BZAN 583: Generative AI with Large Language Models for Business Applications

Introduction

Michel Ballings

Outline I

What is Generative AI?

- model = algorithm + data
- AI = very smart model
- Generative (in the context of text):
  - used in the common sense that the model's goal is to generate data that is of a similar format as the inputs (i.e., we input text, and the model outputs text), also called sequence-to-sequence when dealing with text (text is a sequence of words), as opposed to
  - 'discriminative' where the goal is to classify data into a class (e.g., we input text and the model outputs a class 'good' or 'bad'), or 'summative' where the goal is to summarize the inputs into a numerical outcome (e.g., we input text and the model outputs a sentiment score), also called sequence-to-value.

Outline I

What is a Large Language Model?

- Language models refer to models that model text, specifically generative models
- The definition of large is a moving target
  - GPT-1 has 117 milllion parameters
  - GPT-2 has 1.5 billion parameters
  - GPT-3 has 175 billion parameters
- Is 'large' a good label? What if a new model has the same capabilities as GPT-3 but is 20 times smaller. Would we still call it 'large'?

Outline I

Scaling laws and compute-optimal models

- How large do models need to be?
- The answer depends on the available dataset size (measured by number of tokens) and the compute budget (hardware, project timeline, budget).
- Researchers at Deepmind found that there is what is called a compute-optimal model: a model with an optimal number of tokens and parameters given a compute budget. They called this compute-optimal model Chinchilla. Hence the name Chinchilla scaling laws. (`https://arxiv.org/abs/2203.15556`)
- The optimal training dataset size (measured in number of tokens) is 20 times the number of parameters.
- The optimal number of parameters is 5% of the training dataset size (measured in number of tokens)
- Before that paper was published, many LLMs had too many parameters for the dataset size they were trained on

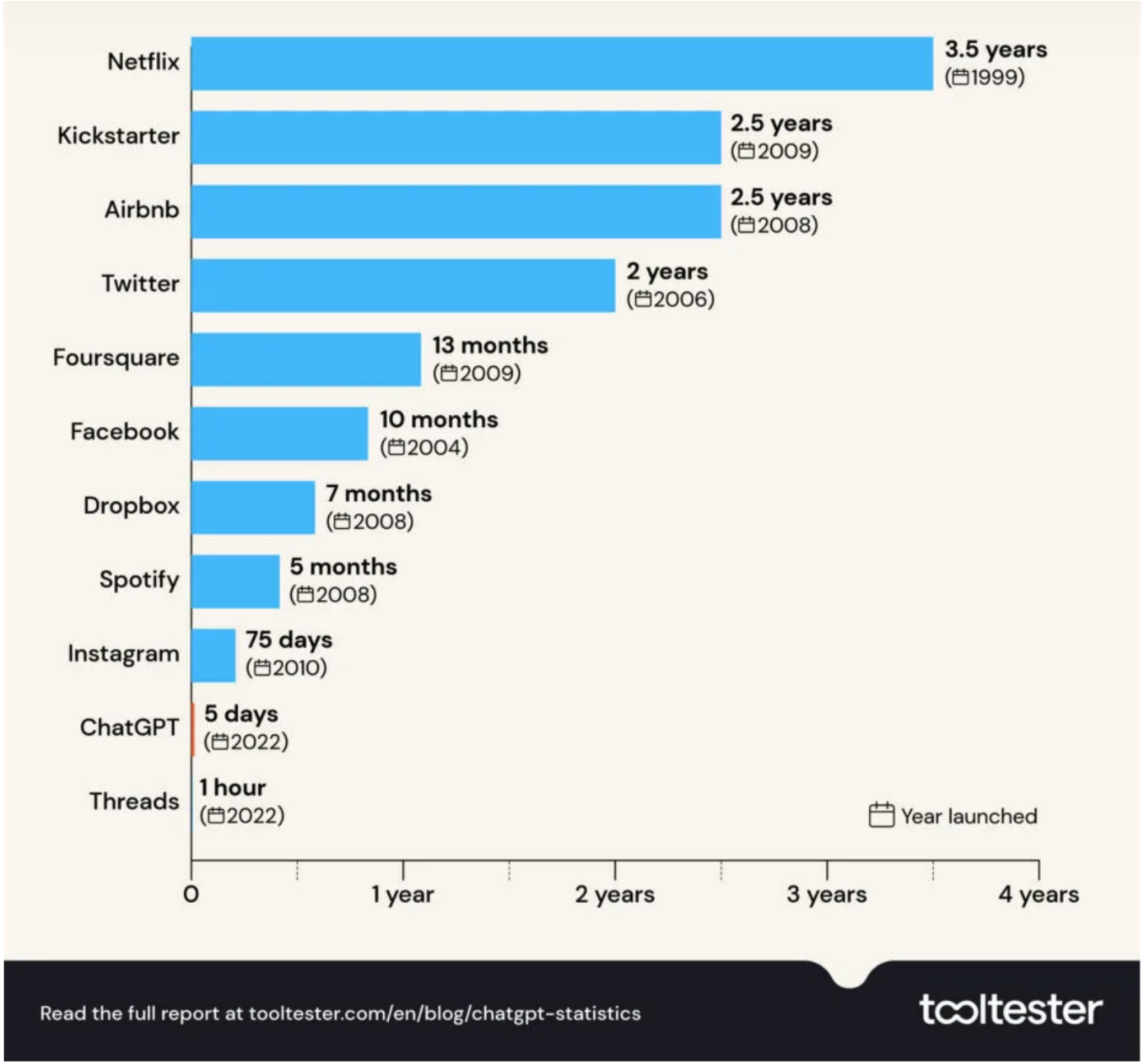
Outline I

Release dates of LLMs

https://en.wikipedia.org/wiki/List_of_large_language_models

Outline I

Time for ChatGPT to reach one million users



Outline I

Investments in AI

- Since OpenAI released ChatGPT on November 30, 2022, big tech has invested massive amounts of money in data centers. It is hard to keep track of how much exactly they invested but we can get an idea by looking at the stock price of NVIDIA, the largest beneficiary of the AI boom.
- In January 2025, NVIDIA became the world's most valuable publicly traded company.
- If you would have invested $100,000 in NVDA on December 2, 2022, your investment would now be worth $871,268.
- NVIDIA has a market capitalization of $4.5 trillion. For NVIDIA's stock to reach this level, the company had to generate unprecedented growth in sales.

Outline I

Use cases and tasks

LLMs are general purpose text tools. Examples of tasks include

- text summarization (e.g., summarize a legal document, summarize a financial report),
- generating text or source code,
- information extraction (e.g., extract numbers from a document),
- classification (e.g., assign a category to a piece of content),
- maintaining a conversation (e.g., self-service customer support),
- detecting toxic content,
- rewriting text (e.g., rewrite in a different tone or for a different audience),
- translation,
- question answering,
- masking personally identifiable information, and
- reasoning (e.g., thinking through the steps of a solution to a problem).

Outline I

Foundation models

A foundation model is a large, general-purpose AI model that is trained once on massive, diverse data and then reused as a base ("foundation") for many different tasks through prompting, fine-tuning, or tool-integration. It is pre-finetuning.

Table: Comparison between traditional task-specific machine learning models and foundation models.

Dimension	**Traditional ML Models**	**Foundation Models**
Training objective	Optimized for a single task or narrowly defined prediction problem.	Pretrained on massive, diverse data to learn general representations and transferable capabilities.
Data scale	Small to medium labeled datasets curated for a specific application.	Internet-scale, multi-domain, largely self-supervised or weakly supervised data.
Model scope	Narrow and specialized.	Broad, general-purpose, multi-task capable.

Foundation models

Table: Comparison between traditional task-specific machine learning models and foundation models.

Dimension	**Traditional ML Models**	**Foundation Models**
Reuse and transfer	Limited transfer; retraining required for new tasks.	Strong transfer; adapted via prompting, fine-tuning, adapters, or retrieval.
Training cost	Relatively low and affordable for individual teams.	Extremely high; requires large-scale compute, infrastructure, and capital investment.
Deployment pattern	One model per application.	One foundation model supports many downstream applications.
Value creation	Value concentrated in individual applications and feature engineering.	Value concentrates in the pretrained model, data pipelines, and compute infrastructure.

Foundation models

Table: Comparison between traditional task-specific machine learning models and foundation models.

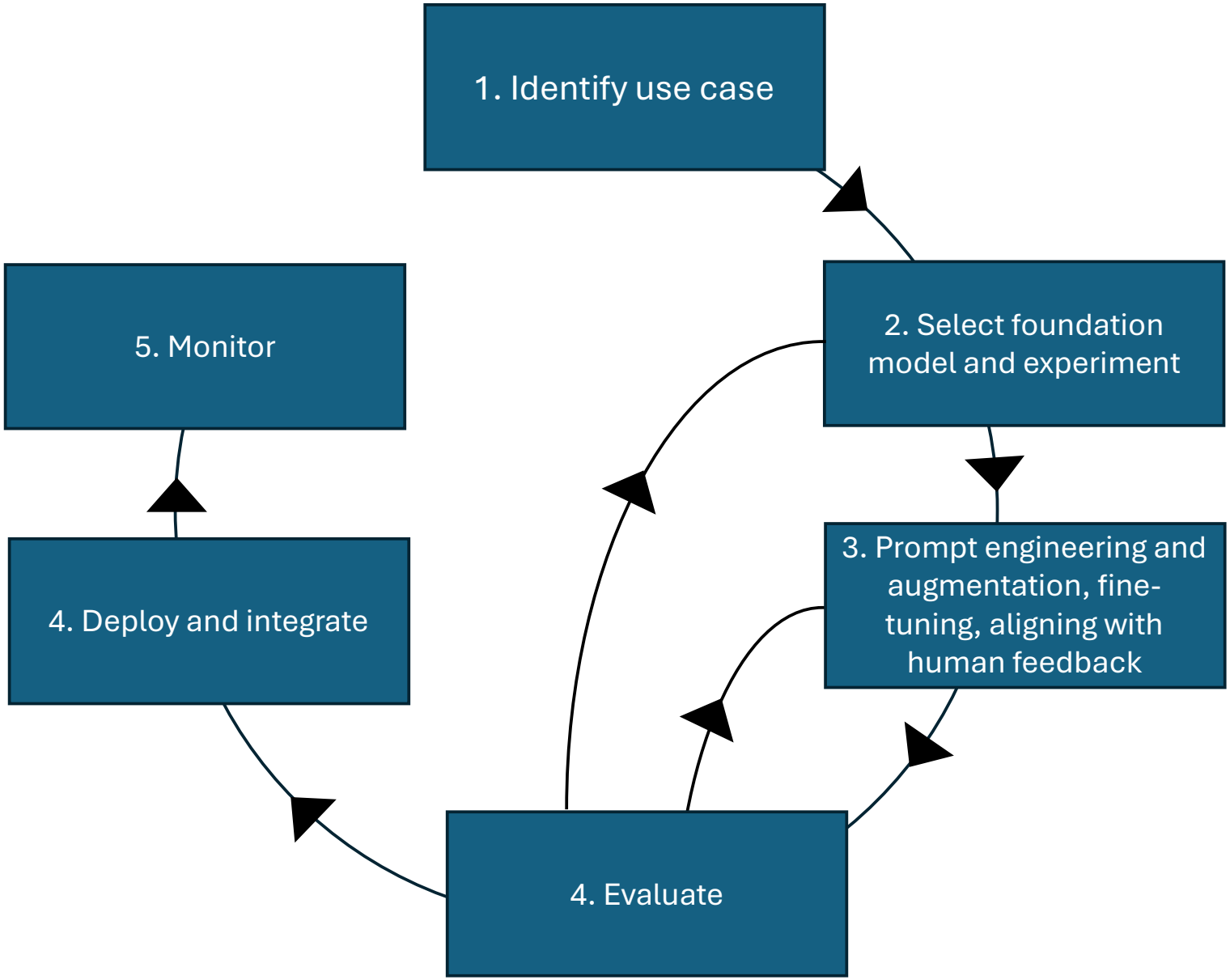
Dimension	**Traditional ML Models**	**Foundation Models**
Economic structure	Fragmented innovation across many small models.	Centralized scale economies and platform dynamics.
Update cycle	Frequent retraining as data or objectives change.	Slow, capital-intensive pretraining cycles with rapid downstream iteration.
Failure modes	Overfitting, dataset shift, brittle generalization.	Hallucination, alignment risk, emergent behavior, governance challenges.

Model hubs

- Because of the high cost of training these models, we will start our generative AI projects with an existing foundation model.
- Foundation models can be obtained from model hubs. The most extensive model hubs are the
  - Hugging Face Model Hub
    `https://huggingface.co/docs/hub/en/models-the-hub`
  - PyTorch Hub
    `https://pytorch.org/hub/`
  - and the Amazon SageMaker JumpStart
    `https://aws.amazon.com/sagemaker-ai/jumpstart/`
- We will use the Hugging Face Model Hub. Hugging face, Inc `https://en.wikipedia.org/wiki/Hugging_Face` has developed an excellent Python module called transformers. We will use this module in this class.

Outline I

Generative AI project lifecycle



Outline I

Text generation loop; LLMs output one word at a time

LLMs are 'autoregressive' meaning they generate text by predicting the next word in a sequence based on the previous words it has already generated, essentially building upon the context created by the preceding words to produce coherent text one word at a time; this is like how a human might complete a sentence by considering the words already spoken. The algorithm below shows the text generation loop; LLMs output one word at a time.

max_length = 20
input = 'How can I make chocolate chip cookies?'
for *t = 1:max_length* **do**
    output_single_word = LLM(input)
    input.append(output_single_word)
end

Outline I

Our first completions

See *generating_first_text.py*.

Outline I

Responsible LLM development and bias

- LLMs can produce open-ended content that varies with repeated invocations, even if the prompt is identical.
- To learn how to do that, the LLM needs to train on open-ended content.
- This is different from traditional applications of machine learning (ML) where inputs and outputs are structured, and where outputs do not vary if inputs are constant.
- Because they are so open ended, it is hard to control for bias in LLMs. Let's first look at how to solve for bias in traditional ML and then have a look at how to solve it in LLMs.
- Example of bias in traditional ML. Consider a prediction model to assist in approving or denying credit card applications. We would want the model to yield fair predictions. For example, if two people have the exact same credit history and profile except for gender, the model should return the exact same approval probability. This is straightforward to accomplish with structured data and models. We could simply remove gender from the inputs, calibrate the model for gender, and audit the model to ensure that the error rate of the model is the same for male and female applicants.

Responsible LLM development and bias

- Can we apply a similar approach with generative AI? For example, consider the following prompt: 'Professor Hewett started teaching class and.' To ensure that the model does not sustain stereotypes, we may want to enforce that the completions of the model assign female and male pronouns equally likely. Of course, we should then probably also do this for other jobs, such as garbage(wo)man, police officer, gynecologist, nurse, and landscaper. This can quickly become intractable. The model also needs to make sure that its answers are consistent with the context of the prompt. What if the prompt mentions long hair? Should this result in completions that are more likely to use female pronouns?
- There are three places where we can intervene to mitigate bias. First, we could filter out biased data from the training data. If the training data are not biased, the model will not produce biased content. Second, we could filter or block user prompts that generate biased content before sending the prompt to the model. Third, we could filter and block biased generated content before sending it to the user.

Responsible LLM development and bias

- Unfortunately, filtering turns out to be very hard, except when bias is obvious. Filtering will most certainly have to involve guardrail models. Such models predict the level of bias in content and require human-annotated data. For example, humans would have to create a table of data with two columns. One column would contain open-ended text and the other column would denote whether - yes or no - the text is biased. Creating the training data will be a daunting task, because humans would have to read the text and decide whether it is biased. Building consensus on the definition of bias will be even harder.
- So what can we do? Researchers have focused on the capability to tell the difference between machine-generated and human-written text to reduce potential harms caused by LLMs, specifically through watermarking. Known harms are models being used at-scale for malicious purposes such as social media bots, fake product reviews, and automated text generation on Wikipedia (`https://arxiv.org/pdf/2306.04634`).

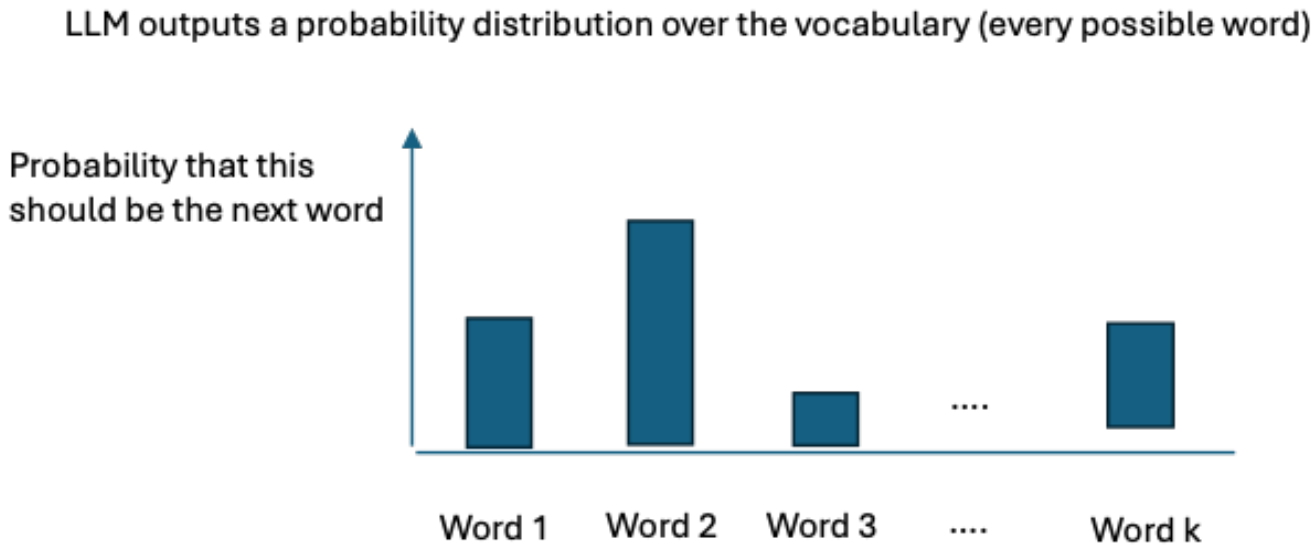
Watermarking

https://arxiv.org/pdf/2301.10226

Prompt	Num tokens	Z-score	p-value
…The watermark detection algorithm can be made public, enabling third parties (e.g., social media platforms) to run it themselves, or it can be kept private and run behind an API. We seek a watermark with the following properties:			
No watermark Extremely efficient on average term lengths and word frequencies on synthetic, microamount text (as little as 25 words) Very small and low-resource key/hash (e.g., 140 bits per key is sufficient for 99.999999999% of the Synthetic Internet	56	.31	.38
With watermark - minimal marginal probability for a detection attempt. - Good speech frequency and energy rate reduction. - messages indiscernible to humans. - easy for humans to verify.	36	7.4	6e-14

Figure 1. Outputs of a language model, both with and without the application of a watermark. The watermarked text, if written by a human, is expected to contain 9 "green" tokens, yet it contains 28. The probability of this happening by random chance is $\approx 6 \times 10^{-14}$, leaving us *extremely* certain that this text is machine generated. Words are marked with their respective colors. The model is OPT-6.7B using multinomial sampling. Watermark parameters are $\gamma, \delta = (0.25, 2)$. The prompt is the whole paragraph marked in blue below.

Watermarking



Watermarking

https://arxiv.org/pdf/2301.10226

2. A simple proof of concept

We start out by describing a simple "hard" red list watermark in Algorithm 1 that is easy to analyze, easy to detect and hard to remove. The simplicity of this approach comes at the cost of poor generation quality on low entropy sequences. We will discuss more sophisticated strategies later.

Algorithm 1 Text Generation with Hard Red List

Input: prompt, $s^{(-N_p)} \dots s^{(-1)}$
for $t = 0, 1, \cdots$ **do**

1. Apply the language model to prior tokens $s^{(-N_p)} \dots s^{(t-1)}$ to get a probability vector $p^{(t)}$ over the vocabulary.
2. Compute a hash of token $s^{(t-1)}$, and use it to seed a random number generator.
3. Using this seed, randomly partition the vocabulary into a "green list" G and a "red list" R of equal size.
4. Sample $s^{(t)}$ from G , never generating any token in the red list.

end for

The method works by generating a pseudo-random red list of tokens that are barred from appearing as $s^{(t)}$. The red list generator is seeded with the prior token $s^{(t-1)}$, enabling the red list to be reproduced later without access to the entire generated sequence.

Watermarking

- 'A third party with knowledge of the hash function and random number generator can re-produce the red list for each token and count how many times the red list rule is violated. We can detect the watermark by testing the following null hypothesis: The text sequence is generated with no knowledge of the red list rule. Because the red list is chosen at random, a natural writer is expected to violate the red list rule with half of their tokens, while the watermarked model produces no violations.'
  `https://arxiv.org/pdf/2301.10226`
- A better performing version of this algorithm is, when generating, the "green" words will have a small 'bias' value added to their logits, thus having a higher chance to be generated. The watermarked text can be detected by calculating the proportion of "green" tokens in the text and estimating how likely it is statistically to obtain that amount of "green" tokens for human-generated text.

Watermarking Applications

- **Organizational Knowledge Provenance.** Watermarking enables firms running on-prem or private LLMs to trace which documents, code, analyses, and decisions originate from the model, allowing measurement of cognitive automation, dependency on AI, knowledge diffusion, and downstream risk propagation.
- **Platform Usage and Economic Attribution.** LLM providers can embed watermarks to measure how much generated content actually propagates into real-world artifacts (documents, repositories, products), enabling true usage measurement beyond API calls, value attribution, pricing optimization, and ecosystem analytics.
- **Content Integrity and Auditability.** Watermarking provides machine-verifiable provenance of AI-generated content for regulated industries, media, and public institutions, supporting compliance, accountability, misinformation detection, and long-term trust in digital information flows.