From Moldova to Inria in 5 easy steps

A hitchhiker's guide
made by
Sergiu MOCANU
“Spiru Haret” Lyceum
Applied Sciences & Humanitarian Sciences
School System Differences

Moldova & France
Alliance Française de Moldavie
Université de Rennes

Licence Info-Électro
Types of Binary Trees

Strict

Complete

Degenerate

Perfect

Computability Theory

Winfried Seig

6. Introduction

Computability is perhaps the most significant and distinctive sector of 20th-century logic that has been introduced in the guise of decidability and effective calculability in the context of logic within philosophy and mathematics. Since it is also the basic theoretical concept for computer science, artificial intelligence and cognitive science, this essay addresses, at the heart of methodological issues that are central to any understandability theory that is not only part of any physical or individual experience. The discussion is grounded in historical developments that are deeply intertwined with conceptual works in the taxonomy of methodologies. Here, it is that possible, the reader ought ask: when the essay is concerned only with computability? The introduction begins to give an answer by first describing the context of foundational investigations in logic and mathematics and then sketching the main lines of the systematic presentation.

6.1 Foundational contexts. In the second half of the 20th century, the issues of decidability and effective calculability were in the focus of discussions concerning the nature of mathematics. The division of these discussions is reflected
Program Synthesis

Using ChatGPT
Examples of Tasks

{"input": "BTR KRNL WK CORN 15Z", "output": "15Z"},
{"input": "CAMP DRY DBL NDL 3.6 OZ", "output": "3.6 OZ"},
{"input": "CHORE BOY HD SC SPNG 1 PK", "output": "1 PK"},
{"input": "FRENCH WORCESTERSHIRE 5 Z", "output": "5 Z"},
{"input": "O F TOMATO PASTE 6 OZ", "output": "6 OZ"},

{"input": "International Business Machines", "output": "IBM"},
{"input": "Principles Of Programming Languages", "output": "POPL"},
{"input": "International Conference on Software Engineering", "output": "ICSE"}
Prompt Engineering

5 Best Practices For Designing Effective Prompts

01 Assign Role
*Imagine you are a Business Development expert in Supply Chain with over 15 years of experience in this field.*

02 Styling Output
*Messages and email should be informative, should have vital data, case, and offer a clear call to action as well.*

03 Be Specific
*Imagine you would like to invite X person to discuss how our supply chain management services can help your business reduce costs and increase efficiency.*

04 Add Conditions
*You also must select the right use case and case study to be mentioned in emails and LinkedIn messages from the provided list.*

05 Provide Data
*Your company description is this "COMPANY DESCRIPTION" and your target client description is this "TARGET CLIENT DESCRIPTION" and your target client is working on "TARGET CLIENT'S USECASE.*

Different Language Models
BERT, RoBERTa, MiniLM, DPR, ERNIE, QFT, T5, Midjourney, PromptBase, DALL-E

Prompt Engineering

Prompt

Completion
Prompt Examples

"You're a programmer specialized in Python-3."

"You're a programmer specialized in Python-3. Give only the code."

"You're a programmer specialized in Python-3. Give only the code without any explanations."

"You're a Python-3 programming expert. Focus on producing code solutions. Omit any explanations."

"You're a Python-3 programming expert. Provide a function implementation that takes parameters and returns the desired output."
This year’s internship subject:

Program Synthesis Using Open LLMs
Internship Objectives:

● In-depth research of LLM architectures
● Prompt Engineering
● Model Fine-Tuning
● Metrics Analysis
● Dataset Construction
● Model Comparison
Three Main Tasks:

- Code Modernisation
- Code Migration
- Code Diversification
Modernisation:

Eventually with inclusion of good security practices
private static boolean checkCriterion(ArrayList<MyComplex> neighbors, double noiselevel, double ratio){
    double mean = neighbors.get(0).getAbs();
    for (int i=1; i<neighbors.size(); i++)
        mean += neighbors.get(i).getAbs();
    mean /= neighbors.size();

    if (mean > noiselevel){
        for (int i=0; i<neighbors.size(); i++)
            if (neighbors.get(i).getAbs() > (mean*ratio))
                return true;
    }
    return false;
}
return true;

# Example of avoiding complex expressions

def square_numbers(numbers):
    """Return a list of the squares of the numbers."""
    squares = []
    for number in numbers:
        squares.append(number**2)
    return squares

def main():
    numbers = [1, 2, 3, 4]
    result = square_numbers(numbers)
    print(result)
Diversification:

```java
class FibonacciRecursive {
    public static int fibonacci(int n) {
        if (n <= 1)
            return n;
        else
            return fibonacci(n - 1) + fibonacci(n - 2);
    }
    public static void main(String[] args) {
        int n = 10; // Change the value of n as needed
        System.out.println("Fibonacci Series up to "+n+" terms:");
        for (int i = 0; i < n; i++) {
            System.out.print(fibonacci(i) + " ");
        }
    }
}
```

```java
class FibonacciIterative {
    public static void fibonacci(int n) {
        int a = 0, b = 1;
        System.out.println("Fibonacci Series up to "+n+" terms:");
        for (int i = 0; i < n; i++) {
            System.out.print(a + " ");
            int temp = a;
            a = b;
            b = temp + b;
        }
    }
    public static void main(String[] args) {
        int n = 10; // Change the value of n as needed
        fibonacci(n);
    }
}
```