Best Practices In Prompt Engineering

João Gardelha
Micael Balza

Prof.Dr. Ivanovitch Silva

CASE STUDY

01 INFERRING

Inferring about abstracts

02 ITERATIVE CONSTRUCTION

Get a code iteratively

Why?

More efficient prompts can help us with research tasks.

How to do this?

ChatGPT Prompt Engineering for Developers - DeepLearning.AI

We employ the main techniques.

01

INFERRING

Inferring about abstracts

Parsifal Blog About Help micaelbalza

micaelbalza / Revisão sobre propostas de navegação com planejamento dinâmico em robos moveis terrestres usando Meta-heurística

Review settings

Review | Planning | Conducting | Reporting

1. Search | 2. Import Studies | 3. Study Selection | 4. Quality Assessment | 5. Data Extraction | 6. Data Analysis

Study Selection

All Sources | ACM Digital Library | EI Compendex | IEEE Digital Library | ISI Web of Science | Science@Direct | Scopus | Web of Science

Abstract

In this paper we present our novel Fuzzy-Genetic techniques for the online learning and adaptation of an intelligent robotic navigator system. Such a system could be used by autonomous mobile vehicles navigating in unstructured and changing environments. In this work we focus on the online learning of the obstacle avoidance behaviour, which is an example of a behaviour that receives delayed reinforcement. We show how this behaviour can be co-ordinated with other behaviours that receive immediate reinforcement (such as goal seeking and edge following) learnt during our previous work to generate an intelligent reactive navigator that can deal with

		operating in unstructured environment based on a novel online Fuzzy-Genetic system						
☐		Efficient path planning algorithm for mobile robot navigation with a local minima problem solving	Velagic, J. and Lacevic, B. and Osmic, N.	Proceedings of the IEEE International Conference on Industrial Technology	2006	micaelbalza	02 May 2023 21:02:22	Duplicated
☐		A new RBF neural network with GA-based fuzzy C-means clustering algorithm for SINS fault diagnosis	Liu, Z. and Chen, J. and Song, C.	2009 Chinese Control and Decision Conference, CCDC 2009	2009	micaelbalza	02 May 2023 21:02:22	Duplicated
☐		Application of fuzzy neural networks based on genetic algorithms in integrated navigation system	Liu, J.	2009 2nd International Conference on Intelligent Computing Technology and Automation, ICICTA 2009	2009	micaelbalza	02 May 2023 21:02:22	Duplicated

```python
# Read the Excel file and extract the 'abstract' and 'title' columns
data_frame = pd.read_excel(file_name)
abstract_column = data_frame['abstract']
title_column = data_frame['title']

# Iterate through each item in the 'abstract' column and display its content
```

```
Learning and adaptation of an intelligent mobile robot navigator operating in unstructured environment based on a novel online Fuzzy-Genetic system
Evolutionary approaches - Positive
Dynamic planning - Positive
Autonomous navigation - Positive
Unknown environment - Positive
Flying or aquatic robots - Positive

Efficient path planning algorithm for mobile robot navigation with a local minima problem solving
Evolutionary approaches - Positive
Dynamic planning - Negative
Autonomous navigation - Positive
Unknown environment - Positive
Flying or aquatic robots - Positive

A new RBF neural network with GA-based fuzzy C-means clustering algorithm for SINS fault diagnosis
Evolutionary approaches - Positive
Dynamic planning - Negative
Autonomous navigation - Negative
Unknown environment - Positive
Flying or aquatic robots - Positive
```

```python
    response.append(get_completion(prompt))

for r in response:
    index = response.index(r)
    print(title_column[index])
    print(r + "\n")
```

es

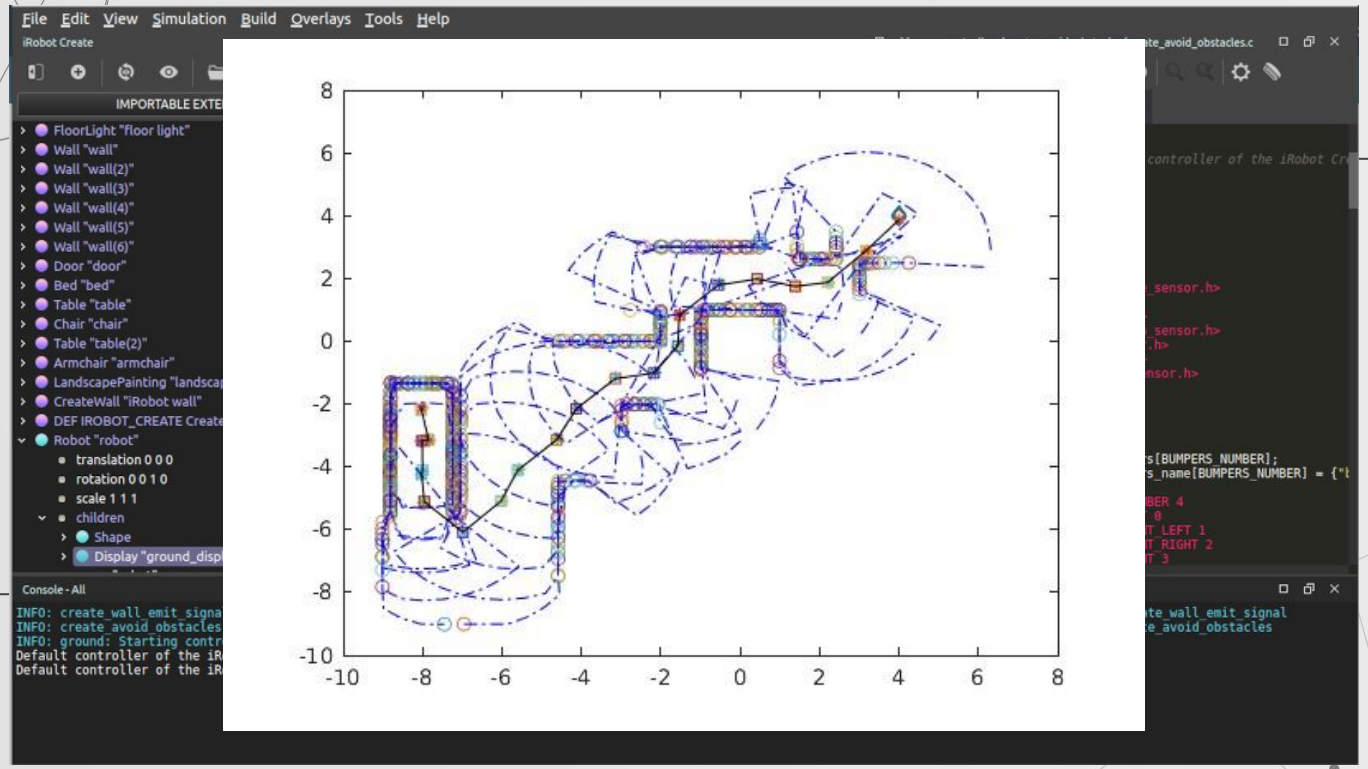
delimiters
ify the steps
e to think (context)
ctured output

https://colab.research.google.com/drive/1vWTEVlxl6-Y2r76ImuvHe8qGrGUWPIC3?usp=sharing

02

ITERATIVE CONSTRUCTION

Get a code iteratively


File Edit View Simulation Build Overlays Tools Help
iRobot Create
IMPORTABLE EXTE
FloorLight "floor light"
Wall "wall"
Wall "wall(2)"
Wall "wall(3)"
Wall "wall(4)"
Wall "wall(5)"
Wall "wall(6)"
Door "door"
Bed "bed"
Table "table"
Chair "chair"
Table "table(2)"
Armchair "armchair"
LandscapePainting "landscap
CreateWall "iRobot wall"
DEF IROBOT_CREATE Create
Robot "robot"
translation 0 0 0
rotation 0 0 1 0
scale 1 1 1
children
Shape
Display "ground_displ
Console - All
INFO: create_wall_emit_signa
INFO: create_avoid_obstacles
INFO: ground: Starting contr
Default controller of the iR
Default controller of the iR
8
6
4
2
0
-2
-4
-6
-8
-10
-10
-8
-6
-4
-2
0
2
4
6
8

ITERATIVE CONSTRUCTION

“Create a linked list in C language that saves x and y arrays with 36 points at each node.

Only functions of creating a list, adding to the end of the list, reading a list item, printing a list item, and deleting a complete list are required.”

Best Pratices

- Iterative construction
- Use delimiters
- Specify the steps
- Time to think (context)
- Structured output
- Specify output and input

VS

“I need a code in C that implements:
“A struct containing X and Y arrays with 36 points each”’ “

———————————————————————————————————

“Using a struct created, i need a linkedList where the struct is added to each node, for that I need the functions in quotes.

1 - Initialize the list by returning the pointer address:
’’’create_LinkedList()’’’

2 - Insert an item only at the end of the list:
’’’insert_LinkdList((&Delimiting_Polygon_LinkedList, polygon, &Delimiting_Polygon_LinkedList->m)’’’ “

———————————————————————————————————

“Now I need to add the functions of:

1 - Read an item from the list:
’’’read_LinkedList(&Delimiting_Polygon_LinkedList, Read_node)’’’

2 - Print an item from the list:
’’’print_linkedList(&Delimiting_Polygon_LinkedList,36)’’’

3 - Delete the list completely:
’’’delete_LinkedList(&Delimiting_Polygon_LinkedList)’’’ “

Results

```c
// Function to add a point to the end of the list
void addPoint(Point** head, int x[], int y[]) {
    Point* newPoint = createPoint();
    for (int i = 0; i < 36; i++) {
        newPoint->x[i] = x[i];
        newPoint->y[i] = y[i];
    }
    if (*head == NULL) {
        *head = newPoint;
    } else {
        Point* current = *head;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = newPoint;
    }
}
// Function to print all points in the list
void printList(Point* head) {
    Point* current = head;
    int count = 0;
    while (current != NULL) {
        printf("Node %d:\n", count);
        for (int i = 0; i < 36; i++) {
            printf("(%d, %d) ", current->x[i], current->y[i]);
        }
        printf("\n");
        current = current->next;
        count++;
    }
}
```

VS

```c
void insert_LinkedList(struct LinkedList* list, struct Point point) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->point = point;
    newNode->next = NULL;

    if (list->head == NULL) {
        list->head = newNode;
    } else {
        struct Node* current = list->head;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = newNode;
    }

    list->size++;
}

void print_LinkedList(struct LinkedList* list, int size) {
    struct Node* current = list->head;
    int i = 0;
    while (current != NULL && i < size) {
        printf("(%d, %d)\n", current->point.x, current->point.y);
        current = current->next;
        i++;
    }
}
```

Best Practices

Principles of prompting

Use delimiters:
Triple quotes: “ ” ”
Triple backticks: ‘ ‘ ‘
Triple dashes: - - -
Angle brackets : < >

Structured output:
C
Json
“ XX - A
YY - B ”

Check the conditions:
If all the information to perform the task is contained

Few-shot prompting:
Clear ≠ short

Give the model time to think

Specify the steps:
1 - …
2 - …

Iterative process:
First question …
Second question …

Understand your ability

Summarizing
Inferring
Transforming
Expanding