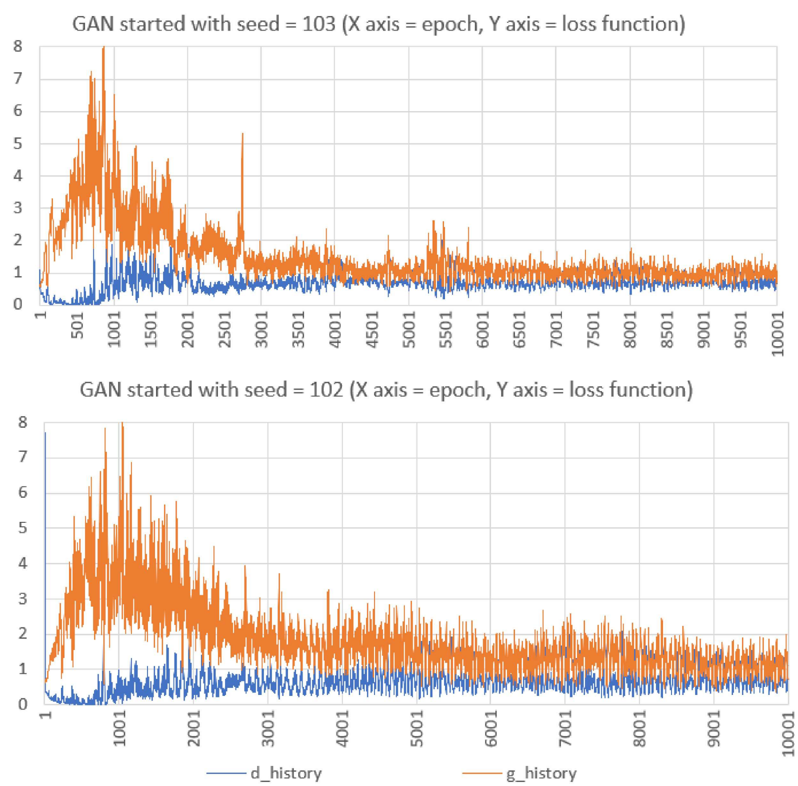
State-of-the-Art GenAI and LLMs Creative Projects, with Solutions



Vincent Granville, Ph.D. | www.MLTechniques.com | Version 4.3, May 2024

Preface

With 23 top projects, 96 subprojects, and over 6000 lines of Python code, this vendor-neutral coursebook is a goldmine for any analytic professional or AI/ML engineer interested in developing superior GenAI or LLM enterprise apps using ground-breaking technology.

Key Features

- 23 enterprise-grade projects, 96 subprojects all with solutions, 6000 lines of Python code, with real-world datasets and case studies.
- Ground-breaking technology covering RAG, multi-LLMs, knowledge graphs, synthetic data generation (tabular, geospatial, time series), generative adversarial network (GANs), model evaluation, adaptive loss functions, and more.
- Learn how to create fast, memory-frugal apps from scratch with explainable AI, to deliver better results.

Target Audience

This vendor-neutral coursebook is a goldmine for any analytic professional or AI/ML engineer interested in developing superior GenAI or LLM enterprise apps using ground-breaking technology. This is not another book discussing the same topics that you learn in bootcamps, college classes, Coursera, or at work. Instead, the focus is on implementing solutions that address and fix the main problems encountered in current applications. Using foundational redesign rather than patches such as prompt engineering to fix backend design flaws.

You will learn how to quickly implement from scratch applications actually used by Fortune 100 companies, outperforming OpenAI and the likes by several order of magnitudes, in terms of quality, speed, memory requirements, costs, interpretability (explainable AI), security, latency, and training complexity.

This book is also an invaluable resource to instructors and professors teaching AI, or for corporate training. Also, it is useful to prepare for job interviews or to build a robust portfolio. And for hiring managers, there are plenty of original interview questions. The amount of Python code accompanying the solutions is considerable, using a vast array of libraries as well as home-made implementations showing the inner workings and improving existing black-box algorithms. By itself, this book constitutes a solid introduction to Python and scientific programming. The code is also on GitHub.

Knowledge of Python and machine learning concepts is required to fully understand the use cases and code examples. However, the first chapter starts from scratch: getting a Python environment. Mathematics are used only when needed, rarely going beyond a first course in calculus, linear algebra and statistics. Advanced concepts, when useful, are introduced and explained in simple English.

What you will learn

- How to create fast, memory-frugal apps from scratch with explainable AI, to deliver better results.
- Solving a problem from start to finish, with Python: Learn how to choose the best algorithms, fine-tune, evaluate, and how to produce great visualizations including data animations (videos) to efficiently convey insights to stakeholders.
- How to use dozens of Python libraries in a business-like setting, including your home-made, and from the author. Covering anything from smart crawling, deep neural networks, scientific computing, model evaluation, NLP, and machine learning.

Step 1: Summary tables. Besides embeddings, what are the summary tables and underlying data structure, for each of them? How are they linked to the output returned to a user query? How to adapt the methodology if the input source is Wikipedia rather than Wolfram?

Step 2: Issues with Python libraries. By looking at my comments in the code or by experimenting yourself, identify bad side effects from standard Python libraries such as NLTK. Examples include stop-words, auto-correct, and singularize. The problems arise due to the specialization of my xLLM, but is otherwise minor in generic applications targeted to non-experts. Suggest workarounds.

Step 3: Improving xLLM. I implemented several enhancements, for instance: ignoring N-grams (permutations of N tokens) [Wiki] not found in the crawled data, using normalized PMI in embeddings as the association metric or weight between two tokens, working with variable-length embeddings (VLE) rather than vectors, managing accented characters, not combining tokens separated by punctuation signs, and minimizing stemming such as singular form. Find other potential enhancements and how to implement them. For instance, making sure that "analysis of variance" and "ANOVA" return the same results, or "San Francisco" is not split into two tokens. Hint: use mappings (synonyms) and double-tokens treated as a single token. Treat uppercase and lowercase differently.

Figure 1: The first three steps in the main xLLM project

Approach

Each project starts with a summary of the problem and an introduction to the concepts needed for completion. In several cases, projects do not consist of writing code from scratch, but instead, to understand the code offered by the author and build on top of it, and/or improve the speed. Each project is broken down into multiple steps of increasing scope and complexity. Two glossaries, a large index, and modern references are included. Considerable efforts have been mode to avoid jargon and advanced mathematics, using simple English as much as possible. Yet, all the terminology you need to know is introduced when needed.

Description

With tutorials, enterprise-grade projects with solutions, and real-world case studies, this coursebook covers state-of-the-art material on GenAI, generative adversarial networks (GAN), data synthetization, and much more, in a compact format with many recent references. It includes deep dives into innovative, ground-breaking AI technologies such a xLLMs (extreme LLMs), invented by the author.

The focus is on explainable AI with faster, simpler, high-performance, automated algorithms. For instance: NoGAN, new evaluation metrics, xLLM (self-tuned multi-LLM based on taxonomies with application to clustering and predictive analytics), variable-length embeddings, generating observations outside the training set range, fast probabilistic vector search, or Python-generated SQL queries. I also discuss alternatives to traditional methods, for instance to synthesize geo-spatial data or music.

Author Bio

Vincent Granville is a pioneering GenAI scientist and machine learning expert, co-founder of Data Science Central (acquired by a publicly traded company), Chief AI Scientist at MLTechniques.com and GenAItechLab.com, former VC-funded executive, author, and patent owner. Vincent's past corporate experience includes Microsoft, Visa, Wells Fargo, eBay, NBC, and CNET.

Vincent is also a former post-doc at Cambridge University. He published in Journal of Number Theory, Journal of the Royal Statistical Society (Series B), and IEEE Transactions on Pattern Analysis and Machine Intelligence. He is the author of multiple books, including "Synthetic Data and Generative AI" (Elsevier). Vincent lives in Washington state, and enjoys doing research on stochastic processes, dynamical systems, experimental math and probabilistic number theory.

Contents

1 Getting Started **6**
1.1 Python, Jupyter Notebook, and Google Colab . . . 6
1.1.1 Online Resources and Discussion Forums . . . 7
1.1.2 Beyond Python . . . 8
1.2 Automated data cleaning and exploratory analysis . . . 8
1.3 Tips to quickly solve new problems . . . 9
1.3.1 Original solution to visualization problem . . . 9
1.3.2 New solution, after doing some research . . . 11

2 Machine Learning Optimization **13**
2.1 Fast, high-quality NoGAN synthesizer for tabular data . . . 13
2.1.1 Project description . . . 13
2.1.2 Solution . . . 15
2.1.3 Python implementation . . . 16
2.2 Cybersecurity: balancing data with automated SQL queries . . . 22
2.2.1 Project description . . . 23
2.2.2 Solution . . . 23
2.2.3 Python code with SQL queries . . . 24
2.3 Good GenAI evaluation, fast LLM search, and real randomness . . . 25
2.3.1 Project description . . . 26
2.3.2 Solution . . . 28
2.3.3 Python implementation . . . 30
2.4 Spectacular use of GenAI evaluation as an adaptive loss function . . . 36
2.4.1 Project and solution . . . 38
2.4.2 Python code . . . 41

3 Time Series and Spatial Processes **47**
3.1 Time series interpolation: ocean tides . . . 47
3.1.1 Project description . . . 48
3.1.2 Note on time series comparison . . . 49
3.1.3 Solution . . . 50
3.2 Temperature data: geospatial smoothness and interpolation . . . 54
3.2.1 Project description . . . 55
3.2.2 Solution . . . 56

4 Scientific Computing **65**
4.1 The music of the Riemann Hypothesis: sound generation . . . 65
4.1.1 Solution . . . 66
4.2 Cross-correlations in binary digits of irrational numbers . . . 68
4.2.1 Project and solution . . . 68
4.3 Longest runs of zeros in binary digits of $\sqrt{2}$ . . . 69
4.3.1 Surprising result about the longest runs . . . 70
4.3.2 Project and solution . . . 71
4.4 Quantum derivatives, GenAI, and the Riemann Hypothesis . . . 73
4.4.1 Cornerstone result to bypass the roadblocks . . . 74
4.4.2 Quantum derivative of functions nowhere differentiable . . . 75
4.4.3 Project and solution . . . 76
4.4.4 Python code . . . 80

5 Generative AI **85**
5.1 Holdout method to evaluate synthetizations . . . 85
5.1.1 Project description . . . 86

5.1.2 Solution . . . 87
5.2 Enhanced synthetizations with GANs and copulas . . . 90
5.2.1 Project description . . . 90
5.2.2 Solution . . . 92
5.2.3 Python code . . . 93
5.3 Difference between synthetization and simulation . . . 103
5.3.1 Frequently asked questions . . . 103
5.3.2 Project: synthetizations with categorical features . . . 104
5.3.3 Solution . . . 105
5.4 Music, synthetic graphs, LLM, and agent-based models . . . 106

6 Data Visualizations and Animations **107**
6.1 Synthesizing data outside the observation range . . . 107
6.1.1 Animated histograms for extrapolated quantiles . . . 107
6.1.2 Python code: video, thumbnails . . . 108
6.2 Curve fitting in bulk . . . 111
6.2.1 Regression without dependent variable . . . 112
6.2.2 Python code . . . 112
6.3 Gradient descent: grid, orthogonal trajectories . . . 117
6.4 More spectacular videos and image generation . . . 122

7 NLP and Large Language Models **124**
7.1 Synthesizing DNA sequences with LLM techniques . . . 124
7.1.1 Project and solution . . . 124
7.1.2 Python code . . . 127
7.2 Creating high quality LLM embeddings . . . 131
7.2.1 Smart, efficient, and scalable crawling . . . 131
7.2.2 User queries, category-specific embeddings and related tables . . . 135
7.2.3 RAG: retrieval augmented generation using book catalogs . . . 144

8 Miscellaneous Projects **151**
8.1 Fast probabilistic nearest neighbor search (pANN) . . . 151
8.1.1 Motivation and architecture . . . 152
8.1.2 Applications . . . 153
8.1.3 Project and solution . . . 155
8.1.4 Python code . . . 156
8.2 Building and evaluating a taxonomy-enriched LLM . . . 160
8.2.1 Project and solution . . . 162
8.2.2 Python code . . . 163
8.3 Predicting article performance and clustering using LLMs . . . 170
8.3.1 Project and solution . . . 171
8.3.2 Visualizations and discussion . . . 173
8.3.3 Python code . . . 174

A Glossary: GAN and Tabular Data Synthetization **183**

B Glossary: GenAI and LLMs **187**

C Introduction to Extreme LLM and Customized GPT **190**
C.1 Best practices . . . 190
C.2 Python utility for xLLM . . . 193
C.3 Comparing xLLM with standard LLMs . . . 199
C.4 Comparing xLLM5 with xLLM6 . . . 200
C.4.1 Conclusion . . . 201

Bibliography **203**

Index **205**