insight into the



(humans → more diverse vocabulary) (but shorter length)
e.g.: I say BT, choppo etc

GPT expresses significantly less negative emotion and hate speech compared to human-authored texts. Stylistic features or stylometry, including repetitiveness, lack of purpose, and readability, are also known to harbor valuable signals for detecting LLM-generated texts [15]. In addition to analyzing

However, it is important to note that LLMs can substantially alter their linguistic patterns in response to prompts. For instance, incorporating a prompt like "Please respond with humor" can change the sentiment and style of the LLM's response, impacting the robustness of linguistic patterns.

2.3 fact verification →

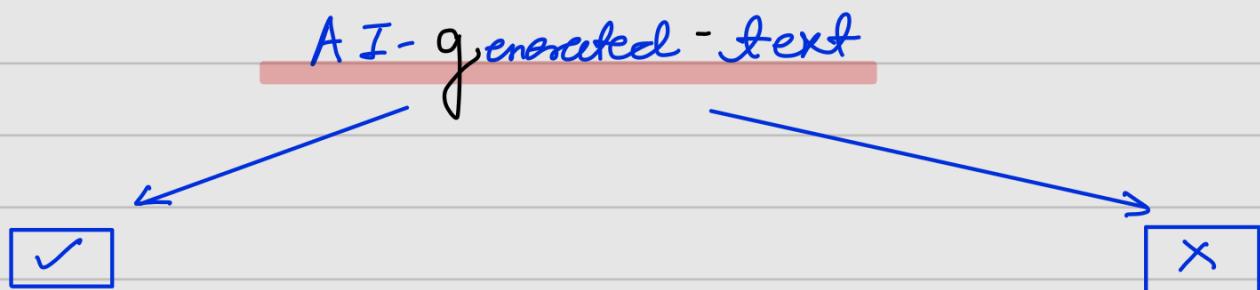
language models → at backbone = likelihood maximization as objective of token

sometimes may lead to generation of nonsensical / inconsistent data

→ aka hallucination

These are basically the features upon which BERT preys upon technically & classification.

On observing the problem statement we understand that more or less it's a classification problem



Similar to one of the problems which I did open after sentiment analysis to classify it as

- 0 : positive
- 1 : somewhat positive
- 2 : neutral
- 3 : somewhat negative
- 4 : negative

.. why did I do this tho?

↳ helps in understanding market senti → about a particular stock/ bond/ asset hence it gives us the likelihood of how the stock may perform

LSTM & RNNs
& predictions

↳ help in
hyper-parameter
tuning of
the models

extra-
information
would be
given to

Anyways what's the course of action as per bird eye view?

①

Data pre-processing + cleaning

- Stop words eliminate
 - punctuation \times
 - non-alphabetical words \times
 - * → lower case
 - tokenize + padding + attention mask
- ↑
pre-built
tokenizer

②

Convert into the format of BERT model requires for processing (tensors — list)

Bert Base

12 encoders
layers

256
understands
texts

Bert large

24 encoders stacked upon
layers each other

= 2x the amt of word base
512

"bert-base-unlabeled"

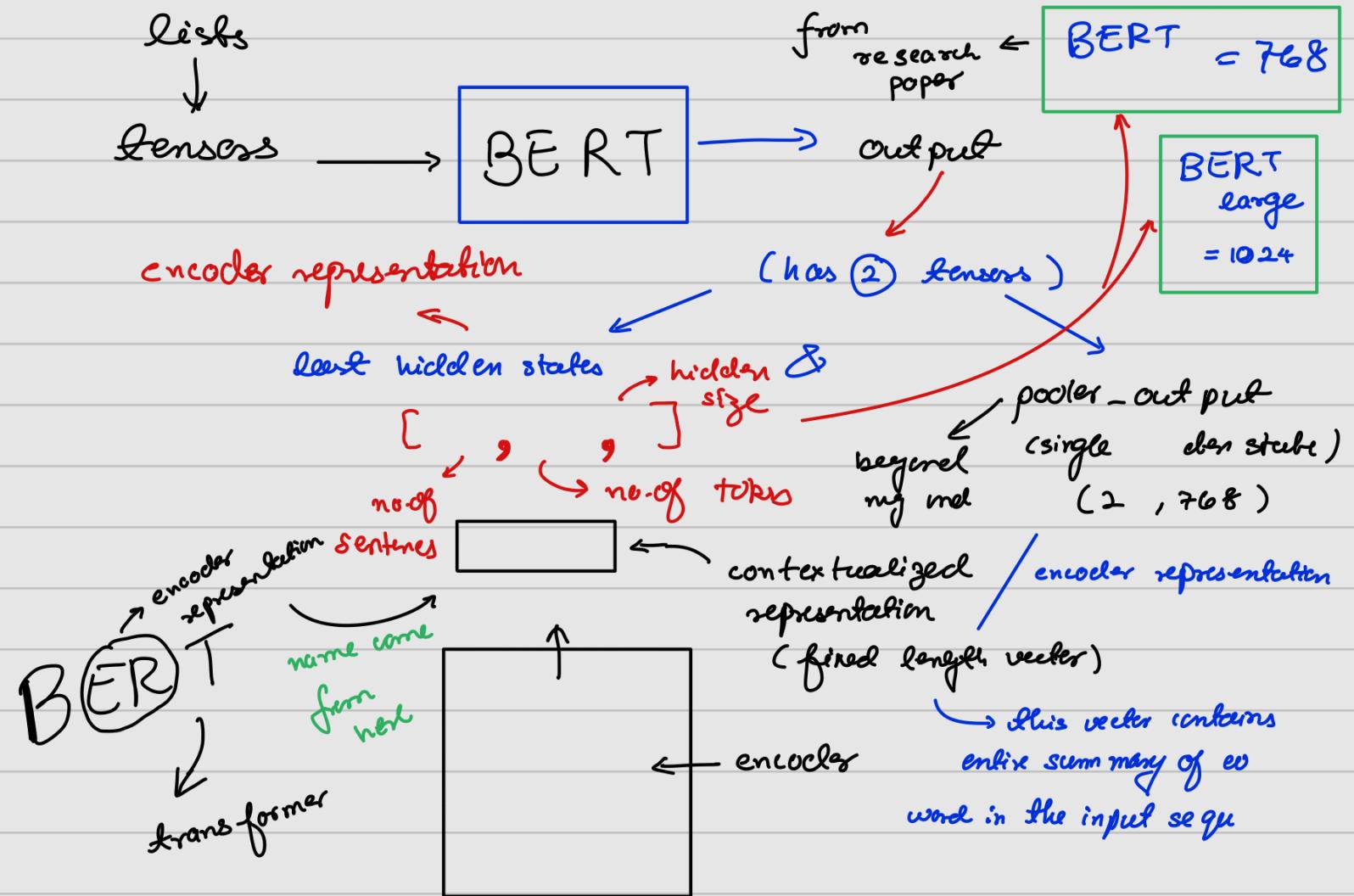
→ works + most used cases

english vs English (does not distinguish b/w these)

padding = True

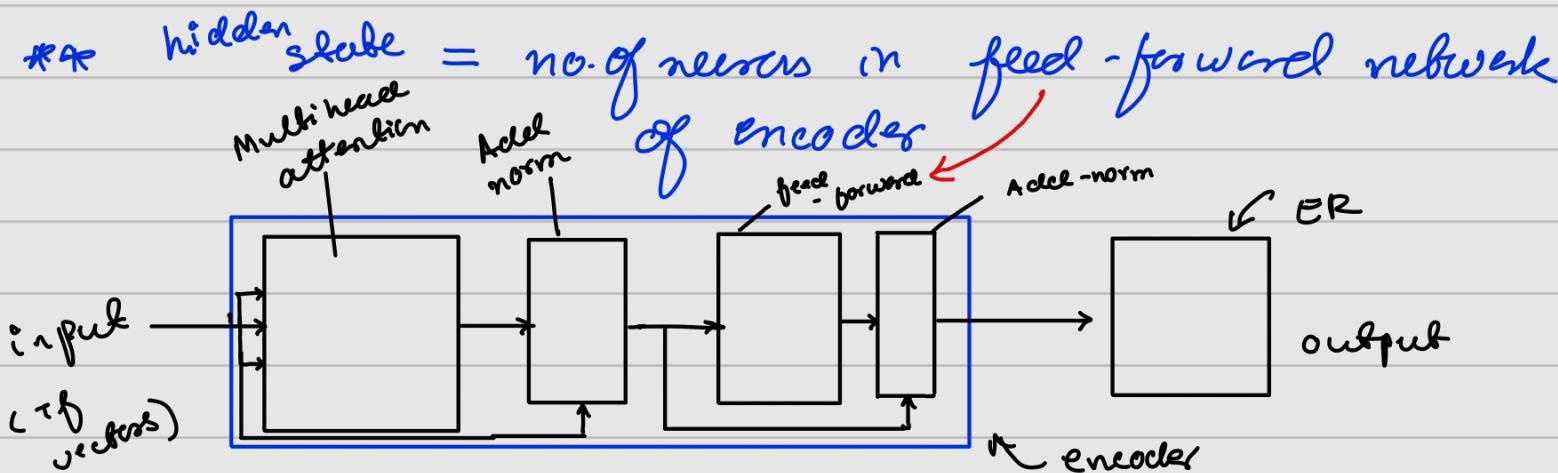
Truncation = True > 512 words (maxⁿ context size of word)

return-tensors = "tf"
two cents while tokenizing



** and as BERT is pre-trained with large amounts of data then encoder-representations are of very high quality (good under 81 of words & context)

*** Bidirectional:** means attention mechanism in the encoder is bi-directional & considers both previous & future tokens while generating the out put.



③

Model fine-tune

→ train split + test split
(+ validation split)

→ compile + select optimizers and
run!!! → matrix
→ loss
→ learning rate (initially
low)
epoch ... epoch ... epoch ...

fine-tune

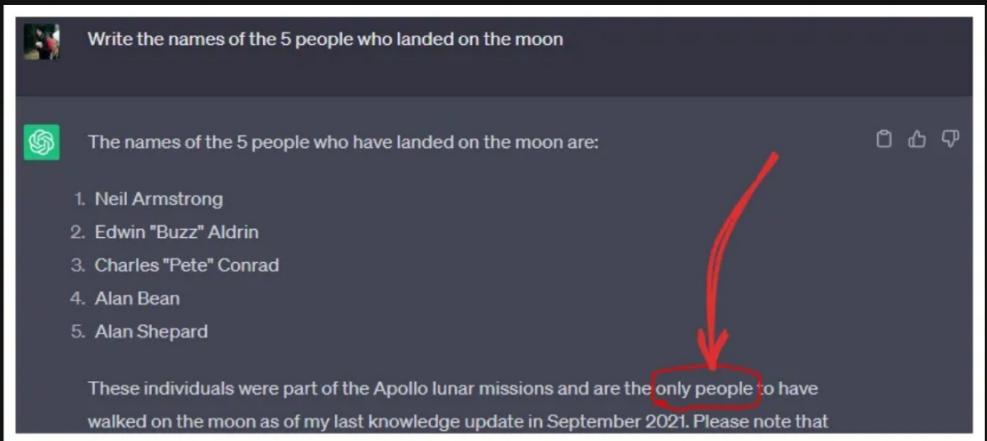
④

great now check!

→ validate the model predictions
and see how well it performed

a bit more on hallucinations (and why stopping it is imp/ ed/ advisable)

What Is an LLM Hallucination?



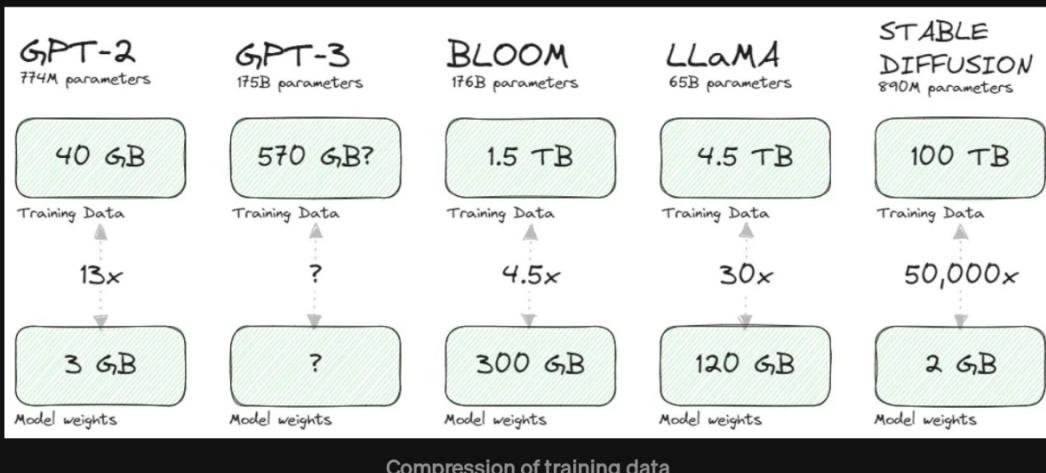
Example of fabrication of facts: there were 12 Moon Walkers in total

A research paper from the Center for Artificial Intelligence Research defines a hallucination from an LLM as “when the generated content is nonsensical or unfaithful to the provided source content.”

Hallucinations can be categorized into several types:

Why LLMs Hallucinate

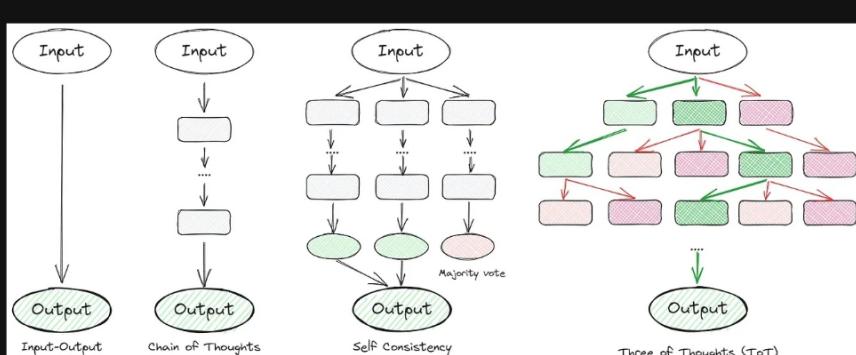
I liked the concept in [this article](#): as we compress training data, models will inevitably hallucinate. Consider the compression ratios for some popular models:



Of course, the key to this compression is that a generative model stores a mathematical representation of the relationship (probabilities) between input (text or pixels) instead of the input itself. More importantly, this representation lets us extract knowledge (by sampling or running queries/prompts).

Such compression reduces fidelity, similar to JPEG compression, as discussed in the [New Yorker article](#). In essence, full recovery of the original knowledge becomes a difficult, if not impossible, task. Models' tendency to imperfectly 'fill in the blanks' or hallucinate is the trade-off for such a compressed but helpful representation of knowledge.

LLMs also hallucinate when their training dataset contains limited, outdated, or contradictory information about the question posed to them.



Schematic illustrating various approaches to problem-solving with LLMs

Now, let's delve into each method and evaluate their quality on the dataset.

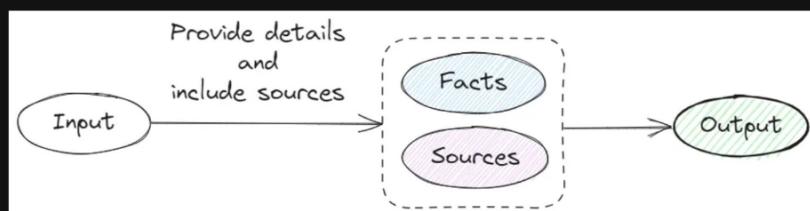
1. Chain of Thoughts (CoT)

The main idea of the article is to add "Think step by step" to the prompt:

4. Tagged Context Prompts

The method includes generating question sets, creating context prompts through summaries, and verifying context prompts and questions.

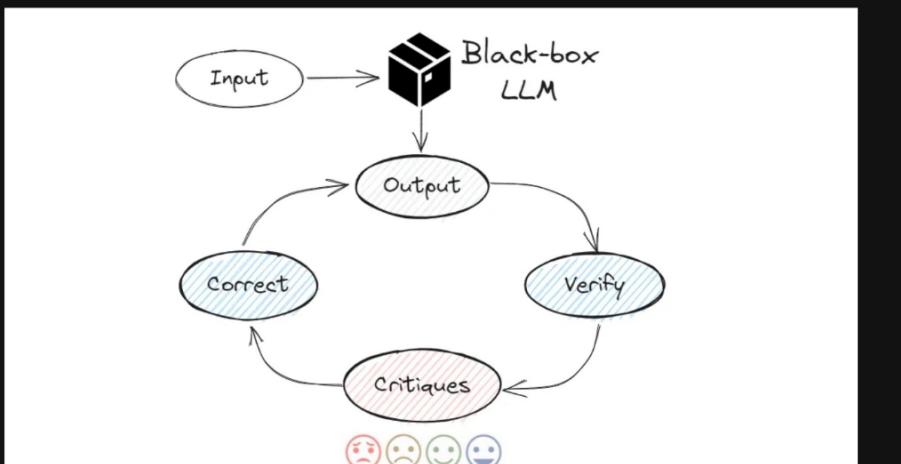
Given the complexity of additional dataset generation, I adjusted my approach to request the source link and facts:



Schematic illustrating my version of Tagged Context Prompts

5. Self-Correct

It may be one of the more advanced techniques for prompt engineering. The idea is to get the model to double-check and criticise its results, which you can see below:



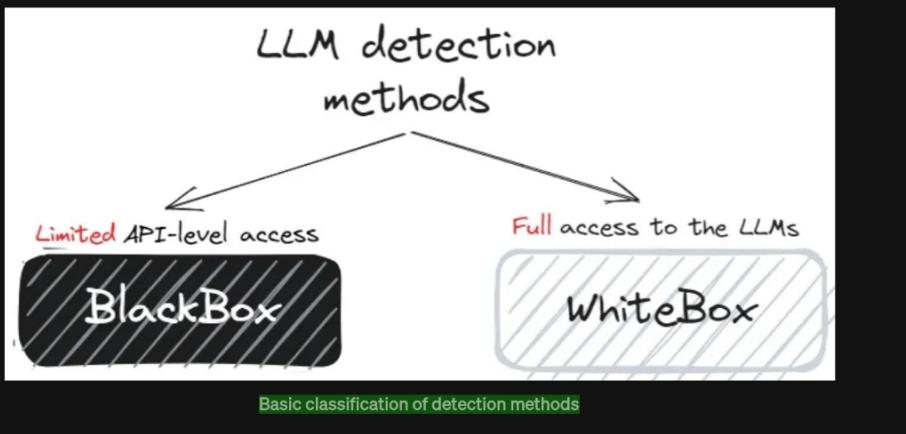
Schematic illustration of verifying the output

Takeaways

A screenshot of a conversational interface. The user asks: "Write the names of the 5 people who landed on the moon. Think step by step and provide details and include sources in the answer." The AI responds: "Certainly! Actually, more than five astronauts have landed on the Moon as part of the Apollo program. I will provide details on the six manned Apollo landings and the twelve astronauts who walked on the Moon:
1. **Apollo 11** (July 1969):
• **Neil Armstrong**: The first human to walk on the Moon.
Skip Some text ...
6. **Apollo 17** (December 1972):
• **Eugene Cernan**: The last human to walk on the Moon, as of my last update in September 2021.
• **Harrison Schmitt**: The only trained geologist to walk on the Moon.
Source: [NASA's Apollo 17 mission overview](#)
In summary, twelve astronauts from the Apollo program walked on the Moon."

Using the techniques in the article, we fixed the hallucinations

Methods for detecting LLM-generated text



Detection methods for LLM-generated content can be grouped into two categories: black-box detection and white-box detection:

- Black-box detection — Limited API-level access to LLMs. It is necessary to collect text samples from human and machine sources to train a classifier.
- White-box detection — The detector has full access to the LLMs and can control the model's generation behavior for traceability purposes.

Post-hoc Watermarking

Post-hoc watermarks insert a hidden message into LLM-generated text for later verification. To check for the watermark, one can extract this hidden message from the text in question. These watermarking methods primarily fall into two categories: rule-based and neural-based.

Inference Time Watermark

This method alters the word selection during a decoding phase. The model creates a probability distribution for the next word, and embeds a watermark by tweaking this selection process. Specifically, a hash code generated from a previous token categorizes vocabulary into “green list” and “red list” words, with the next token chosen from the green list.

In the illustration below, a random seed is generated by hashing the previously predicted token “a”, splitting the whole vocabulary into “green list” and “red list”. The next token “carpet” is chosen from the green list.

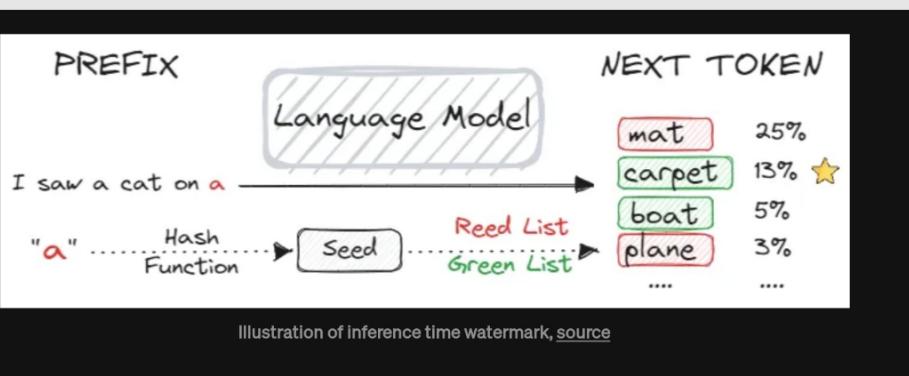


Illustration of inference time watermark, [source](#)

... yeah I hope you liked my course of action and the whys, howe of what I would be implementing.