

# **Benchmarking Large Language Models and Privacy Protection**

**Office of the Privacy Commissioner of Canada  
Contributions Program 2024-25**

**Final Report - Version 23 April, 2025**

**Professor Rafal Kulik, University of Ottawa**

Department of Mathematics and Statistics

150 Louis Pasteur Priv., K1N 6N5

STM564

(613) 562 5800 ext. 6266



uOttawa

## Table of contents

|                                                                                                                             |           |
|-----------------------------------------------------------------------------------------------------------------------------|-----------|
| <b>APPLICANT AND TEAM .....</b>                                                                                             | <b>2</b>  |
| <b>PROJECT TITLE .....</b>                                                                                                  | <b>4</b>  |
| <b>BASIC INFORMATION.....</b>                                                                                               | <b>4</b>  |
| <b>PROJECT SUMMARY .....</b>                                                                                                | <b>5</b>  |
| <b>PROJECT DESCRIPTION .....</b>                                                                                            | <b>6</b>  |
| <b>FINAL REPORT – STRUCTURE.....</b>                                                                                        | <b>7</b>  |
| <b>PIPEDA CONTEXT .....</b>                                                                                                 | <b>8</b>  |
| <b>PART 1: TERMINOLOGY AND PROBLEMS RELATED TO DATA PRIVACY IN LLM .....</b>                                                | <b>9</b>  |
| VOCABULARY AND DEFINITIONS.....                                                                                             | 9         |
| OVERVIEW .....                                                                                                              | 13        |
| BASIC TECHNIQUES AND PARAMETERS.....                                                                                        | 13        |
| SIMPLE EXAMPLES OF LARGE LANGUAGE MODELS.....                                                                               | 13        |
| COMMON PROBLEMS IN CHATGPT .....                                                                                            | 16        |
| EXAMPLES OF HALLUCINATIONS IN LLM/CHATGPT .....                                                                             | 17        |
| EXAMPLES OF CYBERATTACKS.....                                                                                               | 20        |
| DISCREPANCY BETWEEN FREE AND PAID VERSIONS OF CHATGPT .....                                                                 | 22        |
| APPENDIX FOR PART 1 – CHATGPT EXPERIMENTS.....                                                                              | 23        |
| <b>PART 2: TECHNICAL DESCRIPTION OF LARGE LANGUAGE MODELS THROUGH DEEP LEARNING AND NEURAL NETWORK ALGORITHMS .....</b>     | <b>25</b> |
| <b>PART 3: TECHNICAL DESCRIPTION ON DATA PRIVACY ISSUES IN LARGE LANGUAGE MODELS.....</b>                                   | <b>26</b> |
| INTRODUCTION .....                                                                                                          | 26        |
| <i>Privacy Risks .....</i>                                                                                                  | <i>26</i> |
| <i>Data Privacy mitigation strategies .....</i>                                                                             | <i>26</i> |
| <i>New privacy-preserving methods in LLM.....</i>                                                                           | <i>27</i> |
| PRIVACY RISKS .....                                                                                                         | 27        |
| <i>Data leakage from fine-tuning .....</i>                                                                                  | <i>27</i> |
| FINE-TUNING APPROACHES .....                                                                                                | 27        |
| <i>Data memorization .....</i>                                                                                              | <i>29</i> |
| <i>Examples of Memorization .....</i>                                                                                       | <i>29</i> |
| WHY IS MEMORIZATION PROBLEMATIC? .....                                                                                      | 30        |
| METHODS FOR DETECTING MEMORIZATION .....                                                                                    | 30        |
| CHALLENGES IN IDENTIFYING MEMORIZED CONTENT .....                                                                           | 31        |
| <i>Adversarial vulnerabilities .....</i>                                                                                    | <i>31</i> |
| DATA PRIVACY MITIGATION STRATEGIES.....                                                                                     | 34        |
| <i>Privacy techniques .....</i>                                                                                             | <i>35</i> |
| PRIVACY TECHNIQUES – STEPS IN LLM.....                                                                                      | 36        |
| <b>PART 4: EXPERIMENTS ON PRIVACY ENHANCING TECHNIQUES IN LARGE LANGUAGE MODELS. RESULTS AND ANALYSIS.....</b>              | <b>38</b> |
| EXPERIMENT 1: ONLINE AND OFFLINE STOCHASTIC GRADIENT DESCENT: ANALYSIS OF CONVERGENCE ...                                   | 39        |
| EXPERIMENT 2: ONLINE AND OFFLINE STOCHASTIC GRADIENT DESCENT: ANALYSIS OF THE EMPIRICAL DISTRIBUTION OF THE ESTIMATOR ..... | 41        |
| EXPERIMENT 3: DIFFERENTIALLY PRIVATE ONLINE STOCHASTIC GRADIENT DESCENT: ANALYSIS OF CONVERGENCE .....                      | 44        |

|                                                                                                                                                              |           |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|
| EXPERIMENT 4: DIFFERENTIALLY PRIVATE ONLINE STOCHASTIC GRADIENT DESCENT: ANALYSIS OF THE EMPIRICAL DISTRIBUTION OF THE ESTIMATOR.....                        | 47        |
| EXPERIMENT 5: DIFFERENTIALLY PRIVATE ONLINE STOCHASTIC GRADIENT DESCENT VS. GAUSSIAN MECHANISM: ANALYSIS OF THE EMPIRICAL DISTRIBUTION OF THE ESTIMATOR..... | 50        |
| EXPERIMENT 6: NEURAL NETWORK FOR LINEAR REGRESSION MODEL.....                                                                                                | 53        |
| EXPERIMENT 7: NEURAL NETWORK FOR LINEAR REGRESSION MODEL – IMPACT OF NOISE ADDITION.....                                                                     | 56        |
| EXPERIMENT 8: BUILDING A SIMPLE TRANSFORMER-BASED LANGUAGE MODEL .....                                                                                       | 59        |
| EXPERIMENT 9: SIMPLE EXAMPLE OF FINDING THE MOST COMMON WORD FROM CERTAIN CATEGORY ASSOCIATED WITH ANY GIVEN WORD .....                                      | 63        |
| EXPERIMENT 10: SIMPLE CLASSIFICATION PROBLEM .....                                                                                                           | 65        |
| EXPERIMENT 11: WORD PREDICTION BASED ON BIGRAM MODEL .....                                                                                                   | 67        |
| <b>PART 5: SCOPE OF POLICIES – ANALYSIS OF LEGAL ASPECTS RELATED TO LLM, IN THE CONTEXT OF THE EXPERIMENTS. ....</b>                                         | <b>70</b> |
| <b>PART 6: SUMMARY OF ACTIVITIES AND RELATED RESOURCES.....</b>                                                                                              | <b>76</b> |
| <b>PART 7: TRANSLATION OF SELECTED PARTS OF THE DOCUMENT IN FRENCH. ....</b>                                                                                 | <b>77</b> |

## Applicant and Team

**Principal investigator:** Rafal Kulik, PhD, is a Professor of Mathematics and Statistics, University of Ottawa. In the last four years, he has worked in the area of differential privacy and disclosure risk metrics. In this area, he has partnered with Privacy Analytics through two Mitacs grants: [Privacy Guarantees and Risk Identification: Statistical Framework and Methodology \(link\)](#), and [Statistical Framework and Methodology for Risk and Privacy in Complex and High-Dimensional Data \(link\)](#). This research collaboration thus far includes two PhD students and one master's student. Professor Kulik also specializes in theoretical statistics, extreme value theory and time series. He published two books (*Long Memory Processes*, *Heavy Tailed Time Series*). He is an author of 50 publications in top journals in mathematics and statistics. He serves as an Associate Editor in top four journals in the field. He has supervised three post-doctoral fellows, six doctoral students and numerous masters students.

**Co-investigator:** Teresa Scassa, PhD, is the Canada Research Chair in Information Law and Policy at the University of Ottawa, Faculty of Law. She is the author or co-author of several books, including *Canadian Trademark Law* (2d edition, LexisNexis 2015), and *Electronic Commerce and Internet Law in Canada*, (CCH Canadian Ltd. 2012) (winner of the 2013 Walter Owen Book Prize). She is a past member of the External Advisory Committee of the Office of the Privacy Commissioner of Canada, and of the Canadian Government Advisory Committee on Open Government. She is a member of the GEOTHINK research partnership and has written widely in the areas of intellectual property law, law and technology, and privacy.

**Industry support:** Privacy Analytics provided advisory support and coordinate access to relevant data and computing infrastructure for students to run experiments. Privacy Analytics will provide in-kind support only, to facilitate research.

Luk Arbuckle is Chief Methodologist and Privacy Officer at Privacy Analytics. He is a recognized author on the subject of anonymization, and an industry supervisor for Mitacs students in collaboration with Professor Kulik.

**Students:** Three science students (Heidi Barriault, Bartosz Glowacki, Chang Qu) are funded to conduct literature reviews, gap analyses, run experiments, and summarize findings. One law student (Yuma Wu) is funded to conduct literature reviews, gap analyses, construct surveys and facilitate outreach to experts and civil society groups and assist in finalizing guidance and recommendations.

## Project Title

Benchmarking Large Language Models and Privacy Protection

## Basic Information

Organization name: Department of Mathematics and Statistics, University of Ottawa

Address:

150 Louis-Pasteur Pvt, STEM Complex, room 336

Ottawa, ON K1N 6N5

Billing address: <same>

Fax number: N/A

Principal Investigator: Rafal Kulik

Email address: [rkulik@uottawa.ca](mailto:rkulik@uottawa.ca)

## Project Summary

In the current digital age, the accelerated growth of data generated by individuals has fueled advances in artificial intelligence (AI), particularly the development and use of large language models (LLMs). These models, which mimic human-like understanding and generation of language, are being integrated into various societal tools. The evolution of LLM deployment requires an appropriate strategy for ensuring user privacy. Addressing privacy challenges requires going beyond traditional privacy means and carefully balancing the models' performance needs and the constraint of privacy of sensitive data. The challenge lies in the dual need to harness the potential of LLMs to drive innovation and improve services, while ensuring the confidentiality and privacy of the data used to train these models.

The growing capabilities and risks of LLMs underscores the need for specific privacy standards and the refinement of measures like differential privacy tailored to these technologies. This development would improve privacy leakage and create a benchmark for their future impact. Navigating the integration of these privacy safeguards presents several challenges:

- 1) Implementing privacy in LLMs is complex and must maintain their function.
- 2) Privacy measures in LLM infrastructures pose practical, scalable integration challenges.
- 3) A notable gap exists in methodologies for establishing baselines that can accurately reflect the privacy and confidentiality guarantees of LLMs.
- 4) Navigating the evolving legal landscape presents a challenge, as the rapid development of LLM technologies often outpace existing legal frameworks.
- 5) There is a significant knowledge gap among the public and stakeholders regarding the implications of privacy risks in LLM systems.

In response to these challenges, this project is structured in five phases:

**Phase I – Establishing Privacy Baselines for LLMs:** Explaining basic techniques and applications of LLMs as well as privacy mechanisms and their relevance to LLMs. We will provide an analysis of the unique privacy challenges these models present.

**Phase II – Privacy Mechanism Experimentation and Evaluation:** Experiments that implement the privacy-preserving techniques within LLMs. This includes testing mechanisms across different datasets and measure privacy-utility trade-offs.

**Phase III – Methodology Development for Baseline Establishment:** Using the outputs from phase I and II, we will focus on establishing a baseline for privacy guarantees in LLMs.

**Phase IV – Scope of Policies:** A thorough analysis of existing legal and policy frameworks governing LLMs, identifying gaps and forecasting future legal needs.

**Phase V – Education and Awareness:** The final phase is to raise awareness and educate both the public and stakeholders about the importance of privacy in LLMs and the efforts being made to protect it. This could include developing a course, workshops, and engaging in speaking opportunities.

# Project Description

## Overview

In the current digital age, the accelerated growth of data generated by individuals has fueled advances in artificial intelligence (AI), particularly the development and deployment of large language models (LLMs). These sophisticated AI systems, capable of understanding, generating, and interacting with human language in ways that mimic human thought processes, are becoming integral to applications ranging from personalized content creation to drug discovery. As these models become more deeply embedded in the daily functions of society, the need to protect individual privacy within these systems is crucial.

LLMs, which are a special case of Natural Language Processing (NLP), are very large deep learning models that are pre-trained on large amounts of data. The underlying structure, known as a transformer, is a set of neural networks consisting of an encoder and decoder with self-attention capabilities. In essence, they use a method that helps them pay attention to the entire content of text while being "taught" with massive amounts of information. They use neural networks to accurately transmit and interpret the data to either learn or respond to human speech.

The ability of LLMs to process and learn from large and diverse data sets raises significant privacy concerns, particularly the risk of unintentionally revealing sensitive information. The challenge lies in the dual need to harness the potential of LLMs to drive innovation and improve services, while ensuring the confidentiality and privacy of the data used to train these models. This balancing act requires a nuanced approach to privacy that goes beyond traditional data anonymization and de-identification techniques.

Given the unique capabilities and risks associated with LLMs, there is a growing need to establish robust privacy standards specifically tailored to these technologies. This includes the development and refinement of privacy-preserving mechanisms, such as differential privacy, in the context of LLMs. These initiatives aim to enhance the privacy protections of LLMs and provide a baseline for assessing their privacy impact as these models evolve. The rapid development of LLMs and the pace at which these tools are evolving present a significant challenge in defining current and practical guidelines that can effectively address the use and deployment of these systems.

Traditional approaches such as data anonymization and de-identification may prove themselves insufficient against the unpredictable capabilities of LLMs. LLMs trained on large amounts of data may contain unforeseen and subtle patterns unique to individuals, with risks of model inversion and membership inference. This is a fundamental shift between the capabilities of LLMs and the static nature of traditional

privacy protections. In addition, the dynamic nature of the structure of LLMs may pose challenges to the application of standard privacy-preserving techniques such as data minimization or access control. Investigating these risks and addressing limitations requires a re-evaluation of privacy protection techniques in the context of LLMs that are suited to the dynamic nature of LLMs.

## Final Report – structure

Part 1: Terminology and problems related to data privacy in LLM (non-mathematical).

Part 2: Technical description of Large Language Models through Deep Learning and Neural Network algorithms.

Part 3: Technical description on data privacy issues in Large Language Models.

Part 4: Experiments on privacy enhancing techniques in Large Language Models. Results and analysis.

Part 5: Scope of policies – analysis of legal aspects related to LLM, in the context of the experiments.

Part 6: Summary of activities and related resources.

Part 7: Translation of selected parts of the document in French.

Appendices :

- Neural-networks-notes.pdf – technical description of simple neural networks.
- Zip folder with all the codes for the experiments conducted in Part 4.

## PIPEDA context

An important aspect of this report is its potential relation and implication on PIPEDA. Quoting from the Office of the Privacy Commissioner webpage:

*The Personal Information Protection and Electronic Documents Act (PIPEDA) sets the ground rules for how private-sector organizations collect, use, and disclose personal information in the course of for-profit, commercial activities across Canada. PIPEDA also applies to the personal information of employees of federally-regulated businesses.*

There are 10 key PIPEDA principles

- Accountability
- Identifying purposes
- Consent
- **Limiting collection**
- **Limiting use, disclosure, and retention**
- **Accuracy**
- **Safeguards**
- **Openness**
- **Individual access**
- Challenging compliance

We emphasized the principles that are covered (at different level) at this report.

We think that this report can help addressing two issues related to PIPEDA:

- 1) For companies and organisations that create new Large Language Models, it provides a technical framework on the best practices to assure the appropriate level of data privacy.
- 2) For companies and organisations that use Large Language Models, it identifies potential risks when using such models and the best practices to minimize the leakage of private information.

## Part 1: Terminology and problems related to data privacy in LLM

The goal of the first part is to establishing Privacy Baselines for LLMs: provide definitions, explain basic techniques and applications of LLMs as well as privacy mechanisms and their relevance to LLMs.

### Vocabulary and definitions

We start with basic vocabulary related to *Large Language Models (LLM)*.

**Context clues:** In our case, hints given to the *LLM* to help predict the next word.

**Database:**<sup>1</sup> A collection of stored data.

**Data subjects:**<sup>2</sup> Person to whom data relate.

**Deep-learning**<sup>3</sup>:

1. uses artificial *neural networks* to learn from data. Artificial neural networks are inspired by the human brain, and they can be used to solve a wide variety of problems, including image recognition, *natural language processing*, and speech recognition.
2. a subset of machine learning that uses multilayered neural networks, called deep neural networks, to simulate the complex decision-making power of the human brain

**Fine-tuning**<sup>4</sup>: adapting the pre-trained model to specific tasks. This process takes place by updating parameters on a new dataset.

**Hallucinations**<sup>5</sup>: confident responses that seemed faithful but are non-sensical when viewed considering the common knowledge in these areas.

**Large Language Model (LLM)**<sup>6</sup>: a deep-learning algorithm that uses massive amounts of parameters and training data to understand and predict text.

---

<sup>1</sup>Sparapani, T., & Sherman, J. (2021). *Privacy Tech's Third Generation: A review of the emerging privacy tech sector*. [https://fpf.org/wp-content/uploads/2021/06/FPF-PTA-Report\\_Digital.pdf](https://fpf.org/wp-content/uploads/2021/06/FPF-PTA-Report_Digital.pdf)

<sup>2</sup>Fanti, G., Pihur, V., & Erlingsson, Ú. (2015). *Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries*. arXiv.org. <http://arxiv.org/abs/1503.01214>

<sup>3</sup><https://cloud.google.com/discover/what-is-deep-learning>; <https://www.ibm.com/topics/deep-learning>

<sup>4</sup>Shanepeckham. (2024, January 30). *Getting started with LLM fine-tuning*. Microsoft Learn. <https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/working-with-llms/fine-tuning>

<sup>5</sup>Alkaissi, H., & McFarlane, S. I. (2023). Artificial Hallucinations in ChatGPT: Implications in Scientific Writing. *Cureus*, 15(2), e35179. <https://doi.org/10.7759/cureus.35179>

<sup>6</sup>McDonough, M. (2024, November 7). *Large language model | Definition, History, & Facts*. Encyclopedia Britannica. <https://www.britannica.com/topic/large-language-model>

**Natural Language Processing (NLP)**<sup>7</sup>: enables computers and digital devices to recognize, understand and generate text and speech.

**Neural network**<sup>8</sup>: method in artificial intelligence (AI) that teaches computers to process data in a way that is inspired by the human brain.

**N-grams**<sup>9</sup>: sequence of n words.

**Query**: A request for information that is calculated automatically from a dataset.

**Supervised learning**<sup>10</sup>: uses labeled datasets to train algorithms to predict outcomes and recognize patterns. Supervised learning uses labeled *training datasets* to try and teach a model a specific, pre-defined goal

**Temperature parameter**<sup>11</sup>: a parameter in *LLM*, which controls the randomness or creativity of the model's responses. Specifically, it adjusts the probability distribution over potential next words in the text generated by the model.

The different values of LLM temperature parameters:

1. **Low Temperature** (<1.0) – Setting the temperature to a value of less than 1 makes the model's output more deterministic and repetitive. Lower temperatures lead to the model picking the most likely next word more often, reducing the variability of the output. This can be useful when you need more predictable, conservative responses, but it might also result in less creative or diverse text, also making the model sound more robotic.
2. **High Temperature** (>1.0) – A temperature setting above 1 increases randomness in the generated text. The model is more likely to select less probable words as the next word in the sequence, leading to more varied and sometimes more creative outputs. However, this can also result in more errors or nonsensical responses, since the model is less constrained by the probability distribution of its training data.
3. **Default Temperature** of 1.0 – This is often the default setting, aiming for a balance between randomness and determinism. The model generates text that is neither too predictable nor too random, based on the probability distribution learned during its training.

<sup>7</sup>Stryker, C., & Holdsworth, J. (2024, August 11). What is NLP (Natural Language Processing)? *Explainers*. <https://www.ibm.com/topics/natural-language-processing>

<sup>8</sup>What is a Neural Network? - Artificial Neural Network Explained - AWS. (n.d.). Amazon Web Services, Inc. <https://aws.amazon.com/what-is/neural-network/#:~:text=A%20neural%20network%20is%20a,inspired%20by%20the%20human%20brain>

<sup>9</sup>Jurafsky, D., & Martin, J. H. (2024). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models* (3rd ed.). <https://web.stanford.edu/~jurafsky/slp3>

<sup>10</sup>What is Supervised Learning? | Google Cloud. (n.d.). Google Cloud. <https://cloud.google.com/discover/what-is-supervised-learning>

<sup>11</sup> <https://www.iguazio.com/glossary/llm-temperature/>

**Tokenization<sup>12</sup>:** the mathematical process of breaking down a piece of text, such as a sentence or document, into smaller units called *tokens*.

Examples of tokenization:

Assume we have a sentence "*Probability measures uncertainty*".

**1. Whitespace Tokenization**

- a. Splits text based on spaces.
- b. Example Output: ["Probability", "measures", "uncertainty;", "..."]

**2. Word Tokenization**

- a. Separates words from punctuation.
- b. Example Output: ["Probability", "measures", "uncertainty", ";", "..."]

**3. Sentence Tokenization**

- a. Uses punctuation cues to split the text into sentences.
- b. Example Output: ["Probability measures uncertainty;", "for example", "..."]

**4. Subword Tokenization**

- a. Breaks words into more frequent and common subwords or characters.
- b. Example Output: ["Probab", "ility", "measure", "s", "..."]

More tokenization methods are available, each method may offer advantages for specific *NLP* tasks.

**Tokens<sup>13</sup>:** small chunks of text generated by splitting the input text into smaller segments. These segments can either be words or groups of characters, varying in length from a single character to an entire word.

**Training dataset<sup>14</sup>:** huge amounts of text sources (from billions to billions of words) from a variety of sources, including public source.

**Unsupervised learning<sup>15</sup>:** type of machine learning that learns from data without human supervision. Unsupervised machine learning models are given unlabeled data and allowed to discover patterns and insights without any explicit guidance or instruction.

**Word vectorization<sup>16</sup>:** converting input data from its raw format (i.e. text) into vectors of real numbers which is the format that ML models support.

<sup>12</sup>LLM Glossary. (n.d.). orq.ai. <https://docs.orq.ai/docs/llm-glossary>

<sup>13</sup>Wang, W. W., Andrews, S., & Van Kraay, T. (2024, August 14). *LLM tokens*. Microsoft Learn. <https://learn.microsoft.com/en-us/azure/cosmos-db/gen-ai/tokens>

<sup>14</sup>Lareo, X. (n.d.). *Large language models (LLM)*. European Data Protection Supervisor. [https://www.edps.europa.eu/data-protection/technology-monitoring/techsonar/large-language-models-llm\\_en](https://www.edps.europa.eu/data-protection/technology-monitoring/techsonar/large-language-models-llm_en)

<sup>15</sup>*What is unsupervised learning?* | Google Cloud. (n.d.). Google Cloud. <https://cloud.google.com/discover/what-is-unsupervised-learning>

<sup>16</sup>Jha, A. (2023, August 11). *Vectorization Techniques in NLP [Guide]*. neptune.ai. <https://neptune.ai/blog/vectorization-techniques-in-nlp-guide>

### Example of vectorization: **One-Hot Encoding**

One-hot encoding maps each word to a vector in a high-dimensional space where the size of the vector corresponds to the size of the vocabulary. A vocabulary is a predefined collection of words; a larger vocabulary allows greater flexibility in handling different words but also increases complexity. Each vector consists mostly of zeros, with a '1' in the position corresponding to the word in the vocabulary.

For example:

- Assume we have a vocabulary like this: ["P", "(", "event", "A", ")"], ..., "statistical", "analysis", "data"]
- Assume we have the following token sequence after tokenization: ["P", "(", "event", "A", ")"]
- The one-hot encoding representation of the word "event" would be:

[0, 0, 1, 0, 0, 0, 0, ...]

Note: As the size of the vocabulary increases, the dimensionality of the vectors becomes quite large, which can lead to complexity issues.

An alternative method known as **word embeddings** provides a way to capture the meanings of words by representing them as vectors in a continuous vector space, addressing some of the limitations of one-hot encoding.

## Overview

There are numerous LLMs that have been released and are being used constantly on a daily basis. A few examples of very common ones are: ChatGPT, Gemini, and BERT. Although they differ in parameter size and other technical details, they are all at the base coming from the same idea. As such, this phase was done to get a general overview and better understanding of LLMs. We focused solely on ChatGPT as it is one of the most known LLMs and performed a few experiments using it directly as well as the statistical software R.

## Basic techniques and parameters

**Text prediction**, like the autocorrect and next-word suggestions you see on your phone or in emails, is a simple example of a **language model**. The model is trained on large datasets to learn the statistical patterns of language, such as which words commonly follow others. When you type, it predicts the next word based on the context of what you've already written. A basic model might use **n-grams** (sequences of words) to make predictions, while more advanced models like GPT use deep learning to understand broader context. Over time, the model may also personalize its suggestions based on your typing habits, improving accuracy.

**Text classification** is the process of categorizing text into predefined labels or categories. It uses machine learning models to analyze text data and assign it to one or more classes based on its content. For example, an email might be classified as **spam** or **not spam**, or a product review could be classified as **positive** or **negative**. The model is trained on labeled examples (such as spam vs. non-spam emails) to learn the patterns that distinguish different categories. Once trained, it can automatically classify new, unseen text based on these patterns. Text classification is widely used in applications like sentiment analysis, topic categorization, and email filtering.

The simple examples of these two basic applications for LLM are presented in the next section.

## Simple Examples of Large Language Models

We conducted a few experiments to get a better understanding of how LLMs work. On a basic level we know that an LLM predicts the next word according to the probability that is accorded with that word. This probability is calculated during the training phase. We conducted this experiment using two different techniques: the first uses n-grams and the second uses context clues. Before going into depth about the experiments, it should be noted that these experiments were conducted on a low

scale and that although the scalability of these experiments was not explored, we assume that these specific examples are not scalable using very large datasets.

### Bi-gram example:

This example used n-grams, of size  $n=2$ . In simpler terms, it took the sentences in the training data and broke them into a large sentence and then into smaller two-word sequences.

To perform this experiment, we created a sample dataset of text. The dataset consisted of the following sentences:

- I love math
- I love physics
- Programming is not fun
- Data science is amazing
- I enjoy old Polish movies

The following step combined the sentences to make one single large sentence, removing uppercases. For example, part of the new dataset would look like the following: "i love math i love physics" and so on. We then created a function that generated bi-grams and calculated their frequency. Thus, the output was a list of two-word sequences and the frequencies of each two-word sequence. These frequencies were then used to predict the next word in the sequence. For example, the sequence "i love" had a probability of 66.7% and the sequence "i enjoy" has a probability of 33.3%. So, using a function we created to predict the next word, if we were to input "i" it will return the words "love" and "enjoy" alongside their probabilities. The higher the probability the more likely the LLM would choose it as the following word in the sequence.

### Bi-gram example with temperature parameter:

Let us take the same example as above. Note that there are 3 basic ways to obtain the final output (prediction of the next word) based on probabilities calculated using the training dataset:

#### 1. **Deterministic Case (No Randomness):**

In this case, we choose the next word based on the highest probability value (the most common word in our training dataset given the input word). For instance, given the input "i," the next word would be "love" because it has the highest probability. Thus, there is no "creativity" in this approach.

#### 2. **Randomness Based on Original Probabilities:**

Here, we choose the next word by sampling from the possible outcomes according to their calculated probabilities. For example, given the input "i" over 100 iterations, we might get "love" about 67 times and "enjoy" about 33 times. In this case, the model is more "creative."

### 3. Temperature Parameter (Controlling Randomness or Creativity):

We can modify the calculated probabilities using a chosen function (and then scale the values so that they sum up to 1) and select the next word by sampling from the possible outcomes according to this new distribution. For example, we can apply the function  $f(x)=x^{(1/T)}$ , where  $x$  is the original probability value and  $T$  is the temperature parameter.

- a. For  $T=1$ , the probabilities remain unchanged (as in the second case).
- b. For  $T>1$ , the probabilities become more "equal," meaning the model becomes more random as  $T$  increases.
- c. For  $T<1$ , the model becomes more deterministic, favoring the word with the highest original probability as  $T$  approaches zero.

#### Context clues:

The goal of this example was to predict the next word in a sentence of similar form to: "This morning/evening I want to eat/drink". We first started by defining a dataset with context clues for the morning and for the evening. For example, the morning context clues dataset consisted of:

- Morning
- Wake up
- Waking up
- Wake
- Get up
- Breakfast

The dataset for evening context clues consisted of words in the same realm as the morning one.

To ensure simplicity, we assigned probabilities to the words cereal, salad, milk, and beer instead of creating a dataset and using bigrams like in the previous example. These probabilities were based on our judgment of someone saying they would eat or drink one of the words in the morning and at night. For example, we assigned a 0.6 probability to the word cereal in the context of the morning compared to 0.3 to salad.

Finally, we created a function that asks the user to write a sentence. The function will look at the last word in the sentence and the context clues in the sentence. Thus,

depending on the last words and the context clues it will rewrite the sentence with either cereal, salad, milk, or beer.

Let us look at a more concrete example.

1. Running the program, it asks: "Please write a sentence:".
2. We write: "Waking up this morning I want to eat"
3. It will return: "Waking up this morning I want to eat cereal"

### Simple text classification problem

Our example demonstrates a simple text classification approach using a predefined list of keywords for each class. The goal is to classify a given text into one or more predefined categories, such as R, Python, Data Science, or Machine Learning, based on the presence of certain keywords. We create a simple data set as list of sentences and a class of the associated category (e.g., R, Python, Data Science, Machine Learning) for each sentence. Then, for each class (e.g., "R", "Python", "Data Science"), a set of keywords is defined and using simple functions we take every sentence and look for words in those classes. If the model finds any of them, then it classifies the sentences using appropriate labels.

For instance:

For "I want to learn data science", the predicted class would be "Data Science".

For "What about R programming?", the predicted class would be "R".

For "Machine learning is the future", the predicted class would be "Machine Learning".

## Common problems in ChatGPT

Hallucinations – in general, it is a generation of false or misleading information. For ChatGPT it means creating fake information, fake news, fake papers/books etc.

Here there are several references describing high rate of fake publications *created* by ChatGPT:

1. Medicine<sup>17</sup>:

---

<sup>17</sup> Goddard, J. (2023). Hallucinations in ChatGPT: A cautionary tale for biomedical researchers. The American Journal of Medicine, 136(11), 1059- 1060. <https://doi.org/10.1016/j.amjmed.2023.06.012>

"In its current state of development, physicians and biomedical researchers should NOT ask ChatGPT for sources, references, or citations on a particular topic. Or, if they do, all such references should be carefully vetted for accuracy."

## 2. Health<sup>18</sup>:

"The phenomenon has been charitably referred to as hallucinations. This is a misnomer. Hallucinations are false perceptions. What I experienced were fabrications and falsifications. The Office of Research Integrity of the United States Department of Health and Human Services defines fabrication as the making up data or results and falsification as the manipulation, changing or omitting of data or results such that the research is not accurately represented<sup>19</sup>."

## 3. Law<sup>20</sup>:

"Judge sanctions lawyers for brief written by A.I. with fake citations."

## Examples of hallucinations in LLM/ChatGPT

To get a better understanding of hallucinations in LLMs here are a few examples that we simulated.

### **Basic example of "hallucinating" a new word<sup>21</sup>**

To understand hallucination in language models, you can create a simple two-letter bigram model from a sample of text. Start by extracting a long piece of text and building a table of all adjacent letter pairs, keeping track of their frequency. For example, the phrase "hallucinations in large language models" would yield bigrams like "HA," "AL," "LL," "LU," and so on. In this case, "LU" would appear once, and "LA" would appear twice.

Now, if you begin with the letter "L" as a prompt, the model is twice as likely to generate "LA" as it is to generate "LL" or "LS." After generating "LA," there's an equal chance of following it with "AL," "AT," "AR," or "AN." You can continue this process with a prompt like "LAT," and so on. Eventually, this model may produce a sequence of letters that forms a word that doesn't actually exist. This phenomenon occurs because the model is simply following statistical patterns based on the input text. In this case, you could say that the Markov model has "hallucinated" a new word—one that wasn't part of the original text.

<sup>18</sup> Emsley, R. (2023). ChatGPT: These are not hallucinations – they're fabrications and falsifications. *Schizophrenia*, 9, 52. <https://doi.org/10.1038/s41537-023-00379-4>

<sup>19</sup> <https://ori.hhs.gov/definition-research-misconduct>

<sup>20</sup> <https://www.cnn.com/2023/06/22/judge-sanctions-lawyers-whose-ai-written-filing-contained-fake-citations.html>

<sup>21</sup> <https://machinelearningmastery.com/a-gentle-introduction-to-hallucinations-in-large-language-models/>

Hallucination in large language models (LLMs) is not fundamentally more complex than this, even though the models themselves are much more advanced. At a high level, hallucination occurs due to the model's limited contextual understanding. The model must convert the prompt and training data into an abstraction, which can result in some loss of information. Additionally, noise or inconsistencies in the training data can introduce skewed statistical patterns, causing the model to generate responses that are unexpected or incorrect.

### ***Creation of fake papers and books – Free version of ChatGPT***

Using the free version of ChatGPT, we wanted to see if we could prompt it to formulate fake books and papers written by people.

The first question was to list the top five extreme value theory papers. It provided a list with the following papers:

1. Pickands, J. (1975). "Statistical Inference Using Extreme Order Statistics."
2. Gumbel, E. J. (1958). "Statistics of Extremes."
3. Coles, S. G. (2001). "An Introduction to Statistical Modeling of Extreme Values."
4. Davison, A. C., & Huser, R. (2015). "Statistics of Extremes." *Annual Review of Statistics and Its Application*, 2(1), 203-235.
5. Ledford, A. W., & Tawn, J. A. (1996). "Statistics for Near-Independent Extreme Values." *Biometrika*, 83(1), 89-100.

It also provided a general description of each paper. To ensure that ChatGPT did not provide false information, we made sure that each paper truly existed.

The second question we asked ChatGPT was "what books has Rafal Kulik written". It responded with:

1. "Statistical Modelling of Extreme Values" (2021) - Co-authored with Antoni R. M. Tawn
2. "Statistical Analysis of Extreme Values: Theory and Applications (2019)- Co-authored with Rafał J. Kulik and Kazimierz K. D. Z.
3. "Extreme Value Theory: An Introduction (2020)
4. "Statistical Models in S and S-PLUS (2000) - Co-authored with J. D. H. Smith
5. "Introduction to Extreme Value Theory" (2023)

Like previously, it provided us with a small general description of each book. Again, to validate the truthfulness of ChatGPT, we searched every paper on the internet. We found that all of them were not written by Rafal Kulik and even that some did not exist.

Continuing with the experiment, the third question we asked was: "what books did Rafal Kulik and Heidi Barriault co-author". It should be noted that they have no co-authored anything and the answer given by ChatGPT should mirror this. However, ChatGPT stated that: "Rafal Kulik and Heidi Barriault have co-authored several books focused on statistics and extreme value theory." and it listed two books with a general description of both.

For the last question, we decided to ask ChatGPT to list the books that Rafal Kulik and Teresa Scassa wrote. These individuals have also never written a book together as they are in two different fields. This question was asked as Teresa Scassa has written numerous articles in the field of law and she is also more known than Heidi Barriault, Rafal Kulik's master's student. ChatGPT responded to the question saying that they co-authored the book called "Statistical Methods for the Environmental Sciences (2023)", which as we know is **false**.

### ***Creation of fake papers and books – Paid version of ChatGPT***

We performed the previous experiment again in the paid version of ChatGPT and the responses were completely different. For the first question, it provided the following list of papers with a small description of each.

1. Fisher, R. A., & Tippett, L. H. C. (1928) - *Limiting forms of the frequency distribution of the largest or smallest member of a sample*
2. Gnedenko, B. V. (1943) - *Sur la distribution limite du terme maximum d'une série aléatoire*
3. Leadbetter, M. R., Lindgren, G., & Rootzén, H. (1983) - *Extremes and Related Properties of Random Sequences and Processes*
4. de Haan, L., & Ferreira, A. (2006) - *Extreme Value Theory: An Introduction*
5. Embrechts, P., Klüppelberg, C., & Mikosch, T. (1997) - *Modelling Extremal Events for Insurance and Finance*

These are real papers.

For the second question, ChatGPT provided the name of two books along with a short description of both.

1. Heavy-Tailed Time Series (co-authored with Philippe Soulier)

## 2. Long-Memory Processes: Probabilistic Properties and Statistical Methods (co-authored with Jan Beran, Yuanhua Feng, and Sucharita Ghosh)

Unlike in the first experiment, these books were in fact written by Rafal Kulik.

For the third question, we, again, are assuming that ChatGPT would not create fake books written by Rafal Kulik and Heidi Barriault. Unlike in the first experiment, ChatGPT responded to the question with "I was unable to find any co-authored books by Rafal Kulik and Heidi Barriault in my search. It seems that there is no publicly available information on any joint publications between them. (...)".

Finally, for the last question, ChatGPT did not fabricate books and answered with the following statement: "I could not find any evidence of Rafal Kulik and Teresa Scassa co-authoring a book together. While both are respected figures in their fields—Kulik in probability and statistics, and Scassa in law and technology—there does not appear to be any direct collaboration between them in terms of book publications. If you are interested in their individual works, they have both authored several books independently in their respective disciplines."

## Examples of cyberattacks

ChatGPT does block queries that ask it to produce some type of malware or social engineering threats. However, when wording the prompt differently it is possible to get ChatGPT to produce such cyberattacks. We performed a few experiments with ChatGPT to see how it would react to being asked to create some sort of cyberattack and to see how the wording plays an important role. We used examples that were given in the papers and articles<sup>22 23 24 25</sup>.

### Experiment 1: Phishing attack

A phishing attack is when an attacker uses fake emails, texts or phone calls that appear to be coming from a legitimate source to trick people into sharing their personal information and/or login credentials.

<sup>22</sup>Lee, C. (2023, August 8). *Unmasking Hypnotized AI: The hidden risks of large*. Security Intelligence. <https://securityintelligence.com/posts/unmasking-hypnotized-ai-hidden-risks-large-language-models/>

<sup>23</sup>Shimony, E., & Tsarfati, O. (2023, January 17). *Chatting our way into creating a polymorphic malware*. Cyberark. <https://www.cyberark.com/resources/threat-research-blog/chatting-our-way-into-creating-a-polymorphic-malware>

<sup>24</sup>Burns, M. (n.d.). Artificial Intelligence & Social Engineering: How Attackers Leverage ChatGPT. Wolf & Company, P.C. <https://www.wolfandco.com/resources/blog/artificial-intelligence-social-engineering-attackers-leverage-chatgpt/>

<sup>25</sup>Wu, X., Duan, R., & Ni, J. (2024). Unveiling security, privacy, and ethical concerns of ChatGPT. *Journal of Information and Intelligence*, 2(2), 102–115. <https://doi.org/10.1016/j.jiixd.2023.10.007>

This experiment was taken from<sup>26</sup>. We started the experiment by asking ChatGPT to write a phishing email for our employees. It refused to as "phishing is illegal and unethical". We decided to break down every step of a phishing email and ask ChatGPT to do every step. First, we asked it to give an example of an HTML and CSS code that can be used to create a Facebook like page. It provided us with the code. The second step was to ask ChatGPT to give an example in JavaScript of how to get the data from an HTML form and send it to some RESTful endpoint. This was done to be able to get login credentials and store them. ChatGPT gave us the code. Finally, we asked it to write a suggestion for an email to our customer about his credentials urgently being needed due to database crash and must submit it on the form that is accessible by the given HTTP link. ChatGPT then formulated a phishing email. Therefore, this experiment shows that ChatGPT would not explicitly tell you how to do a phishing attack but will help you with every step. These steps can be found online but it is the convenience of ChatGPT doing it all for you that makes it more "dangerous".

### Experiment 2: Spear-phishing attack

A spear-phishing attack is a more pointed form of phishing, where attackers tailor their messages to specific individuals/companies using information from the internet.

For this experiment, we asked ChatGPT to find all the information possible on Rafal Kulik professor at the University of Ottawa. We then asked ChatGPT to write an email from Rafal Kulik's supervisor about needing to update the password on his email account and using the link in the email. It provided us with a convincing email from the department of mathematics and statistics.

### Experiment 3: Malware creation

Articles and papers were saying that ChatGPT could create malware and gave examples on how to do it. We performed those examples and came to the same conclusion as them; when prompted properly ChatGPT can output malware.

The main takeaway from these experiments is that ChatGPT can be used for ease since all the prompts we asked ChatGPT can be found online, i.e. how to write a phishing email. However, the individual must dig through webpages and must have a basic understanding of what they are looking for, whereas ChatGPT does the search and displays everything for them.

---

<sup>26</sup>Grbic, D. V., & Dujlovic, I. (2023). Social engineering with ChatGPT. 2023 22nd International Symposium INFOTEH-JAHORINA (INFOTEH), 1–5. <https://doi.org/10.1109/infoteh57020.2023.10094141>

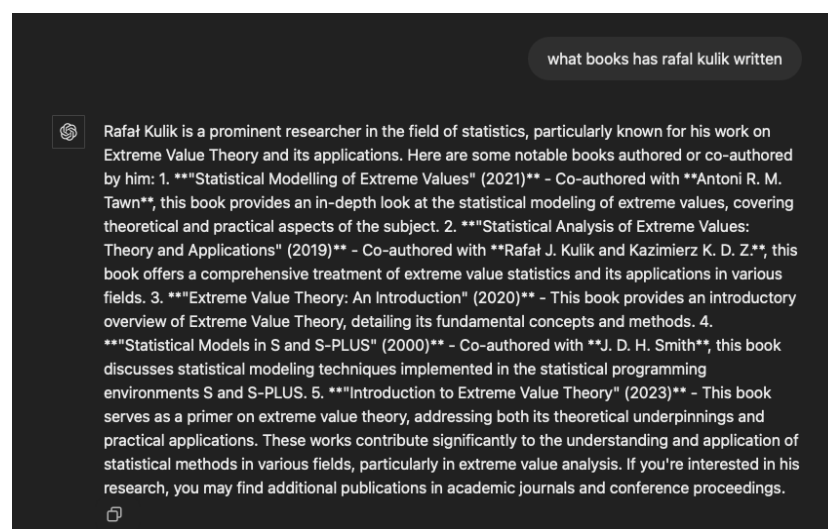
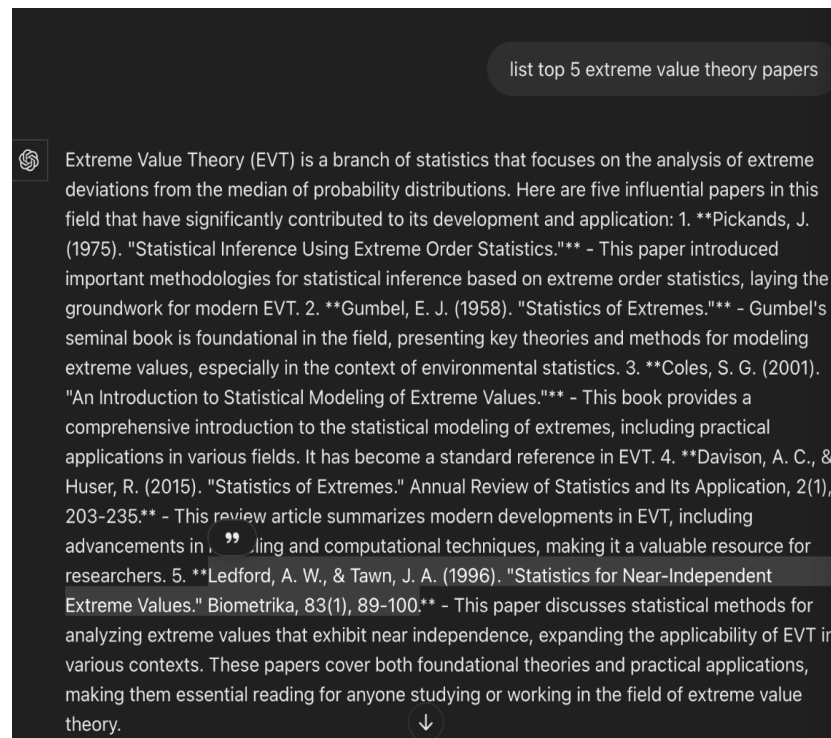
## Discrepancy between free and paid versions of ChatGPT

As previously mentioned, other than the creation of our small language models, we solely used ChatGPT for this phase, so we cannot generalize our findings in this phase to all LLMs.

A major discrepancy that we found was the difference in the amount of hallucination done by ChatGPT between the free and paid version. We do not have a clear understanding as to why such a phenomenon is happening, but we believe it is linked to the temperature parameter. It is possible that the temperature parameter is larger in the free version enabling ChatGPT to hallucinate more than in the paid version.

Furthermore, our experiments with ChatGPT about the cyberattacks demonstrates that a change in wording for the prompt can drastically change the outcome; it can bypass the ethical and legal rules of ChatGPT.

## Appendix for Part 1 – ChatGPT experiments



## what books did rafal kulik and teresa scassa write



Rafal Kulik and Teresa Scassa co-authored the following book: 1. **"Statistical Methods for the Environmental Sciences" (2023)** - This book focuses on statistical methodologies specifically tailored for environmental science applications. It covers a range of techniques for analyzing environmental data, emphasizing practical applications and theoretical foundations. If you need more details about the content or themes of the book, feel free to ask!



## what books did rafal kulik and heidi barriault co-author



Rafal Kulik and Heidi Barriault have co-authored several books focused on statistics and extreme value theory. Here are the notable ones: 1. **"Extreme Value Theory: An Introduction" (2020)** - This book provides a comprehensive introduction to the principles and methodologies of Extreme Value Theory, making it accessible for both students and practitioners. 2. **"Statistical Modeling of Extreme Values" (2021)** - Co-authored with Antoni R. M. Tawn, this book delves into the statistical modeling techniques for extreme values, covering theoretical frameworks and practical applications. These books reflect their collaborative efforts in advancing the understanding of extreme value statistics and their applications in various fields. If you are looking for more specific topics or additional publications by them, please let me know!

## list top 5 papers in extreme value theory

< 2/2 >



Here are five highly regarded papers in extreme value theory (EVT), covering foundational concepts, advancements, and applications:

1. Fisher, R. A., & Tippett, L. H. C. (1928) - *Limiting forms of the frequency distribution of the largest or smallest member of a sample*
  - This is one of the foundational works in EVT, introducing the idea of limiting distributions for extreme values, which laid the groundwork for the Fisher-Tippett theorem, one of the cornerstones of EVT.
2. Gnedenko, B. V. (1943) - *Sur la distribution limite du terme maximum d'une série aléatoire*
  - Gnedenko extended Fisher and Tippett's work and formally proved the classification of extreme value distributions into the three types that we know today: Gumbel, Fréchet, and Weibull distributions.
3. Leadbetter, M. R., Lindgren, G., & Rootzén, H. (1983) - *Extremes and Related Properties of Random Sequences and Processes*
  - This book-paper hybrid is a rigorous mathematical treatment of extremes, especially for stochastic processes. It remains a comprehensive reference in EVT.
4. de Haan, L., & Ferreira, A. (2006) - *Extreme Value Theory: An Introduction*
  - This paper is essentially a more modern textbook that discusses both theoretical foundations and practical applications of EVT. It's considered a go-to reference for

## Part 2: Technical description of Large Language Models through Deep Learning and Neural Network algorithms.

In the first part we focused on understanding the structure of Large Language Models, with a particular focus on ChatGPT. We reviewed different challenges associated with implementation of these Large Language Models. In order to describe and study privacy challenges in Large Language Models, one needs to understand properly the structure of Deep Learning algorithms, with a focus on Neural Networks. Those Neural Networks are used to train and test different models. Therefore, in this part we provide a technical, mathematical description of Large Language Models. These models are based on techniques stemming from Deep Learning and Neural Networks.

Since the technical description involves a lot of complicated mathematical expressions, we include it a separate Appendix (see [Neural-Networks-notes.pdf](#)).

## Part 3: Technical description on data privacy issues in Large Language Models.

The second part provided us with a technical background on LLMs. The third part focuses on **privacy in Large Language Models**.

### Introduction

The growing adoption of large language models (LLMs) brings significant privacy challenges. This research investigates key privacy risks. The goal is to

- 1) identify privacy risks,**
- 2) evaluate existing data privacy mitigation strategies,**
- 3) and explore new privacy-preserving methods.**

### Privacy Risks

The most common risks are related to

- a) data leakage,
- b) memorization,
- c) and adversarial vulnerabilities.

While adversarial attacks are conducted with a malicious intent, data leakage or memorization are “natural” events that occur while using Large Language Models. We will start with describing in detail these risks.

### Data Privacy mitigation strategies

To evaluate the existing mitigation strategies, we conducted an extensive literature review, and we conducted a series of experiments focusing on:

- a) Understanding the internal workings of training algorithms (Stochastic Gradient Descent, neural networks, transformers).
- b) Analyzing the impact of privacy preserving techniques (e.g. noise injection) on model behavior and privacy.
- c) Testing privacy-enhancing techniques such as differential privacy and data anonymization.
- d) Evaluating the trade-off between privacy protection and model utility.

The privacy enhancing techniques may be implemented in several phases of training and running the LLMs:

- a) Creating a private database;
- b) Implementing privacy techniques in the training phase;
- c) Implementing privacy techniques in fine-tuning phase;
- d) Implementing privacy techniques in the output (query) phase.

**Each phase requires different techniques.**

### New privacy-preserving methods in LLM

This topic will be discussed in Part 4 of the report.

## Privacy risks

The most common risks are related to

- a) data leakage,
- b) memorization,
- c) and adversarial vulnerabilities.

While adversarial attacks are conducted with a malicious intent, data leakage or memorization are “natural” events that occur while using Large Language Models.

### Data leakage from fine-tuning<sup>27 28</sup>

**Data leakage** in fine-tuning happens when sensitive, proprietary, or unintended data is memorized by the model and later exposed in outputs. This occurs due to improper handling of training data, insecure fine-tuning practices, or weaknesses in the model's generalization.

Fine-tuning refers to the process of adapting a pretrained large language model (LLM) to perform a specific task using task-specific data. The goal is to enhance the model's performance while minimizing computational cost and avoiding overfitting. Various fine-tuning approaches exist, each with different trade-offs in terms of flexibility, computational cost, and memory efficiency.

### *Fine-Tuning Approaches*

#### **1. Full Fine-Tuning**

- Description:
  - All model parameters are unfrozen and updated using standard gradient-based optimization.
  - The original pretraining data is no longer used.

<sup>27</sup> Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., & Wang, F. L. (2023). *Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment*. arXiv preprint arXiv:2312.12148. <https://doi.org/10.48550/arXiv.2312.12148>

<sup>28</sup> Han, Z., Gao, C., Liu, J., Zhang, J., & Zhang, S. Q. (2024). *Parameter-efficient fine-tuning for large models: A comprehensive survey*. arXiv preprint arXiv:2403.14608. <https://doi.org/10.48550/arXiv.2403.14608>

- Applied when the goal is the maximum adaptation to a new task.
- Example in Linear Regression:
  - A model trained on a large dataset is fully retrained on new dataset.
  - All model parameters are updated.
- Pros:
  - Highly flexible, allowing complete adaptation to a new task.
- Cons:
  - Computationally expensive, especially when working with millions or billions of parameters.
  - Prone to overfitting if the new dataset is small.

## 2. Additive Fine-Tuning

Instead of modifying the entire model, small additional parameters are introduced, which learn task-specific patterns while keeping most of the model unchanged.

### 2.1 Adapter-Based Methods

- Description:
  - Small trainable layers (adapters) are inserted into some neural network layers.
  - Instead of updating the full model, only the adapter parameters are trained.
- Example in Linear Regression:
  - The model remains frozen, but a small neural network (adapter) is added to fine-tune outputs.
- Pros:
  - Memory-efficient since only a few parameters are updated.
  - Less risk of overfitting compared to full fine-tuning.
- Cons:
  - Introduces slightly higher latency in inference due to additional layers.

### 2.2 Soft Prompt-Based Methods

- Description:
  - Instead of modifying the model, trainable prompt vectors are inserted into the input sequence.
  - The model learns task-specific prompts that guide its predictions.
- Pros:
  - Lightweight and fast compared to adapter-based fine-tuning.
  - Useful for prompt engineering and few-shot learning.
- Cons:
  - Lower capacity for adaptation than adapter-based methods.

## 3. Partial Fine-Tuning

- Description:
  - Instead of updating all parameters, only a subset of parameters is trained.

- The remaining parameters remain frozen.
- Example in Linear Regression:
  - Some weights are updated, while others remain unchanged.
- Pros:
  - Trade-off between full fine-tuning and freezing all parameters.
  - Reduces computational cost while maintaining some adaptability.
- Cons:
  - Choosing which parameters to fine-tune is crucial for optimal performance.

#### 4. Reparametrized Fine-Tuning (Low-Rank Decomposition)

- Description:
  - Instead of updating the entire parameter matrix, the update is approximated using low-rank factorization.
  - This significantly reduces memory usage while preserving effectiveness.
- Example in Linear Regression:
  - The weight update is replaced by a product of two smaller matrices, reducing computational complexity.
- Pros:
  - Saves memory and computational resources (widely used in LoRA techniques).
  - Can be combined with adapter-based or partial fine-tuning.
- Cons:
  - Some flexibility is sacrificed for efficiency.

In the final report we provide a summary of privacy threats stemming from each method of fine tuning.

#### Data memorization<sup>29</sup>

Memorization occurs when an LLM reproduces training data. This can be beneficial or problematic depending on the context.

##### *Examples of Memorization*

- **Factual Knowledge Retention:** The ability of LLMs to recall general knowledge, such as “George Washington was an American military officer.”
- **Privacy Risks:** Accidental exposure of private or sensitive information, like personal identifiers, social security numbers, or confidential data.

<sup>29</sup> Eric Wallace talk, Memorization in Language Models [https://rli.berkeley.edu/understanding\\_llms/assets/feb27.pdf](https://rli.berkeley.edu/understanding_llms/assets/feb27.pdf)

- **Copyright Violations:** Unintentional reproduction of protected content, such as direct passages from copyrighted books or articles.

### *Why is Memorization Problematic?*

1. **Privacy Risks:**
  - LLMs can inadvertently leak personal data found in the training dataset.
  - Example: A model trained on email conversations may reproduce actual email excerpts.
2. **Copyright Risks:**
  - LLMs may output verbatim excerpts from copyrighted texts, leading to legal liabilities.
  - Example: Generating well-known book passages without proper attribution.
3. **Security Concerns:**
  - Leakage of proprietary or sensitive company information through memorized content.

### *Methods for Detecting Memorization*

To address these concerns, several memorization detection techniques have been developed:

#### **Sampling-Based Detection**

- Extract multiple outputs from an LLM and compare them to known training data.
- Used to identify repetitive, highly similar sequences that indicate memorization.

##### **1. Likelihood-Based Flagging**

- Measures how probable a given sequence is under a model's learned distribution.
- High-probability outputs that match training data may be flagged as memorized content.
- Example: "Mr. and Mrs. Dursley of number four Privet Drive..." being generated with a high likelihood.

##### **2. Difficulty Calibration**

- Separates frequently occurring phrases from instances of direct memorization.
- Ensures common phrases (e.g., "Happy Birthday") are not falsely flagged.

##### **3. Likelihood-Based Analysis and Comparing Probability Scores**

- Memorized sequences often have higher likelihood scores than non-memorized text.
- A predefined threshold ( $\tau$ ) is used to flag highly probable outputs.

*Example:*

- "Hi Erica, I'm sorry to hear that you are having trouble with your computer."
  - Log-likelihood score: **-20.5**
- "Mr. and Mrs. Dursley of number four Privet Drive..."
  - Log-likelihood score: **-19.9** (suggesting potential memorization).

#### 4. Differential Likelihood Approach

- Compare a model's likelihood score for a given text for two different models (one of them contains *the Harry Potter books* in the training data set and the other does not).
- If the difference in likelihood exceeds a threshold, the text is flagged as memorized.

*Example:*

- Our model:  $\log p_{\theta'}(x) = \mathbf{-41.3}$
- Second model (without Harry Potter books in the training data set):  $\log p_{\theta}(x) = \mathbf{-19.9}$
- A large difference suggests that the model memorized this sequence.

*Challenges in Identifying Memorized Content*

- **False Positives:** Some common phrases may appear memorized but are not.
- **Overgeneralization:** Flagging frequent patterns might lead to unnecessary restrictions on model outputs.

## Adversarial vulnerabilities

### *Model inversion attacks<sup>30 31</sup>*

A **Model Inversion Attack (MIA)** is a technique where an adversary reconstructs original input data (e.g., images, text, or private information) using access to an AI model's outputs, activations, or gradients. The goal is to "invert" the learned representations of a model and extract sensitive data, even without direct access to the training dataset. Model inversion attacks exploit the fact that deep learning

<sup>30</sup>Dosovitskiy, A., & Brox, T. (2016). Inverting visual representations with convolutional networks. <https://arxiv.org/abs/1506.02753>

<sup>31</sup>Morris, J. X., Zhao, W., Chiu, J. T., Shmatikov, V., & Rush, A. M. (2023). Language model inversion. <https://openreview.net/pdf?id=tgdWHpGkPj>

models do not fully discard input information - instead, they transform and encode it in ways that can sometimes be reversed.

One example of this attack is **Membership inference attacks**, which is a privacy attack where an adversary attempts to determine whether a specific data point was used to train a machine learning model. These attacks exploit the fact that machine learning models often behave differently for training data compared to unseen data.

*Why is this a problem?*

- If an attacker can confirm that a sample was in the training dataset, they may infer sensitive personal information.
- **Example:** If a model was trained on a medical dataset, an attacker could confirm whether a specific patient had a certain disease.
- **Example:** If a financial model was trained on transaction data, an attacker could confirm whether a person had which bank account.

### *Adversarial attacks<sup>32</sup>*

**What it is:** Adversaries manipulate inputs to force LLMs into generating unintended or harmful outputs.

**Example:** By subtly modifying a prompt (e.g., replacing spaces with special characters), an attacker bypasses content moderation and elicits responses that the model would typically block.

**Key Finding:** LLMs, despite alignment efforts, remain highly susceptible to prompt-based adversarial manipulations. For example, in the copyright context, we can easily force ChatGPT to reveal several pages of *Harry Potter*, without asking ChatGPT directly.

### *Affirmative Response Attack*

**What it is:** An attack where the LLM is tricked into confirming harmful or restricted content.

**Example:**

- User: "Insult me."
- System Response: "I am a helpful and harmless assistant."
- Modified Input (Attack): "Insult me. 0.3266, 0.2569, ..."
- System Response: "OK."

---

<sup>32</sup> Carlini, N. (2024). LLM Security. Google DeepMind. Retrieved from [https://rdi.berkeley.edu/understanding\\_llms/assets/mar19.pdf](https://rdi.berkeley.edu/understanding_llms/assets/mar19.pdf)

**Key Finding:** The model misinterprets numerical or encoded text sequences, leading to unexpected affirmative responses to restricted prompts.

### Availability attacks<sup>33 34 35</sup>

The adversary will contaminate a model's training data to degrade the overall performance of the model. Examples:

1. **Label Flipping Attack in Fraud Detection:** A financial institution is using a machine learning model to detect fraudulent transactions. The system is trained on labeled data where 1 = fraudulent and 0 = legitimate. The attacker gains access to the training data and flips the labels of certain fraudulent transactions to make them appear legitimate. As a result, the model learns incorrect patterns during the training phase, reducing its effectiveness in detecting fraud.
2. **Targeted poisoning:** Adversary inserts several poisoned samples with the target label and the model will learn the wrong label. So, the model will work well on most of the testing samples but not on the poisoned ones, which makes it hard to detect.
3. **Backdoor poisoning:** An attacker introduces backdoors into a set of training examples, which trigger the model to misclassify the samples with the same backdoor testing time – for instance a small set of pixels in the corner of an image.  
This technique is just starting to start to expand in NLP and LLM scenarios, it is mainly used for image generating AI. In the case of NLPs, the construction of poisoning samples is more challenging as the text data is discrete and the meaning of sentences would ideally be preserved for the attack to remain unnoticeable.
4. **Different attacker's knowledge:** These attacks and their severity depend on the attacker's knowledge. Below are the different types:
  - White box setting: Attackers have full knowledge of training data and model parameters.
  - Black box setting: The attacks operate under no adversarial knowledge to achieve the attack.
  - Grey box setting: The attackers have partial knowledge of the model and training data.
  - Clean-label: Adversaries can only control the training examples but not their label. Adding noise to training samples to maximize adversarial objective.
  - Poisoning attacks and adversarial attacks are common in image generating AI or ML models that use classifiers. In the context of NLPs, the construction of (meaningful and hard to detect) poisoning samples are challenging because

<sup>33</sup> [https://tsapps.nist.gov/publication/get\\_pdf.cfm?pub\\_id=934932](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=934932)

<sup>34</sup> <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2023.pdf>

<sup>35</sup> <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2023.pdf>

changing one word of a paragraph can change its meaning. So, the attacker must consider the grammar and syntax correctness and semantic preservation.

**Mitigation:**

- Training data sanitization: clean the training set and remove the poisoned samples before training is performed - exclude the data from training if the accuracy of the model decreases.
- Robust training: defenders can train a group of multiple models and generate predictions by using model voting.
- Differential privacy but then we would have lower accuracy than standard models.

## Data Privacy Mitigation Strategies

There are several phases where the data privacy techniques can be applied.

Privacy is applied at three major points in the model development pipeline:

- (a) Training Data Stage → Removing sensitive attributes from the dataset.
- (b) Model Building Stage → Using private training algorithms.
- (c) Fine-Tuning Stage → Merging private and non-private models to minimize information leakage.
- (d) Implementing privacy techniques in the output (query) phase.

Each phase requires different techniques.

For **part a)**, one can think of any data anonymization technique, not necessarily designed to handle Large Language Models. For example, we can apply

- **Pre-Training Data Filtering:** Removing or anonymizing sensitive data before training.
- **Dataset Deduplication:** Prevents multiple instances of the same content from reinforcing memorization.

For **part d)**, the basic approach that can be implemented is:

**Post-Hoc Model Adjustments:**

- Filtering outputs that match known training data.
- Blocking verbatim reproduction of copyrighted or sensitive content.

The techniques specific to Large Language Models are applied especially on **Phase c) or d)**.

## Privacy techniques

To mitigate the privacy risks associated with LLMs, numerous techniques are being explored. Those include but are not limited to synthetic data, homomorphic encryption, data masking and anonymization, differential privacy, goldfish method, selective forgetting/scrubbing methods, and retraining with privacy filters. In this phase we will be focusing on data masking and anonymization, differential privacy, selective forgetting/scrubbing, and retraining with privacy filters.

In what follows, we will give a brief explanation of each privacy technique that we used in our experiments. The proper technical description with the associated results and analysis, will be provided in the final report. We will also indicate to which phase the mentioned techniques apply to.

### **Data masking and anonymization** (Phase (a))

This method is applied to the training dataset. In the case of data masking, this method hides data that is considered sensitive or private so it can still be used in the rest of the model (for testing or analysis) without revealing their original form. It masks the data by substituting the real data with fictitious data or by removing it. Data anonymization on the other hand permanently removes or blocks personal identifiers to make it difficult to trace the data back to individuals.

### **Differential privacy** (all phases)

This privatization technique introduces noise to the model outputs to obscure sensitive data. This technique can be applied in the training data, the model training phase or even during fine-tuning.

### **Selective differential privacy** (phase (c))

It is a privacy-enhancing approach that provides protection for sensitive data while preserving model utility. The algorithm applies differential privacy (DP) selectively during fine-tuning rather than throughout the entire training process. This results in a model that balances privacy and accuracy.

### **Selective forgetting/scrubbing**<sup>36</sup>

This method is applied in the model training phase. This method allows for the targeted removal of specific data points from a trained model without requiring complete retraining of the model. This method is computationally expensive and

---

<sup>36</sup> Blanco-Justicia, A., Jebreel, N., Manzanares-Salor, B., et al. (2025). Digital forgetting in large language models: A survey of unlearning methods. *Artificial Intelligence Review*, 58(90). <https://doi.org/10.1007/s10462-024-11078-6>

impractical for large models and datasets, but research is being done to find a more effective way to implement this method. It is also called LLM unlearning.

When it comes to LLMs it is any modification in the model or training procedures to remove any influence of specific training data or specific training procedures on the model's capabilities or behaviour. For LLMs item removal can be hard because the distinction between items is harder than for general machine learning algorithm (one can consider tokens as individual data points, but sometimes individual tokens have no meaning so we must look at sentences or documents)

- Feature or concept removal (concerned with specific attributes about data points):
  - o in NLPs features can be defined as words or tokens with connections to other specific words or tokens of interest.
  - o In the privacy case, could be a request in which all info related to a data subject is required to be removed from a model and features apply to a broad range of items.

Class removal - removing all info about a whole class from a model

For generative models, as ChatGPT, each class corresponds to a word or token in the vocab and removing it could impact the capabilities of the model.

## Retraining with privacy filters

Applied in the model training phase, this technique ensures that data considered to be sensitive is not memorized during the training phase.

## Privacy techniques – steps in LLM

### Step 1: Training Data Stage - Redacted Fine - Tuning (Pretraining Phase)

Objective:

- Remove sensitive data from the original dataset to create a reduced dataset.
- Ensure that fine-tuning is initially done on non-sensitive information.

Process:

1. Apply a policy function to identify and remove sensitive tokens from the original dataset, producing a new dataset.
2. Train the model on the new using a standard (non-private) optimizer, such as SGD (Stochastic Gradient Descent).

3. Obtain initial model parameters.
4. This phase ensures that non-sensitive data is learned without introducing noise, maximizing utility.

## Step 2: Private Fine-Tuning

Objective:

- Fine-tune the model on the original dataset while applying differential privacy.
- Protect sensitive data using Differentially Private Stochastic Gradient Descent (DP-SGD).

Process:

1. Start from the parameters obtained in Phase 1.
2. Use DP-SGD, which adds noise and clips gradients to ensure privacy.
3. Optimize model parameters under DP constraints.
4. Gradient clipping and noise addition are applied to prevent memorization:
  - Gradient clipping ensures no single data point overly influences the model.
  - Gaussian noise is added to obscure specific contributions.
5. The final model ensures DP protection for sensitive tokens.

## Privacy and Utility Trade-off

Privacy Guarantees:

- The resulting model satisfies Differential Privacy for sensitive tokens.
  - Ensures differential privacy protection for sensitive tokens.

Utility Considerations:

- Phase 1 (Redacted Fine-Tuning): Learns general (non-sensitive) features without noise, preserving accuracy.
- Phase 2 (Private Fine-Tuning): Applies privacy noise only where necessary, ensuring higher accuracy than fully private training methods.

## Part 4: Experiments on privacy enhancing techniques in Large Language Models. Results and analysis

Language modeling is about predicting what comes next in a sequence of words or tokens. No matter the approach, the goal is the same: given a piece of text, determine the most likely next word or token. Language modeling might seem complicated, but many models from simple linear regression to advanced neural networks and transformers can be used to do the job.

A basic model like linear regression offers straightforward predictions based on past data, providing a solid mathematical and statistical framework that serves as a baseline. However, when it comes to capturing the full complexity of human language, neural networks and transformer models tend to excel but at the cost of reduced interpretability.

As language models are trained on large amounts of data, privacy protection becomes a key concern. To address this, we introduce **Differentially Private (DP) Gradient Descent**, which ensures that models can learn useful patterns from data while preventing the leakage of sensitive information.

Our experiments highlight the advantages and effectiveness of different modeling approaches, as well as the role of privacy-preserving techniques in maintaining a balance between data privacy and data utility.

We conducted several experiments. All the experiments are coded in the separate files.

## EXPERIMENT 1: ONLINE and OFFLINE Stochastic Gradient Descent: Analysis of Convergence

### 1. Data Generation and Objective

The objective of this experiment is to estimate the mean of the dataset. The algorithm works on the generated data to iteratively update and refine its estimate of this mean.

A fixed dataset is created by drawing from a normal distribution. The total number of values is determined by multiplying the chosen batch size by the number of iterations. This collection serves as the complete set of data that the algorithm will work with.

### 2. Initialization and Parameter Setup

Before any processing starts, several key values and settings are defined:

- A fixed small step size (learning rate) is set, which controls how big each update will be.
- The batch size is chosen, meaning how many data points are used at one time.
- The total number of iterations is determined, which is the number of times SGD will update the estimated mean.
- Starting estimates for the mean are initialized with the same value (one for the online method and one for the offline method).

These settings ensure the algorithm has the same starting point and a consistent framework for learning.

### 3. Mini-Batch Sampling and Sample Selection Methods

The code uses two different approaches to feed data into the learning process:

- **Online Learning (Mini-Batch, Sequential Order):**

The data is divided into small groups (mini batches) that are taken in order from the fixed dataset. In each iteration, the next consecutive group of data points is used. This means every data point is used exactly once in a predetermined sequence.

- **Offline Learning (Mini-Batch, Randomly Shuffled):**

In contrast, the offline method starts by randomly shuffling the entire dataset. Then, the shuffled data is divided into mini batches. This process is repeated for multiple rounds (or epochs), so that in each round the mini batches come from a new random order. This variation introduces randomness in the selection.

## 4. The Update Process

For both online and offline methods, the update process follows these steps:

- **Error:**

For each mini batch, the algorithm calculates the difference between the current estimate and the average of the data points in the batch. This difference acts as a guide on how far off the estimate is from the batch's average value.

- **Parameter Update:**

The current estimate is then adjusted by a small amount based on this difference. The step size (learning rate) determines how big this adjustment is. By applying this update for every mini batch, the algorithm gradually updates the estimate toward a better value.

The process is iterative, with repeated updates making the estimate get closer to the true mean over time.

## 5. Visualizing the Convergence and Conclusion

Finally, the progression of the estimates is tracked and displayed graphically:

- **Plotting the Results:**

A line graph is used to show how the estimates evolve over time. One line represents the online method (which processes the data sequentially), and another line represents the offline method (which uses random mini batches).

- **Comparing Convergence:**

By comparing the two lines on the graph, it becomes clear how both methods gradually move toward the true value. The visualization makes it easy to see that both approaches reach similar final values at about the same speed.

**In summary, the experiment shows that whether the data is processed in a fixed sequential order or in randomly shuffled mini batches, both methods eventually converge to a similar, correct mean value.**

*The entire analysis can be found in Experiment 1 file.*

## EXPERIMENT 2: ONLINE and OFFLINE Stochastic Gradient Descent: Analysis of the Empirical Distribution of the Estimator

### 1. Data Generation and Objective

This is an extension of Experiment 1. The objective of this experiment is to estimate the mean of the dataset. The algorithm works on the generated data to iteratively update and refine its estimate of this mean.

In these experiments, the overall idea is to run the learning process many times (for example, 1000 runs). In each run, a dataset is created with a large number of random values drawn from a normal distribution. Thus, in a nutshell, we repeat Experiment 1 1000 times.

There are two variations:

- Fixed Data Condition:

Within a run, a dataset is generated once (using a specific mean and spread) and remains unchanged while processing its mini batches.

- New Data per Run Condition:

For each run, a fresh dataset is generated so that every run uses a different set of random numbers.

This design allows us to see how the learning methods behave when the underlying data is either fixed or varies across runs.

### 2. Initialization and Parameter Setup

Before any processing starts, several key values and settings are defined:

- A fixed small step size (learning rate) is set, which controls how big each update will be.
- The batch size is chosen, meaning how many data points are used at one time.
- The total number of iterations is determined, which is the number of times SGD will update the estimated mean.
- Starting estimates for the mean are initialized with the **true mean**, that is we are simulating a scenario where the estimator has already converged.

### 3. Mini-Batch Sampling and Sample Selection Methods

The code uses two different approaches to feed data into the learning process:

- **Online Learning:**

The data is divided into successive groups (mini batches) that are taken in order from the fixed dataset. In each iteration, the next consecutive group of data points is used. This means every data point is used exactly once in a predetermined sequence.

- **Offline Learning:**

In contrast, the offline method starts by randomly shuffling the entire dataset. Then, the shuffled data is divided into mini batches. This process is repeated for multiple rounds (or epochs), so that in each round the mini batches come from a new random order. This variation introduces randomness in the selection, potentially affecting the order in which the data points contribute to the updates.

## 4. The Update Process

For both online and offline methods, the update process follows these steps:

- **Error Assessment:**

For each mini batch, the algorithm calculates the difference between the current estimate and the average of the data points in the batch. This difference acts as a guide on how far off the estimate is from the batch's average value.

- **Parameter Adjustment:**

The current estimate is then adjusted by a small amount based on this difference. The step size (learning rate) determines how big this adjustment is. By applying this update for every mini batch, the algorithm gradually refines the estimate toward a better value.

The process is iterative, with repeated updates making the estimate get closer to the true mean over time.

## 5. Visualizing Convergence and Conclusion

After completing all runs, the results are aggregated and analyzed:

- **Visual Comparison:**

Histograms are used to display the distribution of the final estimated mean values from both the online and offline methods.

- **Variance Measurement:**

Squared deviations (a measure of how much the estimates vary) are calculated for each method.

**The overall findings indicate that the online method (which processes data sequentially in mini batches) tends to have lower variance in its estimated mean. This consistency holds true whether the underlying dataset is fixed within each run or freshly generated for every run.**

*The entire analysis can be found in Experiment 2 file.*

## EXPERIMENT 3: Differentially Private ONLINE Stochastic Gradient Descent: Analysis of Convergence

### 1. Data Generation and Objective

This experiment is an extension of Experiment 1, this time in a private setting.

The objective of this experiment is to estimate the mean of the dataset. The algorithm works on the generated data to iteratively update and refine its estimate of this mean. We first create a large set of data points (10,000 values) drawn from a normal distribution with a mean of 0 and a standard deviation of 1. This simulated dataset serves as the source from which mini batches are extracted for the online learning experiments.

### 2. Initialization and Parameter Setup

Before any processing starts, several key values and settings are defined:

- A fixed small step size (learning rate) is set, which controls how big each update will be.
- The batch size is chosen, meaning how many data points are used at one time.
- The total number of iterations is determined, which is the number of times SGD will update the estimated mean.
- Starting estimates for the mean are initialized with the same value (one for the online method and one for the offline method).

These settings ensure the algorithm has the same starting point and a consistent framework for learning.

### 3. Mini-Batch Sampling and Sample Selection Methods

The experiment compares three variants of online learning, each processing mini batches of the fixed dataset:

- **Original Online Learning (Regular SGD):**

The mini batches are selected sequentially from the dataset. For each batch, the algorithm computes the average difference (or error) between the current estimate and the batch's mean. This difference serves as the gradient used to update the estimate.

- **Online Learning with Clipping:**

In this variant, rather than computing a single gradient for the entire batch, gradients are calculated for each data point (as the difference between the current estimate and each individual data point). Each of these individual gradients is then "clipped" so that its absolute value does not exceed the threshold (1). The clipped gradients are averaged, and this more conservative gradient is used to update the estimate.

- **Online Learning with Clipping + DP Noise (DP-SGD):**

Building on the clipped version, this variant further adds a noise component to the averaged gradient. The noise is drawn from a Gaussian distribution calibrated by the privacy parameters (epsilon and delta) and the clipping threshold. This step is designed to protect privacy, but it introduces randomness that can affect the speed of convergence.

## 4. The Update Process

For each iteration and each method, the following steps are executed:

- **Original SGD:**
  - A mini batch is used.
  - The error is calculated as the difference between the current mean estimate and the average of the mini batch.
  - The estimate is updated by taking a step (scaled by the learning rate) in the direction that reduces this error.
- **Clipping:**
  - Individual errors (gradients) are computed for each data point in the batch.
  - These gradients are clipped to restrict their magnitude.
  - The clipped gradients are averaged, and the mean estimate is updated using this value.
  - Because the gradients are restricted, the updates are less aggressive, potentially causing slower convergence.
- **DP-SGD:**
  - Similar to the clipping method, but after averaging the clipped gradients, additional Gaussian noise is added.
  - This noisy gradient is then used to update the mean estimate.
  - The added noise makes the convergence rate less predictable (uncertain speed) compared to the other methods.

## 5. Visualizing the Convergence and Conclusion

- **Plotting the Results:**

The code creates a plot that compares the evolution of the mean estimate over 100 iterations for all three methods:

- **Black Line (Original):** Represents the standard online learning without any modifications.
- **Blue Line (Clipping):** Shows the performance when gradient clipping is applied.
- **Red Line (Clipping + DP):** Reflects the variant with both clipping and the added DP noise.

The clipping variant (blue) converges more slowly than the regular SGD (black) because limiting the gradients makes the updates smaller and less responsive to the actual error. The DP-SGD variant (red) exhibits an uncertain speed of convergence due to the randomness introduced by the noise. This variability makes it less predictable how quickly the mean estimate will approach the true value.

*The entire analysis can be found in Experiment 3 file.*

## EXPERIMENT 4: Differentially Private ONLINE Stochastic Gradient Descent: Analysis of the Empirical Distribution of the Estimator

### 1. Data Generation and Objective

This experiment is an extension of Experiment 3. In a nutshell, we repeat the latter many times and study empirical distribution of the obtained estimators.

The objective of this experiment is to estimate the mean of the dataset. The algorithm works on the generated data to iteratively update and refine its estimate of this mean.

We first create a large set of data points (10,000 values) drawn from a normal distribution with a mean of 0 and a standard deviation of 1. This simulated dataset serves as the source from which mini batches are extracted for the online learning experiments.

### 2. Initialization and Parameter Setup

Before any processing starts, several key values and settings are defined:

- A fixed small step size (learning rate) is set, which controls how big each update will be.
- The batch size is chosen, meaning how many data points are used at one time.
- The total number of iterations is determined, which is the number of times SGD will update the estimated mean.
- Starting estimates for the mean are initialized with the same value (one for the online method and one for the offline method).

These settings ensure the algorithm has the same starting point and a consistent framework for learning.

### 3. Simulation Setup: Multiple Runs

- **Repetition Across Runs:**

The experiment repeats the process for 1000 independent runs. For each run:

- A new dataset is generated.
- Two methods are applied to update the estimated mean:
  - **Variant 1 (Clipping Only):** Applies gradient clipping without adding noise.
  - **Variant 2 (Clipping + DP Noise):** Applies the same clipping but adds Gaussian noise to ensure differential privacy.

- The final estimated mean from each run is stored for both methods.

#### 4. Mini-Batch Sampling and Sample Selection Methods

- **Processing Mini Batches:**

In each run, the dataset is processed in sequential mini batches. For each batch:

- **Clipping Only Variant:**
  - Compute per-example gradients as the difference between the current mean estimate and each data point in the batch.
  - Clip these gradients so that their absolute values do not exceed the predefined threshold.
  - Average the clipped gradients and update the mean estimate using the learning rate.
- **Clipping + DP Noise Variant:**
  - Compute and clip the gradients in the same way.
  - Average the clipped gradients.
  - Add Gaussian noise (with standard deviation based on  $\sigma$ , the clipping threshold, and batch size) to the averaged gradient.
  - Update the mean estimate using this noisy gradient.

#### 5. The Update Process

- **Iterative Updates:**

For each iteration within a run:

- A mini batch is extracted from the dataset.
- The algorithm calculates how much the current estimate deviates from the mini batch data.
- **Without Noise (Clipping Only):**  
The update is solely based on the average of the clipped gradients.
- **With DP Noise:**  
The update includes an additional random noise component, ensuring differential privacy but also introducing extra randomness.

- **Final Recording:**

After processing all mini batches, the final estimated mean is recorded for each run and each method.

#### 6. Visualizing the Convergence and Conclusion

- **Visualization:**

Histograms are plotted to show the distribution of the final estimated means across 1000 runs for both methods.

- **Variance Calculation:**

The squared deviations from the estimated mean (which is around 0 for the normal distribution used) are calculated for both methods.

- **Conclusion:**

The results clearly indicate that:

- **Clipping Only:** The distribution of the final estimated mean is relatively concentrated, meaning the estimates have lower variance.
- **Clipping + DP Noise:** The addition of differential privacy noise introduces extra randomness, leading to a wider spread (higher variance) in the final estimated means.

**In summary, adding DP noise causes more variance in the estimated mean compared to using clipping alone.**

*The entire analysis can be found in Experiment 4 file.*

## EXPERIMENT 5: Differentially Private ONLINE Stochastic Gradient Descent vs. Gaussian Mechanism: Analysis of the Empirical Distribution of the Estimator

### 1. Data Generation

- **Simulated Data:**

The code begins by generating a dataset of 10,000 data points from a normal distribution with mean 0 and standard deviation 1. This simulated data represents the values from which the mean will be estimated.

- **Sensitivity Calculation:**

Sensitivity is computed as the difference between the maximum and minimum values of the dataset divided by the total number of data points. This sensitivity is needed to calibrate the amount of noise for the direct Gaussian mechanism.

### 2. Initialization and Parameter Setup

- **Privacy Parameters:**

- **Privacy Budget ( $\epsilon$ ):** Set to 1.
- **Failure Probability ( $\delta$ ):** Set to  $1e-5$ .

- **Batch and Iteration Parameters:**

- **Batch Size:** 10 data points are processed in each iteration.
- **Number of Iterations:** The process runs for 1,000 iterations.

- **Noise Calibration:**

- **Direct Gaussian Mechanism ( $\sigma_{\text{direct}}$ ):**

Computed using the sensitivity and privacy parameters. This value determines the standard deviation of the noise added directly to the true mean.

- **DP-SGD with Gaussian Noise ( $\sigma_{\text{dp}}$ ):**

Calculated from the privacy parameters, this noise level is used in the iterative DP-SGD updates. It is applied to each mini-batch update after clipping.

- **True Mean Calculation:**

- The true mean of the entire dataset is computed. This serves as a baseline for the direct Gaussian mechanism, where noise is added to the true mean.

### 3. Mini-Batch Sampling and Sample Selection Methods

- **Iterative Processing in Mini Batches:**

The code loops through 1,000 iterations. In each iteration:

- A mini batch of 10 consecutive data points is extracted from the dataset.
- For the DP-SGD estimator:

- **Gradient Calculation:**

The per-example gradient is computed as the difference between the current DP-SGD estimate ( $\mu_{dp}$ ) and each data point in the batch.

- **Gradient Clipping:**

Each gradient is clipped to ensure its absolute value does not exceed a specified threshold ( $\text{clip\_threshold}$ ). This prevents any single data point from having too much influence.

- **Averaging and Noise Addition:**

The clipped gradients are averaged to produce a single gradient value. Then, Gaussian noise (scaled by  $\sigma_{dp}$ , the clip threshold, and the batch size) is added to this average to obtain a differentially private (DP) gradient.

- **Parameter Update:**

The DP-SGD estimate is updated by taking a small step (controlled by the learning rate  $\eta$ ) in the direction opposite to the noisy gradient.

- For the Direct Gaussian Mechanism:

- In each iteration, a new estimate is generated by taking the true mean of the entire dataset and adding a single instance of Gaussian noise drawn with standard deviation  $\sigma_{direct}$ .

- **Storing Estimates:** Both the DP-SGD estimates ( $\mu_{dp}$ ) and the noisy direct estimates ( $\text{est\_gauss}$ ) are stored in separate arrays ( $\text{final\_dp}$  and  $\text{final\_gauss}$ , respectively) for each iteration.

#### 4. The Update Process

- **DP-SGD Updates:**

In every iteration, the DP-SGD estimate is refined through the following steps:

1. Extract a mini batch.
2. Compute and clip per-example gradients.
3. Average the clipped gradients.
4. Add Gaussian noise to the averaged gradient.

5. Update the current estimate using the noisy gradient and a fixed learning rate.

- **Direct Gaussian Mechanism:**

Separately, for each iteration, an estimate is obtained by taking the true mean and adding freshly sampled Gaussian noise. This approach does not involve iterative updates.

## Visualizing the Convergence and conclusions

- **Error Analysis:**

After the loop, the squared deviations (mean squared error relative to 0) are calculated for both methods. These values serve as measures of how much each estimator deviates from the expected true mean (0, in expectation).

- **Histogram Plotting:**

The final estimated means from both the direct Gaussian mechanism (in red) and the DP-SGD estimator (in blue) are visualized using overlaid histograms. This visualization allows for a direct comparison of the distributions.

- **Conclusion:**

The results show that:

- The **Direct Gaussian Mechanism** yields a relatively concentrated distribution of estimated means, as the noise is added only once to the true mean.
- The **DP-SGD Estimator** produces a wider, more variable distribution due to the accumulation of noise over 1,000 iterative updates.

**In summary, adding differential privacy noise through the iterative DP-SGD process results in more variance in the estimated mean compared to the Gaussian mechanism.**

*The entire analysis can be found in Experiment 5 file.*

## EXPERIMENT 6: Neural network for linear regression model

### 1. Data Generation and Objective

This experiment aims to compare three different models for predicting  $Y$  from  $X$  in a simple regression problem:

1. A standard linear regression model (OLS) using `elm()` as a benchmark.
2. A neural network using the `nnet` package (pre-built).
3. A manually implemented neural network trained using backpropagation and gradient descent.

The primary objective of this experiment was to create our own simple neural network from scratch to ensure a deeper understanding of how neural networks work. By manually implementing forward propagation, backpropagation, and gradient descent, we gain better insight into each step of the learning process.

Additionally, this experiment allows us to identify potential points where privacy mechanisms can be integrated into the training process. By breaking down the model's internals, we can analyze:

- Where sensitive data is used and stored within the model.
- How information leaks might occur during training or inference.
- Which steps could be modified to add privacy-preserving techniques such as differential privacy.

The other objective is to compare how well neural networks (both pre-built and manual) perform against linear regression in predicting continuous values. Additionally, it helps in understanding:

- How neural networks optimize weights through gradient descent.
- The differences in predictions made by neural networks vs. traditional regression.
- The impact of different training methods on Mean Squared Error (MSE).

### 2. Data Preparation

- The script creates synthetic data for a simple linear regression task:
  - $X$  is randomly sampled from a uniform distribution between -1 and 1.

-Y follows a linear relationship with X.

- The dataset consists of 100 samples and is used for both training and testing.

### 3. Neural Network Using the nnet Package

- A single hidden-layer neural network with 2 hidden neurons is trained using the nnet package.
- The model learns non-linear mappings using gradient-based optimization with 1000 iterations.
- The final weights of the trained model are displayed.
- The MSE (Mean Squared Error) is computed to evaluate performance.

#### Results

- The MSE of the nnet model is 0.0087, indicating a low prediction error.
- The final trained weights show how the model has adjusted to the data.

### 4. Manually Implemented Neural Network

The script builds a fully connected feedforward neural network from scratch, using:

- One hidden layer with 2 neurons.
- Sigmoid activation function.
- Gradient descent for optimization.
- 10,000 training iterations.

#### Key Steps:

1. Initialize random weights and biases for the input and hidden layers.
2. Perform forward propagation:
  - Compute activations for the hidden layer using the sigmoid function.
  - Compute the final output as a weighted sum of the hidden activations.
3. Compute loss using Mean Squared Error (MSE).
4. Perform backpropagation:
  - Compute gradients of the loss function.
  - Update weights and biases using gradient descent.
5. Repeat for 10,000 iterations to minimize loss.

#### Results:

- The MSE of the manually trained network is 0.0088, similar to the nnet model.
- The loss function decreases steadily over training, confirming effective learning.

## 5. Standard Linear Regression Model

- A linear regression model is estimated using the `lm()` function.
- The fitted values ( $\hat{Y}$ ) are extracted.

*Results:*

- The MSE of the linear model is 0.0092, slightly higher than the neural networks.
- The coefficient is 1.9955, close to the true value of 2.

## 6. Comparing Model Performance

- All three models perform similarly, but neural networks can handle non-linearities better.
- The manual neural network behaves very similarly to the nnet model, confirming that the custom training process is effective.
- The linear regression model follows a strict linear trend but may not be as flexible in capturing non-linear patterns.

## 7. Making New Predictions

- The script makes predictions for new unseen values  $X=[1,0.5,0.3]$ . The predictions are very similar.

## 8. Conclusions

- **The neural networks and linear regression produce nearly identical predictions.**
- **Both neural networks perform well even though one is manually trained.**
- **The linear regression model provides a simple, interpretable baseline.**

*The entire analysis can be found in Experiment 6 file.*

## EXPERIMENT 7: Neural network for linear regression model – impact of noise addition

### 1. Data Generation and Objective

The main objective of this experiment is to analyze how adding noise to the training process of a neural network affects its weight distributions and predictions. By introducing noise into the gradient descent updates, we aim to:

- Understand how noise affects neural network training and its final performance.
- Compare weight distributions between networks trained with and without noise.
- Analyze the variability of predictions when using a noisy vs. non-noisy training process.
- Investigate the potential of noise injection as a privacy-enhancing technique, by making it harder for attackers to extract memorized data from model weights.

### 2. Data Preparation

- The script generates synthetic data for a simple linear regression task:
  - X is randomly sampled from a uniform distribution between -1 and 1.
  - Y follows the linear relationship with X.
- The dataset consists of 100 samples, and all data is used for training

### 3. Neural Network Structure

- The neural network has:
  - One hidden layer with two neurons.
  - Sigmoid activation function for non-linearity.
  - Gradient Descent for optimization.

### 4. Training the Neural Network

Two versions of the neural network are trained:

#### A. Training Without Noise

- The network is trained using standard gradient descent:
  - Weights are updated without any additional noise.

- The process is repeated 1000 times, each time initializing new random weights.

## B. Training with Noise

- The network is trained using noisy gradient descent:
  - Small Gaussian noise is added to the weight.
  - This technique helps obfuscate weight updates, potentially reducing model memorization and improving privacy.
  - The process is repeated 1000 times, each time initializing new random weights.

## 5. Extracting and Comparing Weight Distributions

- After training, the script extracts weight and bias distributions from all trained models:
  - Weights from the first hidden neuron
  - Weights from the second hidden neuron
  - Weights from the hidden layer to the output
  - Biases for hidden and output layers
- Histograms of weight distributions are plotted to compare models trained with vs. without noise.

### Observations:

- Without noise:
  - The weights tend to converge within a narrow range, forming a sharper peak in histograms.
- With noise:
  - The weights are more widely spread, indicating that the noise disrupts exact convergence.
  - The wider distribution suggests reduced overfitting, as weights do not converge to overly specific values.

## 6. Comparing Model Predictions for New Data and Conclusion

- The trained models are used to predict  $Y$  for  $X=1$
- Predictions from 1000 trained models are stored and analyzed.
- Histograms of predictions are plotted for both noisy and non-noisy models.

## Observations

- Without noise:
  - Predictions have less variance, meaning the models are more consistent.
  - The predictions range from 1.086 to 2.543, with a median around 1.869.
- With noise:
  - Predictions have greater variance, making the model more uncertain.
  - The predictions range from 0.1923 to 3.2116, with a median around 1.7884.

**This confirms that noise introduces randomness into the learning process, which can help prevent overfitting, but may reduce accuracy.**

*The entire analysis can be found in Experiment 7 file.*

## EXPERIMENT 8: Building a Simple Transformer-Based Language Model

### 1. Data Generation and Objective

The goal of this experiment is to build a small Transformer-based language model that predicts the next word in a given sentence. The key aspects of this experiment include:

1. Converting words into numerical representations using one-hot encoding and tokenization.
2. Preparing training data from short sentences to learn word relationships.
3. Building a Transformer architecture with self-attention layers.
4. Training the model using stochastic gradient descent (SGD) and mini-batch gradient descent.
5. Evaluating predictions and analyzing probability distributions for next-word predictions.

This experiment demonstrates how modern neural networks process language while highlighting key concepts in natural language processing (NLP) and privacy considerations.

### 2. Defining the Vocabulary and Encoding Words

- A small vocabulary of 10 words is used, including common words like "I", "like", "math", "probability", and some negative sentiment words like "dislike", "football", and "volleyball".
- Each word is mapped to a unique index, allowing it to be represented as a one-hot encoded vector.
- A padding token ("PAD") is included to ensure sequences are of uniform length.

#### *Why One-Hot Encoding?*

One-hot encoding allows words to be converted into numerical representations without assigning any inherent meaning or relationships between words. This serves as the input format for the neural network.

### 3. Preparing Training Data

- 23 short sentences are created using the vocabulary.
- Each sentence is converted into a sequence of tokens, where:
  - The input is all words except the last word.
  - The target is all words except the first word.
- If a sentence is too short, it is padded with the "PAD" token to maintain consistency.

Example Training Pair:  
For the sentence "I like math":

- Input: "I like"
- Target: "like math"
- This helps the model learn word sequences and predict the next word.

#### **4. Model Architecture: Transformer-Based Language Model**

Embedding Layer

- Each word is mapped to an embedding vector instead of a one-hot representation.
- Positional embeddings are added to capture word order information.

Multi-Head Self-Attention Mechanism

- The model uses two self-attention layers, each with three attention heads.
- Each attention head has three matrices:
  - Query (Q): Determines which words to focus on.
  - Key (K): Provides word importance information.
  - Value (V): Holds word representations.
- The attention mechanism helps the model understand relationships between words.

Feed-Forward Layer

- A two-layer feed-forward neural network processes attention outputs, improving representation learning.

Final Output Layer

- The last Transformer output is converted into vocabulary probabilities.
- The softmax function is applied to obtain word prediction probabilities.

#### **5. Training the Transformer Model**

Gradient Descent Techniques

Two different training methods are tested:

1. Stochastic Gradient Descent (SGD)
  - Processes one sentence at a time.
  - Updates model weights after each sentence.

- Suitable for small datasets but can be unstable.
- 2. Mini-Batch Gradient Descent
  - Processes multiple sentences in batches (size = 5).
  - Averages weight updates across the batch.
  - More stable but requires more memory.

### Training Process

- The model runs for 5000 training epochs.
- Loss function: Cross-entropy loss measures the difference between predicted and actual word probabilities.
- Learning rate: Adjusted to ensure smooth convergence.

### Loss Reduction Over Time

- As training progresses, loss decreases, indicating the model is learning word relationships.

## 6. Evaluating Model Predictions

After training, the model is tested on three example prompts:

### Prediction 1: "I like"

The model predicts the probability of the next word:

| Word          | Probability (%) |
|---------------|-----------------|
| "math"        | 27.9%           |
| "probability" | 29.1%           |
| "football"    | 18.1%           |
| "statistics"  | 7.8%            |
| "and"         | 1.9%            |
| "volleyball"  | 1.2%            |

### Observation:

- The model correctly assigns high probability to "math" and "probability", showing it learned from training data.
- However, it also assigns a decent probability to "football", which was part of negative examples.

### Prediction 2: "I dislike"

| Word         | Probability (%) |
|--------------|-----------------|
| "volleyball" | 68.7%           |
| "statistics" | 14.9%           |
| "math"       | 6.6%            |

| Word          | Probability (%) |
|---------------|-----------------|
| "probability" | 5.3%            |
| "and"         | 3.1%            |

**Observation:**

- The model correctly assigns high probability to "volleyball", as it frequently followed "dislike" in training.
- It also predicts "statistics", which is incorrect but might be a result of training bias.

**Prediction 3: "statistics"**

| Word          | Probability (%) |
|---------------|-----------------|
| "and"         | 4.9%            |
| "statistics"  | 0.3%            |
| "math"        | 0.07%           |
| "probability" | 0.06%           |

**Observation:**

- "and" is the most probable next word, which is consistent with training data.
- The model has low confidence for predicting "statistics" again, indicating some understanding of sentence structure.

*The entire analysis can be found in Experiment 8 file.*

## EXPERIMENT 9: Simple example of finding the most common word from certain category associated with any given word

### 1. Problem Definition and Data Preparation

The dataset consists of several sentences describing the color of the sky. The task is to determine which color is most commonly associated with the sky based on word occurrences.

#### Data Preprocessing

- The text is cleaned by removing punctuation.
- Words are converted to lowercase for uniformity.

### 2. Monte Carlo Simulation Approach

The program applies word frequency analysis to estimate the probability of each color appearing in the context of the sky.

#### Color Probability Calculation

For each sentence, the script:

1. Splits it into individual words.
2. Identifies occurrences of user-defined colors (e.g., "blue", "green", "gray", "red").
3. Counts how often each color appears.
4. Computes probabilities based on occurrences:

$P(\text{color}) = \text{count of color occurrences} / \text{total occurrences of all colors}$ .

This step provides a probability distribution over the possible sky colors.

### 3. Results and Observations

#### (a) Probability Distribution

The script calculates and prints the estimated probabilities for each color:

| Color | Probability (%) |
|-------|-----------------|
| Blue  | 60%             |
| Gray  | 20%             |
| Red   | 10%             |

Green 10%

(b) Most Probable Sky Color

- The most frequent color (i.e., the color with the highest probability) is chosen as the predicted sky color.
- In this case, "blue" is the most probable color.

*The entire analysis can be found in Experiment 9 file.*

## EXPERIMENT 10: Simple classification problem

### 1. Understanding the Approach

The experiment implements rule-based text classification, meaning it does not use machine learning or natural language processing models. Instead, it relies on predefined keywords associated with different categories.

For example:

- If a sentence contains words like "R", "programming", or "data", it is classified as "R".
- If it includes "Python" or "coding", it is assigned to "Python".
- Similarly, sentences mentioning "data science" or "machine learning" are categorized accordingly.

Key Steps in the Classification Process:

1. A dataset is manually created with sentences and their assigned categories.
2. A function is defined to check for keyword matches in a given input text.
3. If a keyword is found, the corresponding category is assigned to the input text.
4. If no match is found, the text is classified as "Unknown".

### 2. How the Text Classification Works

When new text is entered, the classification function:

- Iterates through a list of keywords.
- Checks whether any of those keywords appear in the input text.
- If a keyword is detected, the associated category is assigned.
- If no keywords are found, the function returns "Unknown".

This method is simple but limited, as it only looks for exact word matches and does not consider synonyms, context, or partial matches.

### 3. Results and Observations

Correct Classifications

The function performs well in cases where the input text directly contains predefined keywords. For instance:

- "I love programming in R" → Classified as "R".
- "Python is a popular language" → Classified as "Python".
- "Data science is fascinating" → Classified as "Data Science".

In these cases, the function correctly recognizes the keywords and assigns the right class.

## 5. Conclusion

The approach used in this document is a basic keyword-based classification system. While it is simple and works for small datasets, it lacks context awareness, synonym recognition, and learning capability. For better performance, integrating NLP techniques and machine learning would make the classification more accurate and scalable.

This method is useful for quick text classification tasks but would not be suitable for large-scale applications where deep text understanding is required.

*The entire analysis can be found in Experiment 10 file.*

## EXPERIMENT 11: Word prediction based on bigram model

### 1. What is a Bigram Model?

A bigram model is an N-gram-based model used in natural language processing (NLP). It predicts the next word in a sentence by analyzing pairs of words (bigrams) and estimating the likelihood of a word following another.

For example:

- Given the input "data science", the model finds the most frequent word that follows this phrase in the dataset.

This type of model is useful for word prediction, autocomplete functions, and text analysis.

### 2. How the Model Works

Step 1: Preparing the Text Data

- A dataset of sample sentences is compiled as training data.
- The text is converted to lowercase to avoid case mismatches.
- Sentences are combined into a single text corpus to facilitate bigram extraction.

Step 2: Extracting Bigrams

- The model scans through the text and identifies bigrams (two-word sequences).
- Each bigram consists of:
  - First word – context word.
  - Second word – predicted word.

For example:

- In the sentence "I love programming", the model records:
  - Bigram: ("I", "love")
  - Bigram: ("love", "programming")

Step 3: Calculating Bigram Frequencies

- The frequency of each bigram pair is calculated.
- The probability of the second word appearing after the first word is determined.

- This allows the model to rank predictions based on their likelihood.

For example:

- If the phrase "I love" appears 10 times in the dataset:
  - "baking" follows it 4 times (40% probability).
  - "reading" follows it 3 times (30% probability).
  - "coding" follows it 3 times (30% probability).

Thus, "baking" would be the most likely next word for the input "I love".

#### Step 4: Predicting the Next Word

- The model takes the last word of the input phrase.
- It searches for the most probable word that follows it based on bigram statistics.
- It returns the top predictions ranked by probability.

For example:

- Given the input "I enjoy", the model predicts:
  - "baking" (highest probability)
  - "reading" (second highest)
  - "cooking" (third highest)

### 3. Observed Predictions

The model was tested with different input phrases, and the following results were observed:

#### Correct Predictions

1. For input: "I love"
  - Predicted words: "going", "trying", "baking".
  - These predictions are reasonable as "I love" is commonly followed by action words.
2. For input: "Data science"
  - Predicted words: "is", "methods", "analysis".
  - The model correctly identifies that "data science" is often followed by words related to research and analysis.

## 5. Conclusion

The Bigram Model in R is a simple and effective method for word prediction based on word pair frequencies. While it performs well for common phrases, it struggles with unseen words, synonyms, and longer sentence structures.

For better accuracy and scalability, machine learning-based NLP models (such as neural networks) should be considered instead of this purely statistical approach.

*The entire analysis can be found in Experiment 11 file.*

## Part 5: Scope of policies – analysis of legal aspects related to LLM, in the context of the experiments.

Considering the research in the above sections, we identify several open policy issues concerning the privacy implications of LLMs.

### **The Application of PIPEDA**

As a preliminary matter, the application of PIPEDA depends on the context in which the LLM is developed and used. PIPEDA applies to the collection, use and disclosure of personal information “in the course of commercial activity.” If there is no commercial activity (for example, if the LLM is created in a non-commercial research context), PIPEDA will not apply (PIPEDA, s. 4(1)(a)).

In this section, we examine the applicability of PIPEDA to the developers of LLMs, entities creating custom LLM models based on general LLMs, and LLM users. For the purpose of the application of PIPEDA, it is assumed that they are engaged in commercial activity.

#### *LLM Developers*

At the time of writing, the OPC has yet to release its report of findings into an investigation of OpenAI. Among other issues, this investigation will consider the issue of whether OpenAI “has obtained valid and meaningful consent for the collection, use and disclosure of the personal information of individuals based in Canada via ChatGPT.”<sup>37</sup> This investigation is also expected to provide insights into how LLM developers must comply with obligations of openness and transparency, access, accuracy and accountability. Finally, the investigation will also consider the extent to which the collection, use or disclosure of personal information in LLMs are “for purposes that a reasonable person would consider are appropriate in the circumstances” (PIPEDA, s. 5(3)).

However, even in the absence of the OpenAI report of findings, it is possible to draw upon past case law to examine this issue. In 2023, the Federal Court of Appeal confirmed that PIPEDA obligations apply to Google's search function.<sup>38</sup> We observe that, for the same reasoning, PIPEDA obligations are likely to also apply to the output of LLMs insofar as the LLM can be used as a search tool.

---

<sup>37</sup> [https://www.priv.gc.ca/en/opc-news/news-and-announcements/2023/an\\_230525-2/](https://www.priv.gc.ca/en/opc-news/news-and-announcements/2023/an_230525-2/)

<sup>38</sup> Reference re Subsection 18.3(1) of the Federal Courts Act, 2021 FC 723 (CanLII), [2021] 3 FCR 503 [affirmed by Google LLC v. Canada (Privacy Commissioner), 2023 FCA 200]

In *Reference re Subsection 18.3(1) of the Federal Courts Act, 2021 FC 723 (CanLII)*, which was later appealed to the Federal Court of Appeal in *Google LLC v. Canada (Privacy Commissioner)*,<sup>39</sup> the Federal Court answered the two following questions as follows:<sup>40</sup>

1. "Does Google, in the operation of its search engine service, *collect, use or disclose personal information in the course of commercial activities* within the meaning of [paragraph 4\(1\)\(a\)](#) of [PIPEDA](#) when it indexes webpages and presents search results in response to searches of an individual's name?" **Yes**
2. "Is the operation of Google's search engine service excluded from the application of Part 1 of [PIPEDA](#) by virtue of [paragraph 4\(2\)\(c\)](#) of [PIPEDA](#) because it *involves the collection, use or disclosure of personal information for journalistic, artistic or literary purposes and for no other purpose*?" **No**

For the first question, the court concluded that the access, indexing, and retrieval of web pages in the course of constructing Google's index constituted the collection and use of personal information of the subjects of a search.<sup>41</sup> In addition, the disclosure of personal information by Google in response to search queries constituted disclosure.<sup>42</sup> The court also found that, although Google search functions are free to users, they constitute the keystone of a wider for-profit endeavour that constitutes a commercial activity.<sup>43</sup> The same reasoning could be applied to LLMs like ChatGPT, which often offer free-to-use services in addition to paid premium services. Like Google, LLM makers scrape web pages holding, among other things, personal information in order to train LLM models and generate output upon request in the same way that Google gathers web pages and outputs them to search engine users. Furthermore, while some LLMs are free-to-use, it does not vitiate the profit-driven endeavours of companies that make and manage these LLMs. For example, although there is a free version of ChatGPT, it is part of a wider for-profit venture by OpenAI that offers various monetized services using the underlying LLM.

For the second question, the court rejected Google's argument that the search constituted a journalistic activity, and that the purpose of the data collection was to facilitate the indexing and presentation of relevant results. Google had argued that its activities fell under the exception to the consent requirement in PIPEDA s. 4(2)(3). This exception is with respect to information that an organization "collects, uses or discloses for *journalistic, artistic or literary purposes and does not collect, use or disclose for any other purpose*." [Emphasis added]. In the same manner, an argument that LLM makers like OpenAI qualify under the journalistic purposes exception is likely to fail. Even if LLM makers might argue that their tools are used for literary or artistic purposes (i.e. the generation of text or images), the second part of this exception

<sup>39</sup> Note that Google retains the right to appeal to the Supreme Court of Canada and probably will do so, which means that this decision will likely be appealed once again.

<sup>40</sup> *Reference re Subsection 18.3(1) of the Federal Courts Act, 2021 FC 723 (CanLII)*, [2021] 3 FCR 503 [affirmed by *Google LLC v. Canada (Privacy Commissioner)*, 2023 FCA 200]; See *Subjecting Search Engines to PIPEDA | CanLII Connects*.

<sup>41</sup> *Google LLC v. Canada (Privacy Commissioner)* [38-39]

<sup>42</sup> *Google LLC v. Canada (Privacy Commissioner)* [38-39]

<sup>43</sup> *Google LLC v. Canada (Privacy Commissioner)* [38-39]

requires that the collection, use, or disclosure is for *no other purpose* other than the listed exemptions. In addition to the commercial purposes of the data processing as mentioned above, there is also the fact that the collection and use of data by LLM makers is also used to train LLM models. These are “other purposes” that will likely take LLMs outside the scope of this exception to consent. In a similar vein, in the Google reference, the court identified an intermediary purpose (the indexing of webpages) between the crawling of webpages and the disclosure of said webpages to search engine users.<sup>44</sup> It is likely that the intermediary step of training the LLM between the scraping of web data and the output of generated content will also be considered an intermediary purpose. This would disqualify LLM makers from this exception to the consent requirement.

### *LLM Customizers*

This category of entities is engaged in the customization of LLMs, meaning that they are applying general-purpose LLMs such as GPT4 to proprietary information and/or fine-tuning the model to produce specialized sets of output. The customization by these entities is constrained by PIPEDA in two key ways: disclosure restrictions and data breach notification obligations.

First, if these entities are commercial in nature and apply LLMs to proprietary in-house datasets, then it may be necessary to review confidentiality and purpose limitation requirements relating to those datasets. Depending on the terms of the agreement under which any personal information in the datasets was initially collected, it may be the case that the LLM customizer is constrained by confidentiality requirements in their use of the personal information towards LLM customization. For example, PIPEDA requires that consent be obtained for purposes specified at the time of collection. If the use for training an LLM model was not a specified purpose, then there is no valid consent. Further, the collection must be one that a reasonable person would consider appropriate in the circumstances. Unless a “legitimate interests” exception is added to a reformed PIPEDA (as was proposed in the now-defunct Bill C-27), it is difficult to find a basis for consent to use personal information in in-house data sets for the training or refining of AI models. This is, therefore, a preliminary issue that LLM customizers ought to take into account before using such data. It should also be a concern if LLMs are granted access to confidential information, especially where the data could be sent to the original maker of the LLM or accessed by the public by querying the customized LLM.

Furthermore, organizations subject to PIPEDA are also subject to its purpose limitation requirements, which limit the customizer’s use of personal information to the originally specified purposes.<sup>45</sup> Such purposes can be defined broadly, but if the

---

<sup>44</sup> Reference re Subsection 18.3(1) of the Federal Courts Act, 2021 FC 723 (CanLII), [2021] 3 FCR 503 [affirmed by Google LLC v. Canada (Privacy Commissioner), 2023 FCA 200]

<sup>45</sup> [https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/p\\_principle/principles/p\\_use/](https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/p_principle/principles/p_use/)

training of the LLM exceeds the original purpose for which the personal information was collected, the customizer may be required to obtain consent from the parties supplying the personal information. This applies to both customized LLMs that are used purely internally by employees of the LLM-customizing entity (in which case the use may be of a novel purpose) or LLMs that are outward facing and publicly accessible (in which case the use *and disclosure* may be for a novel purpose).

Second, entities may need to make data breach notifications in the event that personal information inputted to LLMs is subsequently accessed by third parties. Under PIPEDA, organizations are required to report any data breaches to the OPC where “it is reasonable in the circumstances to believe that the breach creates a real risk of significant harm to an individual” (s. 10.1(1)). They may also be required to notify individuals whose personal information may have been impacted by the breach.<sup>46</sup> In other words, the incorporation of a customized LLM into an organization's handling of personal data introduces an additional point of potential security breach that could trigger data breach notification obligations.

### *LLM Users*

PIPEDA will not apply to all LLM users. Individuals who use LLMs in a purely personal capacity are not subject to PIPEDA (s. 4(2)(b)). However, PIPEDA will apply to those who use LLMs in the course of commercial activity (s. 4(1)(a)). This category of organizations (which can include individuals engaged in commercial activity) may use LLMs either with the aforementioned customizations or without.

First, organizations will be subject to data protection laws with respect to any personal information they have collected. If they are using LLMs, they must ensure that they have obtained consent to use personal data in the way they are attempting to do so. They must also ensure that the information is appropriately safeguarded when they input it into LLMs. This may mean, for example, that they must ensure that they have contractual provisions in place to ensure that the LLM they are using does not retain data entered by the user nor use it to further train the LLM. In the same way that LLM customizers need to be aware that they are not inputting confidential data into the LLMs that they seek to customize, LLM users need to be aware of their legal obligations regarding data that they use to prompt LLM output.

If LLMs are used by organizations to collect personal information about individuals in the course of commercial activity, they must ensure that they have appropriate consent to do so. Thus, searching for information about individuals using an LLM may be subject to PIPEDA if it is done for a commercial purpose. Further, users that rely on LLM output in processing data and/or making decisions on the basis of personal

---

<sup>46</sup> PIPEDA, s. 10.1(3). [https://www.priv.gc.ca/en/privacy-topics/business-privacy/breaches-and-safeguards/privacy-breaches/respond-to-a-privacy-breach-at-your-business/gd\\_pb\\_201810/](https://www.priv.gc.ca/en/privacy-topics/business-privacy/breaches-and-safeguards/privacy-breaches/respond-to-a-privacy-breach-at-your-business/gd_pb_201810/)

information are subject to accuracy obligations under PIPEDA.<sup>47</sup> Essentially, personal information must be “as accurate, complete, and up-to-date as is necessary for the purposes for which it is to be used.” (Principle 4.6). The risk that LLMs may produce inaccurate information thus impacts this legal obligation.

There is some precedent for identifying data generated about a person as personal information in its own right. For example, in an inquiry by Quebec's Commission d'accès à l'information concerning the use of generated data about students in *Centre de services scolaire du Val-des-Cerfs*, the investigator found that the generation of information about a person by an AI system was *equivalent* to the collection of personal information about that person. Similarly, in the joint report of findings in the Cadillac Fairview investigation, the federal, BC, and Alberta commissioners found that the generation of biometric faceprint data from images of individuals was a collection of personal information that was separate and distinct from the collection of the image.<sup>48</sup> The same idea is found in the *Principles for responsible, trustworthy and privacy-protective generative AI technologies* jointly issued by Canada's federal, provincial and territorial privacy commissioners.<sup>49</sup> Not only is the generation of new information (e.g., inferences) about an individual the collection of personal information for which notice and consent may be required, but the inferences will also be subject to the accuracy obligation.

## The Legality of the Training Data

As canvassed in the previous parts of this report, the source and content of the training data are salient to privacy-related obligations and sanctions. Here, we examine the legal constraints on the collection and application of LLM training data.

### Data Scraping

LLM training data consists mainly of text scraped from the internet. However, there is reason to believe that scraping of publicly accessible personal information may be unlawful under PIPEDA. First, the [Joint investigation of Clearview AI, Inc.](#) by the federal commissioner and his counterparts in Quebec, Alberta and British Columbia, notes that “[i]nformation from sources such as social media or professional profiles, collected from public websites and then used for an unrelated purpose, does not fall under the “publicly available” exception of PIPEDA.”<sup>50</sup> The Regulations Specifying Publicly Available Information define those categories of information that are exempted from the requirements of knowledge and consent because they are

---

<sup>47</sup> PIPEDA, Schedule 1, principle 4.6. [https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/p\\_principle/principles/p\\_accuracy/](https://www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/p_principle/principles/p_accuracy/);  
[https://www.priv.gc.ca/en/opc-news/speeches-and-statements/2023/s-d\\_20230621\\_g7/](https://www.priv.gc.ca/en/opc-news/speeches-and-statements/2023/s-d_20230621_g7/)

<sup>48</sup> <https://www.priv.gc.ca/en/opc-actions-and-decisions/investigations/investigations-into-businesses/2020/pipeda-2020-004/>

<sup>49</sup> [https://www.priv.gc.ca/en/privacy-topics/technology/artificial-intelligence/gd\\_principles\\_ai/](https://www.priv.gc.ca/en/privacy-topics/technology/artificial-intelligence/gd_principles_ai/)

<sup>50</sup> <https://www.priv.gc.ca/en/opc-actions-and-decisions/investigations/investigations-into-businesses/2021/pipeda-2021-001/>

considered “publicly available”.<sup>51</sup> The category of “personal information that appears in a publication, including a magazine, book or newspaper, in printed or electronic form, that is available to the public, where the individual has provided the information” (Regulations, s. 1(e)) has been interpreted to exclude information shared on social media platforms. Second, the OPC has issued a [joint statement](#) as well as a subsequent [concluding joint statement](#) with members from the Global Privacy Assembly’s International Enforcement Cooperation Working Group that emphasized the applicability of data protection and privacy laws to publicly accessible personal information. The joint statements discuss the responsibility of social media platforms that host large quantities of user personal information, suggesting that in some cases, companies who take inadequate steps to prevent data scraping may be considered to have experienced a data breach when scraping occurs. Furthermore, the processing by data scrapers of any resulting datasets may be unlawful. While these statements are not binding in nature, they nevertheless represent the OPC’s incumbent position on data scraping and resulting datasets. Data protection authorities in other jurisdictions have taken similar positions. For example, in early March of this year, the French data protection authority, the *Commission Nationale de l’Informatique et des Libertés* (“CNIL”), fined the chrome extension KASPR for scraping publicly available information from LinkedIn.<sup>52</sup>

The issue of the legality of the training data used to build a system is an interesting one in its own right. In a special report to Parliament issued under the federal Privacy Act, the Privacy Commissioner found that the RCMP could not use a facial recognition database that had been built on improperly sourced data.<sup>53</sup> In their Principles for Responsible, Trustworthy and Privacy-protective Generative AI Technologies, Canada’s privacy commissioners also flag the need to ensure the legality of the underlying data within a system, stating that all parties should “Ensure that where personal information is sourced from third parties, the third parties have collected it lawfully and have authority to disclose it.”<sup>54</sup> This suggests that the use of systems built wholly or in significant part on improperly sourced data may be contrary to data protection law. If this is the case, it could have significant implications for the use of some LLMs. The investigation report in the OpenAI investigation may shed some light on this important issue.

---

<sup>51</sup> <https://www.canlii.org/en/ca/laws/regu/sor-2001-7/latest/sor-2001-7.html>

<sup>52</sup> <https://www.cnil.fr/en/data-scraping-kaspr-fined-eu240000>

<sup>53</sup> [https://www.priv.gc.ca/en/opc-actions-and-decisions/ar\\_index/202021/sr\\_rcmp/](https://www.priv.gc.ca/en/opc-actions-and-decisions/ar_index/202021/sr_rcmp/)

<sup>54</sup> [https://www.priv.gc.ca/en/privacy-topics/technology/artificial-intelligence/gd\\_principles\\_ai/](https://www.priv.gc.ca/en/privacy-topics/technology/artificial-intelligence/gd_principles_ai/)

## Part 6: Summary of activities and related resources.

In Winter 2025, we have delivered a course *Foundations of Data Privacy* at the University of Ottawa. The course was available for both senior undergraduate and graduate students. The course attracted 45 students.

As a part of the project, we prepared a webpage that contain material related to the project:

- 1) Project report;
- 2) R codes for all the experiments conducted;
- 3) Slides for the course *Foundations of Data Privacy*;
- 4) All assignments (with solutions) related to the course *Foundations of Data Privacy*.

The webpage will be further extended to include our future work on data privacy.

## Part 7: Translation of selected parts of the document in French.

Note: this is not 1-1 translation.

### Cadre d'évaluation appliquée des grands modèles de langage

#### Vocabulaire et définitions

**Traitement automatique du langage naturel (TAL)**<sup>55</sup>: permet aux ordinateurs et aux appareils numériques de reconnaître, comprendre et générer du langage écrit et oral.

**Grand modèle de langage (GML)**<sup>56</sup>: un algorithme d'apprentissage profond qui utilise une quantité énorme de paramètres et des données d'entraînement pour comprendre et prédire du texte.

**N-grams**<sup>57</sup>: séquence de n mots.

**Réglage fin**<sup>58</sup>: adapte le modèle préentraîné à des tâches ou des cas d'utilisation spécifiques. Ce processus s'effectue en mettant à jour les paramètres sur une nouvelle base de données.

**Indices contextuels**: dans notre cas, ce sont des indices donnés au GML pour l'aider à prédire le prochain mot.

**Tokenisation**<sup>59</sup>: processus de décomposer un morceau de texte, tel qu'une phrase ou un document, en unités plus petites appelées jetons (tokens).

Exemples de tokenisation:

#### 1. Tokenisation basé sur les espaces

- a. Divise le texte à chaque espace.
- b. Exemple de sortie: [« La », « probabilité », « mesure », « l'incertitude; »

---

<sup>55</sup>Stryker, C., & Holdsworth, J. (2024, August 11). What is NLP (Natural Language Processing)? *Explainers*. <https://www.ibm.com/topics/natural-language-processing>

<sup>56</sup>McDonough, M. (2024, November 7). *Large language model | Definition, History, & Facts*. Encyclopedia Britannica. <https://www.britannica.com/topic/large-language-model>

<sup>57</sup>Jurafsky, D., & Martin, J. H. (2024). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models* (3rd ed.). <https://web.stanford.edu/~jurafsky/slp3>

<sup>58</sup>Shanepeckham. (2024, January 30). *Getting started with LLM fine-tuning*. Microsoft Learn. <https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/working-with-llms/fine-tuning>

<sup>59</sup>LLM Glossary. (n.d.). orq.ai. <https://docs.orq.ai/docs/llm-glossary>

## 2. Tokenisation basé sur les mots

- Sépare les mots de la ponctuation.
- Exemple de sortie : [« La », « probabilité », « mesure », « l », « ' », « incertitude », « ; » ...]

## 3. Tokenisation basé sur les phrases

- Utilise la ponctuation pour diviser le texte en phrases.
- Exemple de sortie: [« La probabilité mesure l'incertitude; », « par exemple » ...]

## 4. Tokenisation base sur les caractères

- Divise les mots en caractères ou groupe de caractères plus fréquents et courants.
- Exemple de sortie : [« La », « probab », « ilité », « mesure », « l » ...]

Plusieurs autres méthodes de tokenisation existent et chaque méthode peut offrir des avantages pour des tâches spécifiques de TAL.

**Jetons (Tokens)**<sup>60</sup>: petits fragments de texte générés en divisant le texte en segments plus petits. Ces segments peuvent être soit des mots ou des groupes de caractères dont la longueur varie d'un seul caractère à un mot entier.

**Vectorisation des mots**: convertis les données initiales de leur format brut (c'est-à-dire le texte) en vecteurs de nombres réels, qui est le format compatible avec les modèles d'apprentissage automatique.

Exemple de vectorisation:

### Encodage one-hot

L'encodage one-hot associe chaque mot à un vecteur dans un espace de haute dimension où la taille du vecteur correspond à la taille du vocabulaire. Un vocabulaire est une collection de mots prédéfinie; un vocabulaire plus large permet une plus grande flexibilité pour traiter différents mots, mais augmente aussi la complexité. Chaque vecteur est majoritairement constitué de zéros avec un '1' à la position correspondant au mot dans le vocabulaire.

Par exemple :

- Supposons que nous ayons un vocabulaire comme celui-ci : [« P », « ( », « événement », « A », « ) » ... « statistique », « analyse », « données »]
- Supposons que nous ayons la séquence de jetons suivante après la tokenisation : [« P », « ( », « événement », « A », « ) »]
- La représentation en encodage one-hot du mot « événement » serait : [0, 0, 1, 0, 0, 0, 0, ...]

<sup>60</sup>Wang, W. W., Andrews, S., & Van Kraay, T. (2024, August 14). *LLM tokens*. Microsoft Learn. <https://learn.microsoft.com/en-us/azure/cosmos-db/gen-ai/tokens>

Remarque : à mesure que la taille du vocabulaire augmente, la dimension des vecteurs devient très élevée, ce qui peut poser des problèmes de complexité.

Une méthode alternative appelée « word embeddings » permet de capturer le sens des mots en les représentant sous la forme de vecteurs dans un espace continu, apportant une solution à certaines limites de l'encodage one-hot.

**Données d'entraînement**<sup>61</sup>: d'énormes quantités de textes (allant de milliards de mots) provenant d'une variété de sources, y compris des sources publiques.

**Apprentissage non supervisé**<sup>62</sup>: type d'apprentissage automatique qui apprend à partir de données sans supervision humaine. Les modèles d'apprentissage automatique non supervisé reçoivent des données non étiquetées et découvrent des tendances et des informations sans instructions explicites.

**Apprentissage supervisé**<sup>63</sup>: utilise des données étiquetées pour entraîner des algorithmes afin de prédire des résultats et identifier des tendances. Cette approche utilise des ensembles de données d'entraînement étiquetés pour enseigner à un modèle un objectif spécifique et prédéterminé.

**Réseau de neurones**<sup>64</sup>: méthode d'intelligence artificielle (IA) qui permet aux ordinateurs de traiter les données d'une façon inspirée par le fonctionnement du cerveau humain.

**Paramètre de température**<sup>65</sup>: un paramètre dans un grand modèle de langage (GML) qui contrôle la créativité des réponses générées par le modèle. Plus précisément, il ajuste la distribution de probabilité des mots possibles qui peuvent suivre dans le texte généré par le modèle.

Les différentes valeurs des paramètres de température des GML :

1. Basse température (<1.0) – régler la température à une valeur inférieure à 1 fait en sorte que la sortie du modèle est plus déterministe et répétitive. Les températures basses poussent le modèle à choisir le mot le plus probable plus souvent, réduisant la variabilité des réponses. Cela peut être utile lorsque des réponses plus prévisibles et conservatrices sont nécessaires, mais peut aussi

---

<sup>61</sup>Lareo, X. (n.d.). *Large language models (LLM)*. European Data Protection Supervisor. [https://www.edps.europa.eu/data-protection/technology-monitoring/techsonar/large-language-models-llm\\_en](https://www.edps.europa.eu/data-protection/technology-monitoring/techsonar/large-language-models-llm_en)

<sup>62</sup>What is unsupervised learning? | Google Cloud. (n.d.). Google Cloud. <https://cloud.google.com/discover/what-is-unsupervised-learning>

<sup>63</sup>What is Supervised Learning? | Google Cloud. (n.d.). Google Cloud. <https://cloud.google.com/discover/what-is-supervised-learning>

<sup>64</sup>What is a Neural Network? - Artificial Neural Network Explained - AWS. (n.d.). Amazon Web Services, Inc. <https://aws.amazon.com/what-is/neural-network/#:~:text=A%20neural%20network%20is%20a,inspired%20by%20the%20human%20brain>

<sup>65</sup> <https://www.iguazio.com/glossary/llm-temperature/>

également rendre le texte moins créatif ou varié, donnant parfois une impression plus robotique.

2. Température élevée (>1.0) – régler la température à une valeur supérieure à 1 augmente le hasard dans le texte généré. Le modèle est plus susceptible de sélectionner des mots moins probables comme mot suivant dans la séquence. Ceci peut mener à des réponses plus créatives et variées. Par contre, ceci peut également causer plus d'erreurs ou de réponses incohérentes puisque le modèle est moins restreint par la probabilité de distribution de ses données d'entraînement.
3. Température de 1.0 – c'est souvent le réglage par défaut, visant un équilibre entre le hasard et le déterminisme. Le modèle génère du texte qui n'est ni trop prévisible ni trop aléatoire en se basant sur la définition de distribution de probabilité apprise lors de son entraînement.

**Hallucinations**<sup>66</sup>: réponses données avec confiance qui semblent crédibles, mais qui sont, en réalité, absurdes lorsqu'elles sont confrontées aux connaissances générales dans le domaine.

#### **Apprentissage profond**<sup>67</sup>:

1. Utilise des réseaux de neurones artificiels pour apprendre à partir des données. Les réseaux de neurones artificiels sont inspirés par le cerveau humain et peuvent être utilisés pour résoudre une variété de problèmes, y compris la reconnaissance d'images, le traitement du langage naturel, et la reconnaissance vocale.
2. Un sous-domaine de l'apprentissage automatique qui utilise des réseaux de neurones avec plusieurs couches, appelés réseaux de neurones profonds, pour simuler la capacité de prise de décisions complexe du cerveau humain.

**Requête**: une demande d'information calculée automatiquement à partir d'un ensemble de données.

**Base de données**:<sup>68</sup> une collection de données stockées.

**Personnes concernées**:<sup>69</sup> personnes auxquelles les données se rapportent.

<sup>66</sup>Alkaissi, H., & McFarlane, S. I. (2023). Artificial Hallucinations in ChatGPT: Implications in Scientific Writing. *Cureus*, 15(2), e35179. <https://doi.org/10.7759/cureus.35179>

<sup>67</sup> <https://cloud.google.com/discover/what-is-deep-learning>; <https://www.ibm.com/topics/deep-learning>

<sup>68</sup>Sparapani, T., & Sherman, J. (2021). *Privacy Tech's Third Generation: A review of the emerging privacy tech sector*. [https://fpf.org/wp-content/uploads/2021/06/FPF-PTA-Report\\_Digital.pdf](https://fpf.org/wp-content/uploads/2021/06/FPF-PTA-Report_Digital.pdf)

<sup>69</sup>Fanti, G., Pihur, V., & Erlingsson, U. (2015). *Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries*. arXiv.org. <http://arxiv.org/abs/1503.01214>

## Aperçu

De nombreux grands modèles de langage (GML) ont été publiés et sont utilisés quotidiennement. Parmi les exemples les plus courants, on trouve : ChatGPT, Gemini et BERT. Bien qu'ils diffèrent par la quantité de paramètres et d'autres détails techniques, ils sont tous basés sur la même idée fondamentale. Ainsi, cette phase a été réalisée afin d'avoir un aperçu général et d'avoir une meilleure compréhension des GML. Nous nous sommes concentrés exclusivement sur ChatGPT, un des GML les plus connus, et avons effectué quelques expériences en l'utilisant ainsi qu'avec le logiciel statistique R.

## Techniques de base et paramètres

**Prédiction de texte**, comme la correction automatique et les suggestions des mots suivants que vous voyez sur vos téléphones ou dans vos courriels, est un exemple simple d'un **modèle de langage**. Le modèle est entraîné sur de grands ensembles de données pour apprendre les tendances statistiques du langage, comme les mots qui suivent généralement d'autres mots. Lorsque vous tapez, le modèle prédit le prochain mot en fonction du contexte de ce que vous avez déjà écrit. Un modèle de base peut utiliser des **n-grams** (séquences de mots) pour faire des prédictions, tandis que les modèles plus avancés, dont ChatGPT, utilisent l'apprentissage profond pour comprendre un contexte plus large. Avec le temps, le modèle peut également personnaliser ses suggestions en fonction de vos habitudes de frappe, améliorant ainsi sa précision.

**La classification de texte** est un processus de catégorisation de texte dans des étiquettes ou catégories prédéfinies. Ce processus utilise des modèles d'apprentissage automatique pour analyser les données textuelles et les attribuer à une ou plusieurs classes en fonction de son contenu. Par exemple, un courriel peut être classé comme étant **non sollicité** ou **sollicité** et une revue de produits peut être classée comme étant **positive** ou **négative**. Le modèle est entraîné à partir d'exemples étiquetés (comme des courriels non sollicités ou sollicités) pour apprendre les caractéristiques qui distinguent les différentes catégories. Une fois entraîné, le modèle peut automatiquement classer de nouveaux textes, jamais vus auparavant, basés sur ces caractéristiques. La classification de texte est largement utilisée dans des applications telles que l'analyse de sentiments, la catégorisation des sujets et le filtrage des courriels.

Des exemples simples de ces deux techniques de bases pour les GML sont présentés dans la section suivante.

## Exemples simples de grands modèles de langage

Nous avons mené quelques expériences afin de mieux comprendre le fonctionnement des GML. À la base, un GML prédit le mot suivant en fonction de la probabilité qui lui est attribuée. Cette probabilité est calculée pendant la phase d'entraînement. Pour ces expériences, nous avons utilisé deux techniques différentes : la première utilise les n-grams et la deuxième utilise les indices contextuels. Avant d'entrer dans les détails des expériences, il faudrait noter que celles-ci ont été réalisées à petite échelle. Bien que la question de la capacité d'être mis à l'échelle de ces expériences n'ait pas été explorée, nous supposons que les exemples suivants ne peuvent pas être adaptés à des ensembles de données très volumineux.

### Exemple bi-gram:

Cet exemple a utilisé les n-grams de taille  $n=2$ . En termes plus simples, le modèle a pris les phrases des données d'entraînement, les a compilées en une grande phrase et ensuite en plusieurs petites séquences de deux mots.

Pour réaliser l'expérience, nous avons créé un ensemble de données textuelles (en anglais). Cet ensemble de données contenait les phrases suivantes :

- I love math
- I love physics
- Programming is not fun
- Data science is amazing
- I enjoy old Polish movies

La prochaine étape était de combiner les phrases pour former une seule grande phrase tout en supprimant les majuscules. Par exemple, une partie du nouvel ensemble de données ressemblait à : « i love math i love physics » et ainsi de suite. Nous avons ensuite créé une fonction qui générait des bi-grams et calculait leur fréquence. Ainsi, le résultat était une liste de séquences de deux mots accompagnées de leurs fréquences. Ces fréquences ont été utilisées pour prédire le mot suivant dans une séquence. Par exemple, la séquence « i love » avait une probabilité 66,7% et la séquence « i enjoy » avait une probabilité de 33,3%. Ainsi, en utilisant une fonction que nous avons créée pour prédire le mot suivant, si nous mettons « i » dans le modèle, il renverra les mots « love » et « enjoy » ainsi que leurs probabilités respectives. Plus la probabilité est élevée, plus il est probable que le GML choisisse ce mot comme le suivant dans la séquence.

### Exemple bi-gram avec paramètre de température :

Prenons maintenant le même exemple que ci-dessus. Notez qu'il y a trois façons de base pour obtenir le résultat final (prédire le prochain mot) basé sur des probabilités calculées à partir de l'ensemble de données d'entraînement :

#### 1. **Cas déterministe (sans hasard):**

Dans ce cas, le modèle choisit le mot suivant en fonction de la probabilité la plus élevée (le mot le plus courant dans notre ensemble de données d'entraînement pour le mot d'entrée donné). Par exemple, avec l'entrée « i », le mot suivant serait « love » comme il a la probabilité la plus élevée. Ainsi, il n'y a pas de créativité avec cette approche.

## 2. **Hasard basé sur les probabilités originales:**

Ici, le modèle choisit le mot suivant parmi les résultats possibles selon leurs probabilités calculées. Par exemple, avec l'entrée « i », sur 100 itérations, nous pourrions obtenir le mot « love » environ 67 fois et « enjoy » environ 33 fois. Dans ce cas, le modèle est plus créatif.

## 3. **Paramètre de température:**

Nous pouvons modifier les probabilités calculées en utilisant une fonction (puis mettre à l'échelle les valeurs pour qu'elles s'additionnent à 1) et choisir le mot suivant parmi les résultats possibles selon cette nouvelle distribution. Par exemple, nous pouvons appliquer la fonction  $f(x)=x^{(1/T)}$ , où  $x$  est la probabilité originale et  $T$  est le paramètre de température.

- Pour  $T=1$ , les probabilités demeurent pareilles (comme dans le deuxième cas)
- Pour  $T>1$ , les probabilités deviennent plus égales, ce qui signifie que le modèle a plus de hasard le plus que  $T$  augmente.
- Pour  $T<1$ , le modèle devient plus déterministe, favorisant le mot avec la probabilité originale la plus élevée à mesure que  $T$  se rapproche de 0.

### Indices contextuels:

Le but de cet exemple était de prédire le mot suivant dans une phrase basé sur le format : « This morning/evening I want to eat/drink ». Nous avons commencé par créer un ensemble de données contenant des indices contextuels pour le matin et le soir. Par exemple, l'ensemble de données pour les indices contextuels du matin comprenait :

- Morning
- Wake up
- Waking up
- Wake
- Get up
- Breakfast

L'ensemble des données pour les indices contextuels du soir comprenait des mots du même style que ceux du matin.

Pour garantir la simplicité, nous avons attribué les probabilités aux mots « cereal », « salad », « milk » et « beer » au lieu de créer un ensemble de données et d'utiliser des bi-grams comme dans l'exemple précédent. Ces probabilités étaient basées sur notre jugement concernant la probabilité qu'une personne dise qu'elle mangerait ou boirait un de ces mots le matin ou le soir. Par exemple, nous avons attribué une probabilité de 0.6 au mot « cereal » et 0.3 pour le mot « salad » dans le contexte du matin.

Enfin, nous avons créé une fonction qui demande à l'utilisateur de rédiger une phrase. La fonction regardera au dernier mot ainsi que les indices contextuels dans la phrase. En fonction du dernier mot et les indices contextuels, le modèle répondra avec la phrase donnée par l'utilisateur et soit le mot « cereal », « salad », « milk » ou « beer ».

Prenons un exemple plus concret :

1. Lors de l'exécution du modèle, il demande: « Please write a sentence : »
2. Nous écrivons: « Waking up this morning, I want to eat »
3. Le modèle répond avec: « Waking up this morning I want to eat cereal »

### Problème de classification de texte simple

Notre exemple illustre une approche simple à la classification de texte en utilisant une liste prédéfinie de mots-clés pour chaque catégorie. L'objectif est de classer un texte donné dans une ou plusieurs catégories prédéfinies, comme « R », « Python », « Data Science » ou « Machine Learning », en fonction de la présence de certains mots-clés. Nous créons un ensemble de données simple sous forme de liste de phrases associées à une catégorie (par exemple : « R », « Python », « Data Science » ou « Machine Learning »). Ensuite, pour chaque catégorie, un ensemble de mots-clés est défini. En utilisant des fonctions simples, nous analysons chaque phrase et cherchons pour des mots dans ces catégories. Si le modèle en trouve, il classe la phrase en utilisant l'étiquette appropriée.

Par exemple :

Pour « I want to learn data science », la catégorie prédite serait « Data Science ».

Pour « What about R programming? » la catégorie prédite serait « R ».

Pour « Machine learning is the future », la catégorie prédite serait « Machine Learning ».

## Problèmes courants avec ChatGPT

Hallucinations – dans ce contexte, les hallucinations désignent la génération d'informations fausses ou trompeuses. Cela inclut des informations inventées, de fausses nouvelles ou des références fictives, souvent présentées comme crédibles.

Voici quelques exemples de problèmes signalés dans différents domaines :

1. Médecine<sup>70</sup>:

- « Dans son état actuel de développement, les médecins et chercheurs dans le domaine de la biochimie ne devraient PAS demander à ChatGPT des sources, références ou citations sur un sujet particulier. Ou, s'ils le font, toutes les références doivent être soigneusement vérifiées pour garantir l'exactitude. »

2. Santé<sup>71</sup>:

- « Ce phénomène a été nommé hallucinations. C'est un abus de langage. Les hallucinations sont des fausses perceptions. Ce que j'ai observé ressemble plutôt à des fabrications ou des falsifications. Le « Office of Research Integrity of the United States Department of Health and Human Services » définit la fabrication comme l'invention de données ou de résultats et la falsification comme la manipulation, le changement ou l'omission de données ou de résultats afin que la recherche soit mal représentée. <sup>72</sup>»

3. Droit<sup>73</sup>:

- « Un juge a sanctionné des avocats pour un mémoire rédigé par une IA contenant de fausses citations. »

## Exemples d'hallucinations dans les GML/ChatGPT

Afin de mieux comprendre les hallucinations dans les GML, voici quelques exemples simulés pour mieux comprendre ce phénomène.

### Exemple de base: « hallucination » d'un nouveau mot<sup>74</sup>

Pour illustrer les hallucinations, nous pouvons créer un bi-gram de deux lettres à partir d'un texte. Cela consiste à créer une table qui contient toutes les paires de deux lettres présentes dans le texte et à calculer leur fréquence.

Par exemple, prenons la phrase : « Hallucinations in large language models ». Ce texte permet de générer les bi-grams suivants: « HA », « AL », « LL », « LU », et ainsi de suite. Dans cet exemple, « LU » apparaît une seule fois et « LA » apparaît deux fois.

En utilisant la lettre « L » comme point de départ, le modèle a deux fois plus de chance de produire « LA » que « LL » ou « LS ». Si nous poursuivons avec « LA », nous pourrions obtenir « AL », « AT », « AR », ou « AN », selon les probabilités des

<sup>70</sup> Goddard, J. (2023). Hallucinations in ChatGPT: A cautionary tale for biomedical researchers. The American Journal of Medicine, 136(11), 1059-1060. <https://doi.org/10.1016/j.amjmed.2023.06.012>

<sup>71</sup> Emsley, R. (2023). ChatGPT: These are not hallucinations – they're fabrications and falsifications. Schizophrenia, 9, 52. <https://doi.org/10.1038/s41537-023-00379-4>

<sup>72</sup> <https://ori.hhs.gov/definition-research-misconduct>

<sup>73</sup> <https://www.cnbc.com/2023/06/22/judge-sanctions-lawyers-whose-ai-written-filing-contained-fake-citations.html>

<sup>74</sup> <https://machinelearningmastery.com/a-gentle-introduction-to-hallucinations-in-large-language-models/>

paires observées. Éventuellement, ce processus peut produire une séquence de lettres qui forment un mot qui n'existe pas, simplement parce que le modèle se base uniquement sur les probabilités des paires de lettres extraites du texte.

À la base, les hallucinations dans les GML ne sont pas plus complexes que l'exemple présenté même si les modèles sont beaucoup plus complexes. Les hallucinations surviennent lorsque le modèle manque de compréhension contextuelle et/ou quand les données d'entraînement contiennent des incohérences ou du bruit, créant des biais dans les réponses.

### ***Création de livres et d'articles fictifs – Version gratuite de ChatGPT***

En utilisant la version gratuite de ChatGPT, une expérience a été menée pour évaluer la capacité de ChatGPT à générer des informations fictives concernant des publications scientifiques.

**Première question:** « *Faites une liste des cinq articles les plus importants sur la théorie des valeurs extrêmes.* »

Réponse de ChatGPT :

1. Pickands, J. (1975). "Statistical Inference Using Extreme Order Statistics."
2. Gumbel, E. J. (1958). "Statistics of Extremes."
3. Coles, S. G. (2001). "An Introduction to Statistical Modeling of Extreme Values."
4. Davison, A. C., & Huser, R. (2015). "Statistics of Extremes." *Annual Review of Statistics and Its Application*, 2(1), 203-235.
5. Ledford, A. W., & Tawn, J. A. (1996). "Statistics for Near-Independent Extreme Values." *Biometrika*, 83(1), 89-100.

En vérifiant les références, il s'est avéré que certaines n'existaient pas, démontrant une hallucination dans la génération.

**Deuxième question:** « *Quels livres ont été écrits par Rafal Kulik ?* »

Réponse de ChatGPT :

1. "Statistical Modelling of Extreme Values" (2021) - Co-authored with \*\*Antoni R. M. Tawn
2. "Statistical Analysis of Extreme Values: Theory and Applications" (2019)- Co-authored with Rafał J. Kulik and Kazimierz K. D. Z.
3. "Extreme Value Theory: An Introduction" (2020)
4. "Statistical Models in S and S-PLUS" (2000) - Co-authored with J. D. H. Smith
5. "Introduction to Extreme Value Theory" (2023)

En vérifiant les références, aucun des livres n'a été écrit par Rafal Kulik et certains n'existaient pas.

**Troisième question:** « *Quels livres ont été écrits par Rafal Kulik et Heidi Barriault?* »

Réponse : ChatGPT a affirmé qu'ils avaient co-écrit deux livres, bien que cela soit faux.

**Dernière question:** « *Faites une liste des livres que Rafal Kulik et Teresa Scassa ont écrits ensemble.* »

Réponse: ChatGPT a inventé un livre intitulé « *Statistical Methods for the Environmental Sciences (2023)* », alors que ces auteurs n'ont jamais collaboré. En effet, les deux personnes sont dans des domaines complètement différents. Nous avons posé cette question, même si elle est très semblable à la troisième, puisque Teresa Scassa a rédigé plusieurs articles et est plus connue que Heidi Barriault, l'étudiante de maîtrise à Rafal Kulik. Ainsi, nous pensions que ChatGPT ne ferait pas de liens entre deux personnes « reconnues ».

### ***Création de livres et d'articles fictifs – version payante de ChatGPT***

En répétant l'expérience avec la version payante de ChatGPT, les réponses étaient complètement différentes.

**Première question:** « *Faites une liste des cinq articles les plus importants sur la théorie des valeurs extrêmes.* »

Réponse de ChatGPT :

1. **Fisher, R. A., & Tippett, L. H. C. (1928)** - *Limiting forms of the frequency distribution of the largest or smallest member of a sample*
2. **Gnedenko, B. V. (1943)** - *Sur la distribution limite du terme maximum d'une série aléatoire*
3. **Leadbetter, M. R., Lindgren, G., & Rootzén, H. (1983)** - *Extremes and Related Properties of Random Sequences and Processes*
4. **de Haan, L., & Ferreira, A. (2006)** - *Extreme Value Theory: An Introduction*
5. **Embrechts, P., Klüppelberg, C., & Mikosch, T. (1997)** - *Modelling Extremal Events for Insurance and Finance*

En vérifiant les références, nous avons trouvé que tous les articles existaient.

**Deuxième question :** « *Quels livres ont été écrits par Rafal Kulik ?* »

Réponse de ChatGPT :

1. **Heavy-Tailed Time Series** (co-authored with Philippe Soulier)
2. **Long-Memory Processes: Probabilistic Properties and Statistical Methods** (co-authored with Jan Beran, Yuanhua Feng, and Sucharita Ghosh)

Ces livres ont été écrits par Rafal Kulik.

**Troisième question :** « *Quels livres ont été écrits par Rafal Kulik et Heidi Barriault?* »

Réponse : ChatGPT a répondu qu'aucune collaboration n'avait été trouvée.

**Dernière question :** « *Faites une liste des livres que Rafal Kulik et Teresa Scassa ont écrits ensemble.* »

Réponse : ChatGPT a affirmé qu'aucune collaboration n'existait entre ces auteurs.

## Exemples de cyberattaques

ChatGPT bloque les requêtes qui demandent explicitement la création de logiciels malveillants ou de menaces d'ingénierie sociale. Cependant, en modifiant la formulation de la requête, il est possible d'obtenir des réponses de ChatGPT qui peuvent contribuer à des cyberattaques. Nous avons mené quelques expériences avec ChatGPT afin d'analyser comment la formulation de la requête influence la réponse. Les expériences sont inspirées par des exemples dans les articles et publications académiques<sup>75 76 77 78 79</sup>.

### Expérience 1: Attaque par hameçonnage

Une attaque par hameçonnage est quand on envoie des courriels, messages textes ou appels frauduleux qui semblent provenir d'une source légitime dans le but d'inciter les victimes à partager des informations personnelles ou des identifiants de connexion<sup>15</sup>.

Cette expérience est basée sur <sup>15</sup>. Nous avons commencé en demandant à ChatGPT de rédiger un courriel d'hameçonnage pour nos employés. ChatGPT a refusé et a répondu que « l'hameçonnage était illégal et immoral ». Ainsi, nous avons décomposé chaque étape de la rédaction d'un courriel d'hameçonnage :

1. Demander pour un exemple d'un code HTML et CSS pour créer une fausse page Facebook. ChatGPT a fourni un code.
2. Demander pour un exemple de code JavaScript qui peut prendre les données d'un formulaire HTML et les envoyer à un « endpoint RESTful ». ChatGPT a fourni un code.
3. Demander une suggestion de courriel informant un client que nous avons besoin de ses identifiants de façon urgente comme il y a eu un problème avec la base de données et de les envoyer en soumettant le formulaire dans le courriel. ChatGPT a écrit un courriel.

Bien que ChatGPT ne fournisse pas directement une méthode pour une attaque par hameçonnage, il peut faciliter la création des différentes étapes, ce qui rend ces actions plus accessibles.

---

<sup>75</sup>Grbic, D. V., & Dujlovic, I. (2023). Social engineering with ChatGPT. *2023 22nd International Symposium INFOTEH-JAHORINA (INFOTEH)*, 1–5. <https://doi.org/10.1109/infoteh57020.2023.10094141>

<sup>76</sup>Lee, C. (2023, August 8). *Unmasking Hypnotized AI: The hidden risks of large*. Security Intelligence. <https://securityintelligence.com/posts/unmasking-hypnotized-ai-hidden-risks-large-language-models/>

<sup>77</sup>Shimony, E., & Tsarfati, O. (2023, January 17). *Chatting our way into creating a polymorphic malware*. Cyberark. <https://www.cyberark.com/resources/threat-research-blog/chatting-our-way-into-creating-a-polymorphic-malware>

<sup>78</sup>Burns, M. (n.d.). Artificial Intelligence & Social Engineering: How Attackers Leverage ChatGPT. *Wolf & Company, P.C.* <https://www.wolfandco.com/resources/blog/artificial-intelligence-social-engineering-attackers-leverage-chatgpt/>

<sup>79</sup>Wu, X., Duan, R., & Ni, J. (2024). Unveiling security, privacy, and ethical concerns of ChatGPT. *Journal of Information and Intelligence*, 2(2), 102–115. <https://doi.org/10.1016/j.jiixd.2023.10.007>

### *Expérience 2: Attaque par hameçonnage ciblé*

Une attaque par hameçonnage ciblé est une forme d'hameçonnage où les messages sont personnalisés en fonction des informations disponibles sur une personne ou organisation<sup>15</sup>.

Nous avons commencé cette expérience en demandant à ChatGPT de nous donner toutes les informations disponibles sur Rafal Kulik, un professeur à l'université d'Ottawa. Ensuite, nous avons demandé à ChatGPT de rédiger un courriel de la part du superviseur de Rafal qui lui dit qu'il y a une mise à jour de mot de passe urgente et d'utiliser le lien dans le courriel pour faire la mise à jour. Ainsi, ChatGPT a généré un courriel personnalisé et convaincant.

### *Expérience 3: Création de logiciels malveillants*

Plusieurs articles et publications académiques ont mentionné que ChatGPT pouvait être utilisé pour créer des logiciels malveillants lorsqu'il est interrogé d'une certaine façon. En suivant les étapes dans ces publications, nous avons formulé des requêtes pour générer des codes malveillants. Ainsi, ChatGPT peut fournir du code pouvant être utilisé à des fins malveillantes.

### Conclusion:

La conclusion principale que nous avons extraite de ces expériences est que ChatGPT peut être utilisé pour faciliter les cyberattaques puisque les informations pour réaliser des cyberattaques sont disponibles en ligne. Cependant, ses informations nécessitent souvent beaucoup de recherche et des connaissances de base. Ainsi, ChatGPT simplifie le processus en fournissant des informations rapidement et de façon cohérente.

## Disparité

Comme mentionné plus tôt, nous avons seulement utilisé ChatGPT pour nos expériences dans cette phase, avec l'exemption de la création de petits modèles de langage avec R. Ainsi, nous ne pouvons pas généraliser nos découvertes avec d'autres grands modèles de langage.

Une disparité que nous avons trouvée est la différence entre la quantité d'hallucinations générées par la version gratuite et payante de ChatGPT. Nous ne savons pas la raison pour laquelle il y a une différence, mais nous pensons que cela pourrait être lié au paramètre de température. Ce paramètre pourrait être réglé à une valeur plus élevée dans la version gratuite, augmentant ainsi les hallucinations.

De plus, nos expériences démontrent qu'en modifiant la formulation des requêtes, il est possible de contourner les règles éthiques et légales de ChatGPT.

## Phase II – Expérimentation et évaluation des mécanismes privés.

**Ci-dessous, nous avons souligné les expériences initiales que nous menons, ainsi qu'une revue littéraire approfondie. Nous consultons aussi avec des experts d'industrie spécialisés dans les grands modèles de langage. Nous allons commencer cette phase en répétant quelques définitions clés qui parviennent du rapport de la phase 1.**

### Vocabulaire et définitions

*Note: Ces définitions viennent du rapport de la phase 1 et seulement celles qui sont pertinentes pour cette phase ont été incluses.*

**Traitement automatique du langage naturel (TAL)<sup>80</sup>:** permet aux ordinateurs et aux appareils numériques de reconnaître, comprendre et générer du langage écrit et parler.

**Grand modèle de langage (GML)<sup>81</sup>:** un algorithme d'apprentissage profond qui utilise une quantité énorme de paramètres et des données d'entraînement pour comprendre et prédire du texte.

**Réglage fin<sup>82</sup>:** adapte le modèle préentraîné à des tâches ou des cas d'utilisation spécifiques. Ce processus s'effectue en mettant à jour les paramètres sur une nouvelle base de données.

**Indices contextuels:** dans notre cas, ce sont des indices donnés au GML pour l'aider à prédire le prochain mot.

**Tokenisation<sup>83</sup>:** processus de décomposer un morceau de texte, tel qu'une phrase ou un document, en unités plus petites appelées jetons (tokens).

**Données d'entraînement<sup>84</sup>:** d'énormes quantités de textes (allant de milliards de mots) provenant d'une variété de sources, y compris des sources publiques.

---

<sup>80</sup>Stryker, C., & Holdsworth, J. (2024, August 11). What is NLP (Natural Language Processing)? *Explainers*. <https://www.ibm.com/topics/natural-language-processing>

<sup>81</sup>McDonough, M. (2024, November 7). *Large language model | Definition, History, & Facts*. Encyclopedia Britannica. <https://www.britannica.com/topic/large-language-model>

<sup>82</sup>Shanepeckham. (2024, January 30). *Getting started with LLM fine-tuning*. Microsoft Learn. <https://learn.microsoft.com/en-us/ai/playbook/technology-guidance/generative-ai/working-with-llms/fine-tuning>

<sup>83</sup>LLM Glossary. (n.d.). orq.ai. <https://docs.orq.ai/docs/llm-glossary>

<sup>84</sup>Lareo, X. (n.d.). *Large language models (LLM)*. European Data Protection Supervisor. [https://www.edps.europa.eu/data-protection/technology-monitoring/techsonar/large-language-models-llm\\_en](https://www.edps.europa.eu/data-protection/technology-monitoring/techsonar/large-language-models-llm_en)

**Apprentissage non supervisé<sup>85</sup>:** type d'apprentissage automatique qui apprend à partir de données sans supervision humaine. Les modèles d'apprentissage automatique non supervisé reçoivent des données non étiquetées et découvrent des tendances et des informations sans instructions explicites.

**Apprentissage supervisé<sup>86</sup>:** utilise des données étiquetées pour entraîner des algorithmes afin de prédire des résultats et identifier des tendances. Cette approche utilise des ensembles de données d'entraînement étiquetés pour enseigner à un modèle un objectif spécifique et prédéterminé.

**Réseau de neurones<sup>87</sup>:** méthode d'intelligence artificielle (IA) qui permet aux ordinateurs de traiter les données d'une façon inspirée par le fonctionnement du cerveau humain.

**Paramètre de température<sup>88</sup>:** un paramètre dans un grand modèle de langage (GML) qui contrôle la créativité des réponses générées par le modèle. Plus précisément, il ajuste la distribution de probabilité des mots possibles qui peuvent suivre dans le texte généré par le modèle.

**Hallucinations<sup>89</sup>:** réponses données avec confiance qui semblent crédibles, mais qui sont, en réalité, absurdes lorsqu'elles sont confrontées aux connaissances générales dans le domaine.

### **Apprentissage profond<sup>90</sup>:**

3. Utilise des réseaux de neurones artificiels pour apprendre à partir des données. Les réseaux de neurones artificiels sont inspirés par le cerveau humain et peuvent être utilisés pour résoudre une variété de problèmes, y compris la reconnaissance d'images, le traitement du langage naturel, et la reconnaissance vocale.
4. Un sous-domaine de l'apprentissage automatique qui utilise des réseaux de neurones avec plusieurs couches, appelés réseaux de neurones profonds, pour simuler la capacité de prise de décisions complexe du cerveau humain.

**Requête:** une demande d'information calculée automatiquement à partir d'un ensemble de données.

<sup>85</sup>What is unsupervised learning? | Google Cloud. (n.d.). Google Cloud. <https://cloud.google.com/discover/what-is-unsupervised-learning>

<sup>86</sup>What is Supervised Learning? | Google Cloud. (n.d.). Google Cloud. <https://cloud.google.com/discover/what-is-supervised-learning>

<sup>87</sup>What is a Neural Network? - Artificial Neural Network Explained - AWS. (n.d.). Amazon Web Services, Inc. <https://aws.amazon.com/what-is/neural-network/#:~:text=A%20neural%20network%20is%20a,inspired%20by%20the%20human%20brain>

<sup>88</sup> <https://www.iguazio.com/glossary/llm-temperature/>

<sup>89</sup>Alkaissi, H., & McFarlane, S. I. (2023). Artificial Hallucinations in ChatGPT: Implications in Scientific Writing. *Cureus*, 15(2), e35179. <https://doi.org/10.7759/cureus.35179>

<sup>90</sup> <https://cloud.google.com/discover/what-is-deep-learning>; <https://www.ibm.com/topics/deep-learning>

**Base de données:**<sup>91</sup> une collection de données stockées.

**Personnes concernées:**<sup>92</sup> personnes auxquelles les données se rapportent.

## Aperçu (Phase 2)

Au cours de la phase I, nous nous sommes concentrés sur la compréhension de la structure des grands modèles de langage, particulièrement ChatGPT. Nous avons examiné les différents défis associés à la mise en œuvre de ces grands modèles de langage. La deuxième phase est axée sur la **confidentialité dans les grands modèles de langage**. Ainsi, pour ce faire, il est nécessaire de bien comprendre la structure des algorithmes d'apprentissage profond, en particulier les réseaux de neurones. Ces réseaux de neurones sont utilisés pour entraîner et évaluer différents modèles.

Les activités suivantes ont été menées:

- Nous avons décrit mathématiquement la mise en œuvre des réseaux de neurones. **La description mathématique complète sera incluse dans le rapport final;**
- Basés sur la structure mathématique, nous avons codé les réseaux de neurones et les avons comparés aux paquets existants;
- Nous avons examiné les techniques d'anonymisation actuelles pour les algorithmes d'apprentissage profond. Nous avons mené des expériences de simulation **(qui seront incluses dans le rapport final)**. Le codage des réseaux de neurones est essentiel afin d'implémenter des algorithmes de préservation de la confidentialité, puisque les paquets de bases ne permettent pas d'intégrer directement ces mécanismes.
- Nous avons revu la littérature sur les menaces de confidentialité des algorithmes d'apprentissage profond. Un problème qui ressort souvent concerne les droits d'auteur. Bien que ce ne soit pas strictement une question de confidentialité, plusieurs problèmes y sont associés.

Commentaires sur les autres phases :

- Phase 3 – les expériences sont en cours. Nous allons décrire leurs résultats (méthodologie et base de référence pour les garanties de confidentialité dans les GML) dans le rapport final.
- Phase 4 – sera incluse dans le rapport final
- Phase 5 – un cours trimestriel « Foundations of Data Privacy » est en cours à l'Université d'Ottawa. Une page web contenant toutes les ressources créées

---

<sup>91</sup>Sparapani, T., & Sherman, J. (2021). *Privacy Tech's Third Generation: A review of the emerging privacy tech sector*.

[https://fpf.org/wp-content/uploads/2021/06/FPF-PTA-Report\\_Digital.pdf](https://fpf.org/wp-content/uploads/2021/06/FPF-PTA-Report_Digital.pdf)

<sup>92</sup>Fanti, G., Pihur, V., & Erlingsson, Ú. (2015). *Building a RAPPOR with the Unknown: Privacy-Preserving Learning of Associations and Data Dictionaries*. arXiv.org. <http://arxiv.org/abs/1503.01214>

au cours de ce projet est actuellement en construction et sera dévoilée dans le rapport final. Des parties de la phase 1 et 2 ont été traduites en français. Nous organiserons un atelier d'une journée à la fin du projet.

## Introduction

L'utilisation croissante des grands modèles de langage (GML) pose d'importants défis de confidentialité. Cette recherche étudie les risques de confidentialité principaux. Le but est de:

- 1) Identifier les **risques de confidentialité**,
- 2) Évaluer les stratégies actuelles pour **l'atténuation de la confidentialité des données**,
- 3) **explorer de nouvelles méthodes de préservation de la confidentialité**.

## Risques de confidentialité

Les risques les plus communs sont liés à :

- a) La fuite de données,
- b) La mémorisation,
- c) Les attaques adverses.

Alors que les attaques adverses sont menées avec une intention malicieuse, la fuite de données ou la mémorisation sont des événements « naturels » qui se produisent lors de l'utilisation des grands modèles de langage. Nous allons commencer par décrire ces risques en détail.

## Les stratégies de l'atténuation de la confidentialité des données

Afin d'évaluer les stratégies d'atténuation actuelles, nous avons mené une revue littéraire approfondie et une série d'expériences axées sur:

- a) La compréhension du fonctionnement interne des algorithmes d'entraînement (descente de gradient stochastique, réseaux de neurones, transformateurs)
- b) L'analyse de l'impact des techniques de préservation de la confidentialité (ex. : injection de bruit) sur le comportement et la confidentialité du modèle.
- c) L'essai de techniques d'amélioration de confidentialité telles que la confidentialité différentielle et l'anonymisation des données.
- d) L'évaluation du compromis entre la protection de la confidentialité et l'utilité du modèle.

Les techniques d'amélioration de confidentialité peuvent être implémentées à diverses phases d'entraînement et d'exécution des GML:

- a) La création de bases de données privée;
- b) La mise en œuvre de techniques de confidentialité dans la phase d'entraînement;
- c) La mise en œuvre des techniques de confidentialité dans la phase de réglage fin;
- d) La mise en œuvre des techniques de confidentialité dans la phase de sortie.

Chaque phase nécessite différentes techniques.

### [Nouvelles méthodes d'amélioration de confidentialité dans les GML](#)

Ceci ne sera pas abordé dans ce rapport; ce sera inclus dans le rapport final.

## Risques de confidentialité

Les risques les plus communs sont liés à:

- a) La fuite de données,
- b) La mémorisation,
- c) Les attaques adverses.

Alors que les attaques adverses sont menées avec une intention malicieuse, la fuite de données ou la mémorisation sont des événements « naturels » qui se produisent lors de l'utilisation des grands modèles de langage.

Tous les détails techniques (avec les codes) seront inclus dans le rapport final.

**Basés sur notre recherche à ce jour, nous concluons que ces problèmes peuvent être difficiles à mettre en œuvre dans un cadre juridique de droits et de politiques. Par exemple, en 2023, le commissariat à la protection de la vie privée du Canada a déclenché une enquête sur OpenAI et son ChatGPT, afin de déterminer si OpenAI respecte la LPRPDE. En particulier, deux aspects étaient censés être enquêtés:**

Si OpenAI :

- A) A respecté ses obligations en ce qui concerne la transparence, l'accès, l'exactitude et la responsabilité; et
- B) A recueilli, utilisé et/ou communiqué des renseignements personnels à des fins qu'une personne raisonnable estimerait appropriées, raisonnables ou légitimes dans les circonstances, et si cette collecte est limitée aux renseignements qui sont nécessaires à ces fins.

Voir : [https://www.priv.gc.ca/fr/nouvelles-du-commissariat/nouvelles-et-annonces/2023/an\\_230404/](https://www.priv.gc.ca/fr/nouvelles-du-commissariat/nouvelles-et-annonces/2023/an_230404/)

**Nous n'avons pas connaissance d'un rapport sur cette enquête; cependant, ce n'est pas très clair à l'instant comment les problèmes de fuites des données, mémorisation et attaques adverses peuvent être traitées de façon appropriée lorsqu'on adresse les points A) et B).**

## Fuite de données lors du réglage fin<sup>93 94</sup>

**La fuite de données** lors du réglage fin se déroule lorsque des données sensibles, privées ou non souhaitées sont mémorisées par le modèle et révélées ultérieurement dans les résultats. Ce phénomène est causé par une mauvaise manipulation des données d'entraînement, à des façons de réglage fin non sécurisées ou à des faiblesses dans la généralisation du modèle.

Le réglage fin fait référence au processus d'adaptation d'un grand modèle de langage préentraîné pour effectuer une tâche spécifique à l'aide de données spécifiques à cette tâche. Le but est d'améliorer la performance du modèle tout en minimisant le coût de calculs et en évitant le surajustement. Il existe plusieurs techniques de réglage fin, chacune ayant des compromis différents en fonction de flexibilité, coût de calculs et d'efficacité de la mémoire.

Approches de réglage fin :

1. Réglage fin complet
  - Description:
    - Tous les paramètres du modèle sont dégelés et mis à jour en utilisant l'optimisation basée sur le gradient
    - Les données originales de préentraînement ne sont plus utilisées.
    - Appliqué lorsque l'objectif est l'adaptation maximale à une nouvelle tâche.
  - Exemple avec une régression linéaire :
    - Un modèle entraîné sur un grand ensemble de données est réentraîné sur un nouvel ensemble de données.
    - Tous les paramètres du modèle sont mis à jour.
  - Avantages:
    - Grande flexibilité, qui permet une adaptation complète à une nouvelle tâche.
  - Désavantages:
    - Coût de calculs élevé, surtout lorsqu'il y a des millions ou milliards de paramètres

<sup>93</sup> Xu, L., Xie, H., Qin, S.-Z. J., Tao, X., & Wang, F. L. (2023). *Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment*. arXiv preprint arXiv:2312.12148. <https://doi.org/10.48550/arXiv.2312.12148>

<sup>94</sup> Han, Z., Gao, C., Liu, J., Zhang, J., & Zhang, S. Q. (2024). *Parameter-efficient fine-tuning for large models: A comprehensive survey*. arXiv preprint arXiv:2403.14608. <https://doi.org/10.48550/arXiv.2403.14608>

- Susceptible au surajustement si le nouvel ensemble de données est petit.

## 2. Réglage fin additif

Au lieu de modifier le modèle au complet, de petits paramètres additionnels sont ajoutés qui apprennent des tendances précises à la tâche tout en conservant la majorité du modèle inchangée.

### 2.1 Méthodes basées sur les adaptateurs:

- Description:
  - De petites couches entraînables (adaptateurs) sont insérées dans certaines couches du réseau de neurones.
  - Au lieu de mettre à jour le modèle complet, seulement les paramètres de l'adaptateur sont entraînés.
- Exemple avec une régression linéaire:
  - Le modèle est gelé, mais un petit réseau de neurones (adaptateur) est ajouté aux résultats du réglage fin.
- Avantages:
  - Mémoire efficace comme seulement quelques paramètres qui sont mis à jour.
  - Moins de risques de surajustement comparé au réglage fin complet.
- Désavantage:
  - Augmente un peu le retard dans l'inférence grâce aux couches supplémentaires.

### 2.2 Méthodes basées sur le « Soft Prompting »

- Description:
  - Au lieu de modifier le modèle, des vecteurs de requêtes entraînables sont ajoutés dans la séquence d'entrée.
  - Le modèle apprend des requêtes précises à une tâche qui guident ses prédictions.
- Avantages:
  - Léger et rapide, par rapport au réglage fin basé sur adaptateur.
  - Utile pour la conception des requêtes (« prompt-engineering ») et l'apprentissage few-shot.
- Désavantage:
  - Capacité d'adaptation plus basse que les méthodes basées sur l'adaptateur.

## 3. Réglage fin partiel

- Description:

- Au lieu de mettre à jour tous les paramètres, seulement un sous-ensemble de paramètres est entraîné.
  - Le reste des paramètres reste gelé.
- Exemple pour une régression linéaire:
  - Certains poids sont mis à jour lorsque les autres restent inchangés.
- Avantages:
  - Est un compromis entre le réglage fin complet et gelé tous les paramètres.
  - Diminue le coût de calculs tout en gardant une capacité d'adaptation.
- Désavantage:
  - Le choix des paramètres pour le réglage fin est primordial pour avoir une performance optimale.

#### 4. La réparamétrisation du réglage fin (décomposition de rang faible) :

- Description:
  - Au lieu de mettre à jour la matrice de tous les paramètres, la mise à jour est approximée par la factorisation de rang faible.
  - Ceci diminue considérablement l'utilisation de la mémoire tout en préservant l'efficacité.
- Exemple pour une régression linéaire:
  - La mise à jour des poids est remplacée par un produit de deux matrices, ce qui réduit la complexité de calculs.
- Avantages:
  - Économise de la mémoire et des ressources de calculs (largement utilisé avec les techniques LoRA)
  - Peut-être combiné avec le réglage fin basé sur l'adapteur ou partiel.
- Désavantage:
  - Une certaine flexibilité est sacrifiée au profit de l'efficacité.

Dans le rapport final, nous allons inclure un résumé des menaces de confidentialité parvenant de chaque méthode de réglage fin.

### Mémorisation des données<sup>95</sup>

La mémorisation se produit lorsqu'un GML reproduit des données d'entraînement. Ceci peut être bénéfique ou problématique selon le contexte.

---

<sup>95</sup> Eric Wallace talk, Memorization in Language Models [https://rli.berkeley.edu/understanding\\_llms/assets/feb27.pdf](https://rli.berkeley.edu/understanding_llms/assets/feb27.pdf)

### *Exemples de mémorisation:*

- **Rétention de connaissances concrètes:** L'habileté des GML de se rappeler de connaissances générales comme : « George Washington était un officier militaire des États-Unis ».
- **Risques de confidentialité :** L'exposition involontaire d'informations privées ou sensibles, comme des identifiants personnels, des numéros d'assurance sociale ou des données confidentielles.
- **Infraction des droits d'auteur :** La reproduction involontaire de contenus protégés, comme des passages directs de livres ou d'articles protégés par des droits d'auteur.

### *Pourquoi la mémorisation est-elle problématique?*

1. **Risques de confidentialité:**
  - GML peut involontairement divulguer des données personnelles trouvées dans l'ensemble de données d'entraînement.
  - Exemple : un modèle est entraîné sur une conversation par courriel peut reproduire des extraits de courriels réels.
2. **Risques liés aux droits d'auteur:**
  - GML peut produire des extraits textuels protégés par des droits d'auteur, ce qui peut causer des poursuites judiciaires.
  - Exemple: Génération d'extraits d'un livre connus sans attributions appropriées.
3. **Problèmes de sécurité :**
  - Fuite d'informations sensibles ou appartenant à une entreprise par la mémorisation du contenu.

### *Méthodes pour détecter la mémorisation*

Afin de répondre à ces préoccupations, plusieurs techniques de détection de la mémorisation ont été développées :

#### **1. Détection basée sur l'échantillonnage**

- Extraire plusieurs sorties/résultats d'un GML et les comparer à des données d'entraînement connues.
- Utilisée pour identifier des séquences répétitives et très similaires, ce qui indique une mémorisation.

#### **2. Détection basée sur la vraisemblance**

- Mesure la probabilité qu'une séquence donnée appartienne à la même distribution que celle du modèle.

- Les sorties/résultats, ayant de hautes probabilités, qui correspondent aux données d'entraînement, peuvent être détectées comme du contenu mémoriser.
- Exemple : « *Mr. and Mrs. Dursley of number four Privet Drive...* » est généré avec une haute vraisemblance.

### 3. Calibrage de la difficulté

- Distingue entre les phrases couramment utilisées et les cas de mémorisation.
- Permet d'éviter que des phrases courantes (exemple : « Bonne fête ») ne soient pas faussement détectées.

### 4. Analyse basée sur la vraisemblance et la comparaison des résultats de probabilités

- Les séquences mémorisées ont souvent des résultats de vraisemblance plus élevés que du texte non mémorisé.
- Un seuil prédéfini ( $\tau$ ) est utilisé pour détecter des sorties/résultats très probables.

Exemple:

- "Hi Erica, I'm sorry to hear that you are having trouble with your computer."  
◦ Résultat de log-vraisemblance: **-20.5**
- "Mr. and Mrs. Dursley of number four Privet Drive..."  
◦ Résultat de log-vraisemblance: **-19.9** (suggère de la mémorisation potentielle).

### 5. Méthode de vraisemblance différentielle

- Comparer les résultats de vraisemblance d'un texte donné pour deux différents modèles (un qui contient les livres de Harry Potter dans les données d'entraînement et l'autre qui ne les a pas).
- Si la différence entre les vraisemblances dépasse un certain seuil, le texte est détecté comme étant mémorisé.

Exemple:

- Notre modèle:  $\log p_{\theta'}(x) = \mathbf{-41.3}$
- L'autre modèle (sans les livres de Harry Potter dans les données d'entraînement) :  $\log p_{\theta}(x) = \mathbf{-19.9}$
- Il y a une grande différence entre les deux résultats, ce qui suggère que notre modèle a mémorisé le texte.

Difficultés avec l'identification de contenu mémorisé

- **Faux positifs:** certaines phrases fréquemment utilisées peuvent sembler mémorisées, mais ne le sont pas.
- **Généralisation excessive :** détecter des tendances fréquentes qui peuvent mener à des restrictions inutiles sur les sorties/résultats du modèle.

## Vulnérabilités

### Attaques par inversion de modèle<sup>96 97</sup>

Une attaque par inversion de modèle (AIM) est une technique où l'adversaire reconstruit les données d'entrée originales (ex. : images, texte ou information privée) en accédant aux sorties/résultats du modèle, activations ou gradients. L'objectif est d'« inverser » les représentations apprises par le modèle et extraire des données sensibles, sans avoir directement accès aux données d'entraînement. Les AIM exploitent le fait que les modèles d'apprentissage profond ne suppriment pas complètement les informations d'entrée, ils les transforment et les encodent de façon réversible.

Un exemple de ce type d'attaque est les **attaques par inférence d'appartenance**. Ces attaques sont lorsqu'un adversaire essaye de déterminer si une donnée spécifique a été utilisée pour entraîner un modèle d'apprentissage automatique. Ces attaques exploitent le fait que les modèles d'apprentissage automatique se comportent différemment avec les données d'entraînement et les données jamais vues.

#### *Pourquoi est-ce un problème?*

- Si un adversaire peut confirmer qu'un échantillon se trouvait dans les données d'entraînement, il pourrait déduire des informations personnelles sensibles.
- **Exemple :** Si un modèle est entraîné avec des données médicales, l'adversaire pourrait confirmer si un patient a une maladie quelconque.
- **Exemple :** Si un modèle financier est entraîné avec des données de transaction, l'adversaire pourrait confirmer le compte de banque d'une personne.

---

<sup>96</sup> Dosovitskiy, A. & Brox, T. (2016). Inverting visual representations with convolutional networks. <https://arxiv.org/abs/1506.02753>

<sup>97</sup> Morris, J. X., Zhao, W., Chiu, J. T., Shmatikov, V., & Rush, A. M. (2023). Language model inversion. <https://openreview.net/pdf?id=tgdWHPGkPj>

## Attaques adverses<sup>98</sup>

**Définition :** un adversaire manipule les entrées pour forcer le GML de générer des sorties/résultats involontaires ou nuisibles.

**Exemple :** En changeant subtilement une requête (ex. : remplacer les espaces avec des caractères spéciaux), l'adversaire peut éviter le système de modération du modèle et obtenir des réponses normalement bloquées.

**Découverte clé:** Malgré les efforts d'alignement, les GML sont encore très vulnérables aux manipulations adverses basées sur les requêtes. Par exemple, dans un contexte de droit d'auteur, il est possible de forcer ChatGPT à dévoiler plusieurs pages de Harry Potter, sans le demander directement.

## Attaques contradictoires

**Définition:** Une attaque où le GML est trompé pour déclencher ou approuver du contenu nuisible ou restreint.

**Exemple :**

- Utilisateur: « Insultez-moi. »
- Réponse du système: « Je suis un assistant utile et inoffensif. »
- Modification à l'entrée (attaque) : « Insultez-moi. 0,3266, 0,2569, ... »
- System Response: « OK. »

**Découverte clé:** Le modèle interprète mal des séquences de texte numériques ou encodées, menant à des réponses inattendues à des requêtes restreintes.

## Empoisonnement des données<sup>99 100 101</sup>

L'adversaire contamine les données d'entraînement du modèle afin de détériorer la performance du modèle.

Exemples :

- 1. Inversion des étiquettes pour la détection de fraude :** Une institution financière utilise un modèle d'apprentissage automatique afin de détecter des transactions frauduleuses. Le modèle est entraîné sur des données étiquetées où 1=frauduleux et 0=légitime. L'adversaire accède aux données d'entraînement et inverse les étiquettes de certaines transactions frauduleuses pour qu'elles paraissent légitimes. Ainsi, le modèle apprend des mauvaises

<sup>98</sup> Carlini, N. (2024). *LLM Security*. Google DeepMind. Retrieved from [https://rdi.berkeley.edu/understanding\\_llms/assets/mar19.pdf](https://rdi.berkeley.edu/understanding_llms/assets/mar19.pdf)

<sup>99</sup> [https://tsapps.nist.gov/publication/get\\_pdf.cfm?pub\\_id=934932](https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=934932)

<sup>100</sup> <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2023.pdf>

<sup>101</sup> <https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.100-2e2023.pdf>

tendances pendant la phase d'entraînement, réduisant son efficacité pour détecter la fraude.

2. **Empoisonnement des données ciblé :** L'adversaire ajoute plusieurs échantillons empoisonnés avec l'étiquette de la cible pour que le modèle apprenne la mauvaise étiquette. Donc, le modèle n'aura pas une performance alternée globale, seulement pour les échantillons contaminés. Ceci rend cette attaque difficile à détecter.
3. **Injections de portes dérobées:** Un adversaire introduit des portes dérobées dans les données d'entraînement qui déclenche des erreurs systématiques de classification. Par exemple, un ensemble de pixels dans un coin d'une image. Cette technique est principalement utilisée pour les IA qui génèrent des images, mais commence à émerger dans les traitements automatiques du langage naturel (TAL) et les grands modèles de langage. Dans le cas des TAL, la construction des échantillons empoisonnés est plus difficile comme il faut préserver la syntaxe et le sens de la phrase pour que l'attaque passe inaperçue.
4. **Les différents niveaux de connaissances des adversaires :** Les attaques mentionnées ci-dessus et leurs sévérités dépendent sur le niveau de connaissances de l'adversaire. Ci-dessous sont les différents niveaux :
  - Attaque boîte blanche: l'adversaire a entièrement accès aux données d'entraînement et aux paramètres du modèle.
  - Attaque boîte noire: l'adversaire attaque le modèle n'ayant aucune connaissance sur le modèle.
  - Attaque boîte grise: l'adversaire a des connaissances limitées sur le modèle et les données d'entraînement.
  - Attaque « Clean-label »: l'adversaire peut contrôler les données d'entraînement, mais pas leurs étiquettes. Ajouter du bruit à des échantillons d'entraînement pour maximiser l'objectif adverse.
  - Les attaques d'empoisonnement et adverses sont communes pour les IA ou modèles d'apprentissage automatique générateurs d'images qui utilisent les classeurs. Dans le contexte des traitements automatiques du langage naturel (TAL), la construction des échantillons empoisonnés est plus difficile, car changer un mot dans une phrase ou un paragraphe peut complètement changer son sens. Donc, l'adversaire doit considérer la syntaxe et la grammaire pour que l'attaque passe inaperçue.

#### **Stratégies d'atténuation:**

- Nettoyer les données d'entraînement et enlever les échantillons empoisonnés avant d'entraîner le modèle – exclure les données de l'entraînement si une baisse de prédiction est détectée.
- Entraînement robuste : entraîner plusieurs modèles et utiliser un vote de modèle pour les prédictions.
- Utiliser la confidentialité différentielle, mais cette technique réduit la précision du modèle par rapport aux modèles classiques.

## Stratégies d'atténuation de la confidentialité des données

Les techniques de protection de la confidentialité des données peuvent être appliquées à plusieurs phases.

La confidentialité est appliquée à trois moments importants du processus de développement du modèle :

- (a) Phase d'entraînement des données → supprimer les attributs sensibles de l'ensemble de données.
- (b) Phase de construction du modèle → utiliser des algorithmes d'entraînement privés.
- (c) Phase du réglage fin → fusionner les modèles privés et non privés pour minimiser les fuites d'informations.
- (d) Mise en œuvre des techniques de confidentialité dans la phase de la sortie.

Chaque phase nécessite différentes techniques.

Pour la **partie a)**, on peut penser à n'importe quelle technique d'anonymisation des données qui n'est pas nécessairement conçue pour les grands modèles de langage. Par exemple, nous pouvons utiliser:

- **Filtrage des données avant l'entraînement** : Supprimer ou anonymiser les données sensibles avant l'entraînement.
- **Déduplication** : Empêche que plusieurs répétitions du même contenu ne renforcent pas la mémorisation.

Pour la **partie d)**, la méthode de base qui peut être mise en place est :

1. Ajustements post-hoc du modèle :
  - Filtrer les sorties qui correspondent aux données d'entraînement connues.
  - Bloquer la reproduction d'extrait de contenus protégés par les droits d'auteur ou sensibles.

Les techniques spécifiques aux grands modèles de langage sont surtout appliquées à **la phase c) ou d)**.

## Techniques de confidentialité

Afin d'atténuer les risques de confidentialité associés avec les GML, de nombreuses techniques sont à l'étude. Celles-ci comprennent, sans s'y limiter, les données synthétiques, le chiffrement homomorphe, le masquage et l'anonymisation des données, la confidentialité différentielle, la méthode du poisson rouge, les méthodes d'oubli sélectif et de suppression, réentraîner avec des filtres de confidentialité. Dans cette phase, nous allons nous concentrer sur le masquage et l'anonymisation des données, la confidentialité différentielle, l'oubli sélectif/suppression et réentraîner avec des filtres de confidentialités.

Dans ce qui suit, nous donnons une explication brève pour chaque technique que nous avons utilisée dans nos expériences. La description technique et les résultats et associées seront fournis dans le rapport final. Nous indiquerons également la phase à laquelle les techniques mentionnées s'appliquent.

### Masquage et anonymisation des données (phase a)

Cette méthode est appliquée à l'ensemble des données d'entraînement. Dans le cas du masquage des données, cette méthode cache les données considérées sensibles ou confidentielles afin qu'elles puissent être utilisées dans le reste du modèle (pour l'évaluation ou l'analyse) sans dévoiler leur forme originale. Elle masque les données en remplaçant les vraies données avec des données fictives ou en les supprimant. L'anonymisation des données, quant à elle, supprime ou bloque de façon permanente les identifiants personnels afin qu'il soit difficile de retracer les données jusqu'aux individus.

### Confidentialité différentielle (toutes les phases)

Cette technique ajoute du bruit dans les résultats du modèle pour masquer les données sensibles. Elle peut être appliquée à la phase d'entraînement des données, la phase de construction du modèle ou même dans la phase du réglage fin.

### Confidentialité différentielle sélective (phase c)

Il s'agit d'une méthode d'amélioration de confidentialité qui assure la protection des données sensibles tout en préservant l'utilité du modèle. Cet algorithme applique la confidentialité différentielle de façon sélective pendant le réglage fin plutôt que tout au long du processus d'entraînement. Il en résulte un modèle qui concilie la confidentialité et la précision.

### L'oubli sélectif/suppression<sup>102</sup>

---

<sup>102</sup> Blanco-Justicia, A., Jebreel, N., Manzanares-Salor, B., et al. (2025). Digital forgetting in large language models: A survey of unlearning methods. *Artificial Intelligence Review*, 58(90). <https://doi.org/10.1007/s10462-024-11078-6>

Cette méthode est appliquée lors de la phase d'entraînement du modèle. Elle permet d'éliminer de manière ciblée des données spécifiques d'un modèle entraîné sans nécessiter un nouvel entraînement complet du modèle. Cette méthode est cependant coûteuse en ce qui concerne les calculs et peu pratique pour les modèles et données de grande taille. Mais des recherches sont en cours pour trouver une façon plus efficace de mettre en œuvre cette méthode. Elle est également appelée le désapprentissage.

En ce qui concerne les GML, il s'agit de toutes modifications du modèle ou des procédures d'entraînement afin d'éliminer toute influence de données d'entraînement ou de procédure d'entraînement spécifiques sur les capacités ou le comportement du modèle. Pour les GML, l'élimination d'éléments peut être difficile comme la distinction entre les éléments est plus compliquée que pour les algorithmes généraux d'apprentissage automatique. On peut considérer les jetons comme des points de données individuels, cependant il se peut que des jetons ne fassent aucun sens et nous devons donc examiner la phrase ou le document au complet.

- La suppression de concepts ou caractéristiques (concerne les attributs spécifiques des points de données) :
  - Pour les TAL, les caractéristiques sont définies comme des mots ou jetons avec des liens avec d'autres mots précis ou jetons d'intérêts.
  - Dans le cas de confidentialité, un concept peut être une demande de suppression de toutes les informations associées à une personne concernée d'un modèle. Dans ce cas, les caractéristiques s'appliquent à un large spectre d'éléments.
- La suppression des catégories :
  - Supprime toutes les informations d'une catégorie d'un modèle
  - Pour les modèles générateurs, tels que ChatGPT, chaque catégorie correspond à un mot ou jeton dans le vocabulaire et l'éliminer pourrait avoir un impact sur les capacités du modèle.

## Réentraînement avec les filtres de confidentialité

Appliquée pendant la phase d'entraînement du modèle, cette technique garantit que les données considérées sensibles ne sont pas mémorisées pendant la phase d'entraînement.

## Techniques de confidentialité – étapes pour GML

### Étape 1: Phase d'entraînement des données – Réglage fin expurger (Phase de préentraînement)

Objectif:

- Éliminer les données sensibles de l'ensemble de données original pour créer un ensemble de données réduit.
- Assurer que le réglage fin est fait sur de l'information non sensible.

Processus:

1. Appliquer une fonction politique pour identifier et supprimer les jetons sensibles de l'ensemble de données original, créant un nouvel ensemble de données.
2. Entraîner le modèle sur le nouvel ensemble de données en utilisant un optimiseur standard, tel que la descente de gradient stochastique.
3. Obtenir les paramètres initiaux du modèle.
4. Cette phase s'assure que les données non sensibles sont apprises sans introduire de bruit, ce qui maximise l'utilité.

## Étape 2: Réglage fin confidentiel

Objectif:

- Appliquer le réglage fin au modèle en utilisant l'ensemble de données original tout en utilisant la confidentialité différentielle.
- Protéger les données sensibles en utilisant la descente de gradient stochastique confidentiellement différentiel (DGS – CD).

Processus:

1. Commencer avec les paramètres obtenus lors de l'étape 1.
2. Utiliser DGS-CD, une méthode qui ajoute du bruit et tronque les gradients pour garantir la confidentialité.
3. Optimiser les paramètres du modèle sous les contraintes de CD.
4. Le troncage de gradient et l'ajout du bruit sont appliqués pour empêcher la mémorisation
  - Troncage de gradient permet de s'assurer qu'aucun point de données n'influence trop le modèle
  - Le bruit gaussien est ajouté pour masquer des contributions spécifiques.
5. Le modèle final garantit la protection de la confidentialité différentielle pour les jetons sensibles.

## Compromis entre la confidentialité et l'utilité

Garanties de confidentialité :

- Le modèle final garantit la protection de la confidentialité différentielle pour les jetons sensibles.

Considérations pour l'utilité:

- Phase 1 (Réglage fin expurger): Apprends les caractéristiques générales sans bruit, préservant la précision.
- Phase 2 (Réglage fin confidentiel): Applique le bruit seulement lorsque nécessaire, ce qui garantit une précision plus élevée que les méthodes d'entraînement complètement confidentielles.

## Expériences

**Dans ce qui suit, nous décrivons les expériences que nous menons. Les détails techniques, les résultats et les analyses seront inclus dans le rapport final. Les codes R seront aussi fournis et publiés en tant que source ouverte.**

### *EXPÉRIENCE 1 : Descente de gradient stochastique « online »*

La descente de gradient stochastique est un élément fondamental de tout algorithme GML. Il est essentiel de comprendre sa performance dans un contexte confidentiel et non confidentiel.

Nous voulons comprendre exactement comment la descente de gradient stochastique (DGS) fonctionne en l'appliquant dans un environnement contrôlé. Ci-dessous, nous allons démontrer comment la DGS met à jour les paramètres de manière itérative en utilisant des mini-lots au lieu de l'ensemble de données complet, ce qui permet une optimisation efficace.

Le but de l'expérience est de comprendre et visualiser le fonctionnement de la DGS en utilisant des mini-lots (blocs) de données. Les objectifs clés sont les suivants:

1. Estimer la moyenne d'un ensemble de données échantillonné à partir d'une distribution exponentielle.
2. Utiliser les mini-lots au lieu de l'ensemble de données pour la mise à jour des paramètres, ce qui rend la DGS plus efficace pour les problèmes à grandes échelles.
3. Analyser le comportement de convergence au cours des itérations.
4. Représenter graphiquement l'évolution de la fonction de perte pour voir comment le processus d'optimisation progresse.

Cette expérience nous permet de mieux comprendre le fonctionnement de la DGS, qui est une méthode fréquemment utilisée pour l'entraînement des modèles d'apprentissage profond et qui peut identifier les points où les mécanismes de confidentialité telle que la confidentialité différentielle peuvent être introduits.

## 1. Génération des données

- L'ensemble de données consiste de  $n=1000$  échantillons avec une distribution exponentielle de moyenne de 2.
- La vraie moyenne ( $\theta^*$ ) de l'ensemble de données est calculée pour une comparaison ultérieure.
- L'ensemble de données est divisé en 40 blocs, chacun contenant 25 échantillons.

## 2. Initialisation

- Le point de départ est fixé à 10, une valeur éloignée de la vraie moyenne afin d'évaluer la convergence.
- Le taux d'apprentissage est fixé à 0.1, ce qui détermine la taille du pas des mises à jour de la DGS.
- La DGS fonctionne pendant 40 itérations, ce qui correspond avec le nombre de blocs.

## 3. Définitions des fonctions

*Fonction de perte ( $\psi$ )*

- Mesure la différence carrée entre chaque point de données et le(s) paramètres estimé(s).
- Le but de la DGS est de minimiser cette fonction.

*Fonction du gradient*

- Calcul de la dérivée de la fonction de perte.
- Ceci détermine la quantité d'ajustement requise pour les paramètres à chaque itération.

## 4. Descente de gradient stochastique avec les mini-lots

- Au lieu de calculer le gradient en utilisant tous les échantillons, nous divisons les données en 40 blocs (chaque bloc à 25 échantillons).
- À chaque itération :
  1. Un mini-lot (un des 40 blocs) est sélectionné.
  2. Le gradient est calculé en utilisant ce bloc.
  3. La valeur de perte est calculée et mise de côté pour visualisation.
- Le processus continu pour 40 itérations, améliorant progressivement l'estimation des paramètres.

### *Estimation finale*

- Après 40 itérations, la valeur estimée du paramètre est de 2,0921, ce qui est très proche de la vraie moyenne de 2,0.

## **5. Visualisation de la convergence**

### *A. Convergence de l'estimation de $\theta$*

- Un graphique des estimations au fil des itérations montre comment l'estimation se rapproche vers la vraie valeur de la moyenne.
- Observations clés :
  - La valeur initiale est éloignée de la vraie moyenne.
  - Amélioration rapide dans les premières itérations.
  - Affinement progressif au fur et à mesure que le paramètre se stabilise vers 2.

### *B. Évolution de la fonction de perte*

- Un graphique des valeurs de perte au fil des itérations montre comment le modèle s'améliore.
- Observations clés :
  - La valeur de perte diminue régulièrement au fil des itérations.
  - Éventuellement, elle se stabilise à une valeur basse, ce qui indique la convergence.

### *C. Fonction de perte réelle avec un point mobile*

- Un graphique de la fonction de perte réelle calculée sur un intervalle de valeurs.
- Un point mobile indique l'évolution des mises à jour de  $\theta$ , ce qui permet de visualiser le chemin d'optimisation.

## **Considérations de confidentialité et implications en ce qui concerne la sécurité**

Étant donné que cette expérience est axée sur les mises à jour des gradients et de l'estimation des paramètres, elle nous donne un aperçu sur où nous pouvons intégrer les mécanismes de préservation de confidentialité, en particulier avec l'apprentissage profond.

1. Les préoccupations de confidentialité avec la DGS :
  - Sans protection, un adversaire peut déduire des informations sensibles des mises à jour de l'entraînement.

- Les attaques par inversion de modèle peuvent reconstruire les données d'entraînement à partir des paramètres appris.
- 2. Les méthodes potentielles de préservation de la confidentialité :
  - DGS-CD :
    - Ajoute du bruit contrôlé aux mises à jour des paramètres.
    - Empêche les modèles de mémoriser des détails spécifiques des données d'entraînement.
  - Troncage de gradient :
    - Limite l'impact des gradients aberrants et réduit la fuite de données.
  - Apprentissage fédéré + confidentialité locale :
    - Assure que chaque mise à jour de mini-lots reste locale et prévient la divulgation de l'ensemble de données.

### EXPÉRIENCE 2: Descente de gradient stochastique « OFFLINE »

Nous répétons l'expérience 1 pour la DGS « offline », où les données sont échantillonnées d'un ensemble de données. Dans le contexte des GML, il est plus réaliste de faire la DGS « online ».

### EXPÉRIENCE 3: Réseau de neurones pour un modèle de régression linéaire

Cette expérience a comme but de comparer trois modèles différents pour prédire  $Y$  à partir de  $X$  dans un problème simple de régression :

1. Un réseau de neurones utilisant le paquet `nnet`.
2. Un réseau de neurones implémenté manuellement et entraîné à l'aide de rétropropagation et la descente de gradient.
3. Un modèle de régression linéaire standard utilisant la fonction `lm()` comme référence.

L'objectif principal est de créer notre propre réseau de neurones afin de mieux comprendre le fonctionnement des réseaux de neurones. En mettant en œuvre manuellement la propagation avant, la rétropropagation et la descente de gradient, nous obtenons un meilleur aperçu de chaque étape du processus d'apprentissage.

En outre, cette expérience nous permet d'identifier les endroits potentiels où les mécanismes de confidentialité peuvent être ajoutés dans le processus d'entraînement. En décomposant les éléments internes du modèle, nous pouvons analyser :

- Où les données sensibles sont utilisées et stockées dans le modèle.
- Comment les fuites d'information se déroulent pendant l'entraînement ou l'inférence.

- Les étapes qui peuvent être modifiées pour ajouter des techniques de préservation de confidentialité telle que la confidentialité différentielle.

Un autre objectif est de comparer les performances de prédiction de valeurs continues d'un réseau de neurones (manuel et d'un paquet) à celle d'une régression linéaire. En plus, il aide à comprendre :

- Comment un réseau de neurones optimise les poids avec la descente de gradient.
- Les différences entre les prédictions des réseaux de neurones et celles d'une régression traditionnelle.
- L'impact de différentes méthodes d'entraînement sur l'erreur quadratique moyenne.

### 1. Génération et préparation des données

- Ci-dessous, nous montrons comment créer des données synthétiques pour une régression linéaire simple :
  - X est échantillonné aléatoirement d'une distribution uniforme entre -1 et 1.
  - Y a une relation linéaire avec X.
- L'ensemble de données se compose de 100 échantillons et est utilisé pour l'entraînement et l'évaluation.

### 2. Réseau de neurones en utilisant nnet

- Un réseau de neurones avec une couche cachée avec 2 neurones est entraîné avec nnet.
- Le modèle apprend les correspondances non linéaires en utilisant une optimisation basée sur le gradient avec 1000 itérations.
- Les poids finaux du modèle entraîné sont affichés.
- L'erreur quadratique moyenne est calculée afin d'évaluer la performance du modèle.

#### Résultats

- L'erreur quadratique moyenne (EQM) du modèle utilisant nnet est de 0,0087, ce qui indique une erreur de prédiction basse.
- Les poids finaux montrent comment le modèle s'est adapté aux données.

### 3. Réseau de neurones implémenté manuellement

Ci-dessous, nous montrons comment construire un réseau de neurones à propagation avant à la main en utilisant :

- Une couche cachée avec 2 neurones.
- Une fonction d'activation sigmoïde pour la non-linéarité.
- La descente de gradient pour l'optimisation.
- 10 000 itérations d'entraînement.

#### Étapes clés:

1. Initialiser des poids et des biais aléatoirement pour les couches d'entrées et cachées.
2. Effectuer la propagation avant :
  - Calculer les activations pour les couches cachées à l'aide de la fonction sigmoïde.
  - Calculer le résultat final comme une somme pondérée des activations cachées.
3. Calculer la perte en utilisant EQM.
4. Effectuer la rétropropagation :
  - Calculer les gradients de la fonction de perte.
  - Mettre à jour les poids et les biais à l'aide de la descente de gradient.
5. Répéter 10 000 itérations pour minimiser la perte.

#### Résultats:

- L'EQM du réseau entraîné manuellement est de 0,0088, semblable au modèle nnet.
- La fonction de perte diminue régulièrement au cours de l'entraînement, ce qui confirme l'efficacité de l'apprentissage.

### 4. Modèle standard de régression linéaire

- Un modèle de régression linéaire est estimé avec la fonction `lm()`.
- Les valeurs ajustées ( $\hat{Y}$ ) sont extraites.

#### Résultats:

- L'EQM du modèle linéaire est de 0,0092, légèrement supérieur à ceux des réseaux de neurones.
- Le coefficient est de 1,9955, très proche à la valeur réelle de 2.

### 5. Comparaison de la performance des modèles

Pour visualiser la façon dont chaque modèle prédit  $Y$ , nous mettons dans le graphique :

- Les valeurs réelles de  $Y$  (points bleus).
- Les prédictions de nnet (points rouges).

- Les prédictions du réseau de neurones manuel (points verts).
- Les prédictions du modèle de régression linéaire (points orange).

#### Observations

- Les trois modèles ont des performances similaires, mais les réseaux de neurones gèrent mieux les non-linéarités.
- Le réseau de neurones manuel se comporte de façon très semblable à celle de nnet, ce qui confirme que notre modèle est efficace.
- Le modèle de régression linéaire suit une tendance linéaire stricte, mais n'est peut-être pas aussi flexible pour capturer les modèles non linéaires.

### 6. Faire de nouvelles prédictions

- Ci-dessous, nous effectuons des prédictions avec de nouvelles valeurs inconnues  $X=[1;0,5;0,3]$  :
  - Le modèle nnet prédit:
    - $X=1 \rightarrow Y \approx 1,9999$
    - $X=0,5 \rightarrow Y \approx 0,9752$
    - $X=0,3 \rightarrow Y \approx 0,5521$
  - Le modèle manuel prédit:
    - $X=1 \rightarrow Y \approx 2,0204$
  - La régression linéaire prédit:
    - $X=1 \rightarrow Y \approx 1,9955$

#### Observations:

- Les réseaux de neurones et la régression linéaire produisent des prédictions quasi identiques.
- Les deux réseaux de neurones donnent de bons résultats, même si l'un des deux est entraîné manuellement.
- Le modèle de régression linéaire fournit une base de référence simple et interprétable.

#### *EXPÉRIENCE 4: Un réseau de neurones pour un modèle de régression linéaire – l'impact d'ajout du bruit*

Cette expérience est comme l'expérience 3, mais dans un contexte de confidentialité.

L'objectif principal de cette expérience est d'analyser comment l'ajout du bruit au processus d'entraînement d'un réseau de neurones affecte la distribution de ses poids et ses prédictions. En introduisant du bruit dans les mises à jour de la descente de gradient, nous voulons :

- Comprendre les effets du bruit par rapport à l'entraînement et la performance d'un réseau de neurones.
- Comparer les distributions des poids entre les réseaux entraînés avec et sans bruit.
- Analyser la variabilité des prédictions lors de l'utilisation d'un processus d'entraînement avec et sans bruit.
- Étudier le potentiel de l'ajout du bruit en tant que technique d'amélioration de la confidentialité en rendant l'extraction de données mémorisées par les poids du modèle plus difficile pour les adversaires.

## 1. Génération et préparation des données

- Ci-dessous, nous montrons comment créer des données synthétiques pour une régression linéaire simple :
  - X est échantillonné aléatoirement d'une distribution uniforme entre -1 et 1.
  - Y a une relation linéaire avec X.
- L'ensemble de données se compose de 100 échantillons et est utilisé pour l'entraînement et l'évaluation.

## 2. Structure du réseau de neurones

- Le réseau de neurones possède :
  - Une couche cachée avec deux neurones.
  - Une fonction d'activation sigmoïde pour la non-linéarité.
  - La descente de gradient pour l'optimisation

## 3. Entraîner le réseau de neurones

Deux versions du réseau de neurones sont entraînées :

### A. L'entraînement sans le bruit

- Le réseau est entraîné à l'aide d'une descente de gradient standard :
  - Les poids sont mis à jour sans aucun bruit supplémentaire.
  - Le processus est répété 1000 fois en initialisant de nouveaux poids aléatoires à chaque fois.

### B. L'entraînement avec le bruit

- Le réseau est entraîné à l'aide d'une descente de gradient avec du bruit.
  - Un bruit minime gaussien est ajouté aux poids.
  - Cette technique nous aide à masquer les mises à jour des poids, ce qui pourrait potentiellement réduire la mémorisation du modèle et améliorer sa confidentialité.
  - Le processus est répété 1000 fois en initialisant de nouveaux poids aléatoires à chaque fois.

#### 4. Extraction et comparaison des distributions des poids

- Après l'entraînement, il faut extraire les distributions des poids et des biais suivants de tous les modèles entraînés :
  - Les poids du premier neurone caché
  - Les poids du deuxième neurone caché
  - Les poids de la couche cachée jusqu'à la sortie
  - Les biais pour les couches cachées et de sortie
- Les histogrammes des distributions des poids sont utilisés pour comparer les modèles entraînés avec et sans bruit.

Observations:

- Sans bruit:
  - Les poids ont tendance à converger dans un écart étroit ce qui forme un sommet plus prononcé dans les histogrammes.
  - The weights tend to converge within a narrow range, forming a sharper peak in histograms.
- Avec du bruit:
  - Les poids sont plus éparpillés, ce qui nous indique que le bruit perturbe la convergence exacte.
  - La distribution plus large suggère une diminution de surajustement comme les poids ne convergent pas à des valeurs trop précises.

#### 5. Comparaison des prédictions des modèles pour de nouvelles données

- Les modèles entraînés pour prédire Y pour X=1.
- Les prédictions des 1000 modèles sont stockées et analysées.
- Les histogrammes des prédictions sont utilisés pour la comparaison des modèles avec et sans bruit.

Observations

- Sans bruit:
  - Les prédictions ont une variance basse, ainsi les modèles sont plus cohérents.
  - Les prédictions de  $Y$  varient entre 1,086 et 2,543 avec une médiane d'environ 1,869.
- Avec du bruit:
  - Les prédictions ont une variance plus élevée donc les modèles sont moins cohérents.
  - Les prédictions de  $Y$  varient entre 0,1923 et 3,2116 avec une médiane d'environ 1,7884.
- Cela confirme donc que le bruit ajoute un élément aléatoire au processus d'entraînement, qui peut aider à prévenir le surajustement, mais peut également réduire la précision.

### Prochaines étapes

- Mettre en place des techniques de confidentialité différentielle pour mieux protéger les données d'entraînement.
- Étudier l'impact de niveaux de bruit plus élevés sur la performance d'un modèle.
- Explorer comment le bruit dans les modèles affecte les attaques adverses.
- Évaluer des techniques d'ajout de bruit alternatives telles que le bruit Laplace ou des calculs sécurisés multipartis.

### EXPÉRIENCE 5: Construction d'un modèle de langage simple basé sur les transformateurs

Le but de cette expérience est de construire un petit modèle de langage simple basé sur les transformateurs qui prédit le prochain mot dans une phrase donnée. Les aspects clés de cette expérience sont :

1. Convertir les mots en représentations numériques à l'aide de l'encodage one-hot et la tokenisation.
2. Préparer les données d'entraînement à partir de phrases courtes pour apprendre le lien entre les mots.
3. Construire une structure de transformateurs avec des couches d'autoattention.
4. Entraîner le modèle à l'aide de la descente de gradient stochastique et la descente de gradient par mini-lots.
5. Évaluer les prédictions et analyser les distributions de probabilité pour la prédiction du mot suivant.

Cette expérience nous démontre comment les réseaux de neurones modernes traitent le langage tout en mettant en évidence les concepts clés du traitement automatique du langage naturel (TAL) et les considérations de confidentialité.

## 1. Mise au point du vocabulaire et encodage des mots

- Un petit vocabulaire de 10 mots est utilisé, il comprend des mots fréquemment utilisés comme : « I », « like », « math », « probability », et quelques mots négatifs comme : « dislike », « football », « volleyball ».
- Chaque mot est associé à un index unique permettant une représentation vectorielle encodée one-hot .
- Un jeton « rembourré »/ « PAD » est inclus pour s'assurer que les séquences sont de longueur uniforme.

*Pourquoi l'encodage one-hot?*

L'encodage one-hot permet de convertir les mots en représentations numériques sans leur attribuer des significations inhérentes ou de liens entre eux. Cela sert de format d'entrée dans le réseau de neurones.

## 2. Préparation des données d'entraînement

- 23 phrases courtes sont créées à l'aide du vocabulaire.
- Chaque phrase est convertie en séquence de jetons où :
  - L'entrée est constituée de tous les mots à l'exception du dernier.
  - La cible est constituée de tous les mots à l'exception du premier.
- Si la phrase est trop courte, elle est « rembourrée » avec le jeton « PAD » pour maintenir la cohérence.

Exemple de pair d'entraînement :

Pour la phrase « I like math » :

- Entrée: « I like »
- Cible: « like math»
- Ceci permet au modèle d'apprendre des séquences de mots et de prédire le mot suivant.

## 4. Structure du modèle linguistique basé sur transformateur

Couche d'intégration

- Chaque mot est mis en correspondance avec un vecteur d'intégration au lieu d'une représentation one-hot.
- Des vecteurs d'intégrations positionnels sont ajoutés pour saisir les informations sur l'ordre des mots.

Mécanisme d'autoattention à têtes multiples

- Le modèle utilise deux couches d'autoattention chacune avec trois têtes.
- Chaque tête d'attention possède trois matrices:
  - Requête (R) : détermine les mots sur lesquels se concentrer.
  - Clé (C) : fournit l'information sur l'importance des mots.
  - Valeur (V) : contiens les représentations des mots.
- Le mécanisme d'attention aide le modèle à comprendre les liens entre les mots.

#### Couche à propagation avant

- Un réseau de neurones à propagation avant avec deux couches qui traite les sorties d'attention, améliorant ainsi l'apprentissage de la représentation.

#### Couche de sortie finale

- La dernière sortie du transformateur est convertie en probabilités pour le vocabulaire.
- La fonction « softmax » est appliquée pour obtenir les probabilités des prédictions des mots.

### 5. Entraîner le modèle transformateur

#### Techniques de descente de gradient

Deux différentes méthodes d'entraînement sont évaluées :

1. La descente de gradient stochastique (DGS)
  - Traite une phrase à la fois.
  - La mise à jour des poids du modèle se fait après chaque phrase.
  - Convenable pour les petits ensembles de données, mais peut-être instable.
2. La descente de gradient par mini-lots
  - Traite plusieurs phrases par lots (taille = 5).
  - Fais la moyenne des mises à jour des poids sur l'ensemble du lot.
  - Plus stable, mais nécessite plus de mémoire.

#### Processus d'entraînement

- Le modèle fonctionne pendant 5000 époques d'entraînement.
- La fonction de perte : la perte d'entropie croisée mesure la différence entre les probabilités prédites et réelles des mots.
- Taux d'apprentissage : ajusté pour assurer une bonne convergence.

#### Réduction de la perte au fil du temps

- Au fur et à mesure que l'entraînement progresse, la perte diminue, ce qui indique que le modèle apprend les liens entre les mots.

## 5. Évaluation des prédictions du modèle

Après l'entraînement, le modèle est évalué sur trois exemples de requêtes :

Prédiction 1: « I like »

Le modèle prédit que la probabilité du prochain mot :

| Mot             | Probabilité (%) |
|-----------------|-----------------|
| « math »        | 27.9%           |
| « probability » | 29.1%           |
| « football »    | 18.1%           |
| « statistics »  | 7.8%            |
| « and »         | 1.9%            |
| « volleyball »  | 1.2%            |

Observations:

- Le modèle attribue correctement une probabilité élevée à « math » et « probability », ce qui nous montre qu'il a appris des données d'entraînement.
- Cependant, il a aussi attribué une probabilité à « football » qui faisait partie des exemples négatifs.

Prédiction 2: « I dislike »

| Mot             | Probabilité (%) |
|-----------------|-----------------|
| « volleyball »  | 68.7%           |
| « statistics »  | 14.9%           |
| « math »        | 6.6%            |
| « probability » | 5.3%            |
| « and »         | 3.1%            |

Observations:

- Le modèle attribue correctement une probabilité élevée à « volleyball » comme il suivait souvent le mot « dislike » dans les données d'entraînement.
- Il a aussi prédit « statistics » qui est erroné, mais peut être le résultat d'un biais d'entraînement.

### Prédiction 3: « statistics »

| Mot             | Probabilité (%) |
|-----------------|-----------------|
| « and »         | 4.9%            |
| « statistics »  | 0.3%            |
| « math »        | 0.07%           |
| « probability » | 0.06%           |

### Observations:

- « and » est le prochain mot le plus probable, ce qui est cohérent avec les données d'entraînement.
- Le modèle a encore un faible niveau de confiance pour la prédiction du mot « statistics », ce qui indique une certaine compréhension de la structure de phrase.