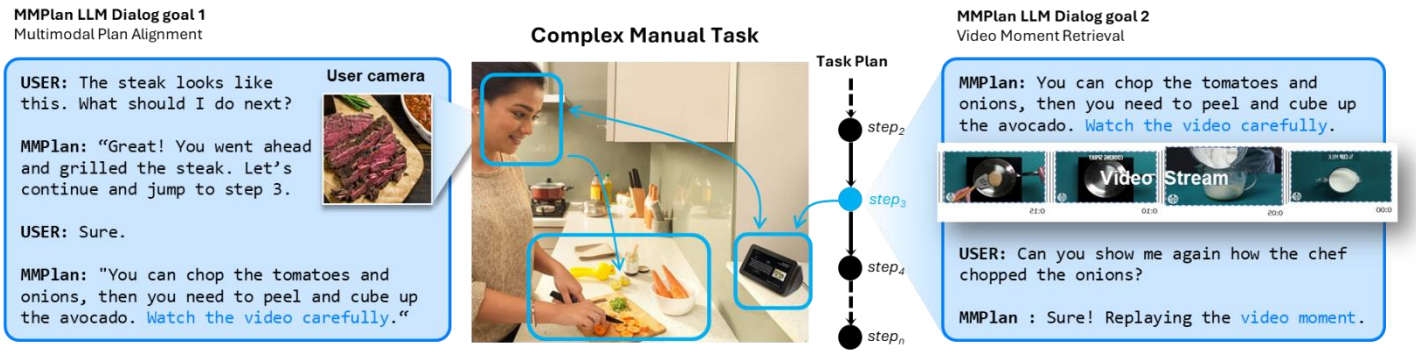
Conversational Task Agent



Web Search and Data Mining

Project Guide

João Magalhães
(jmag@fct.unl.pt)

This project guide will be updated with the available models and computational resources.

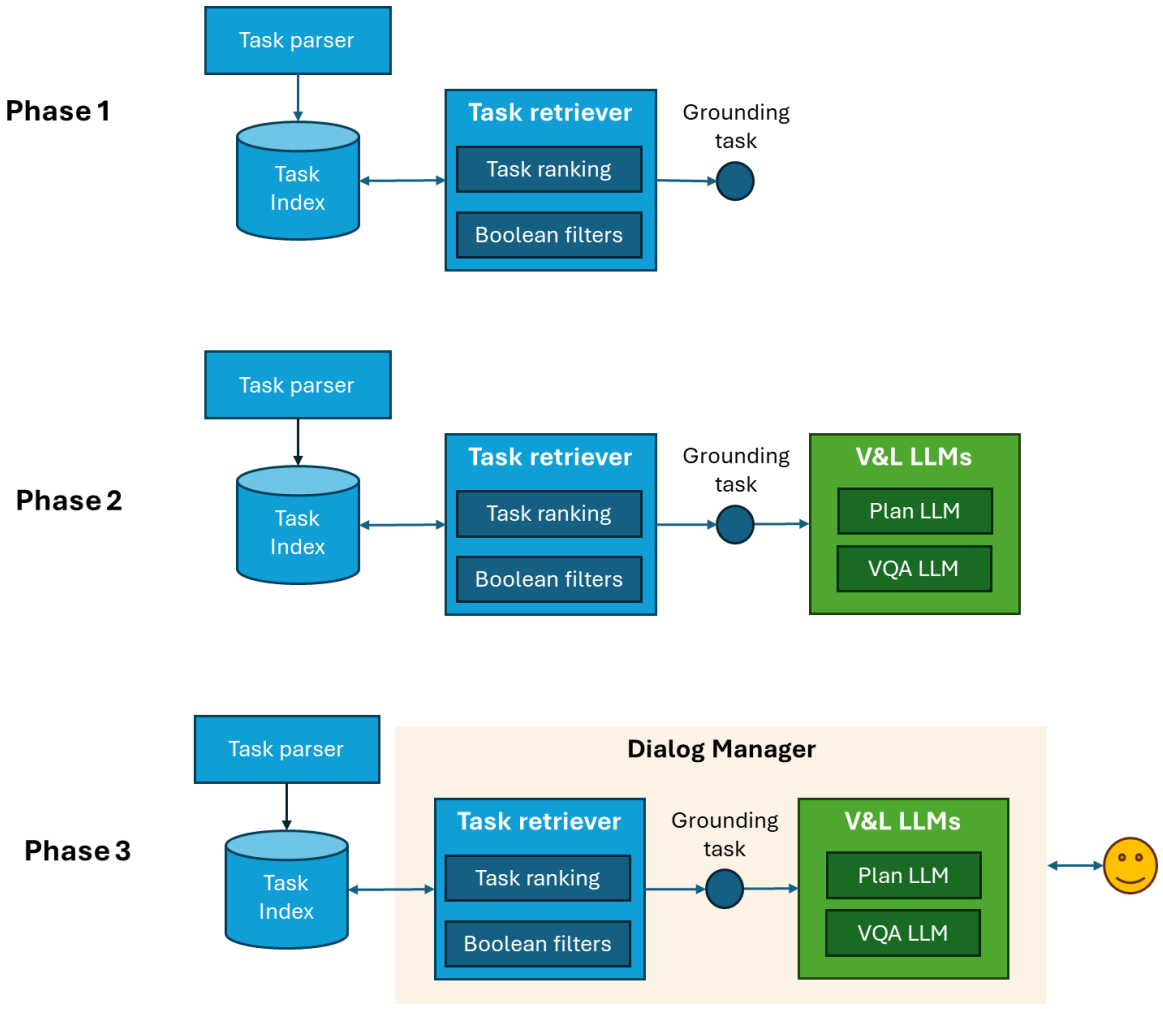
Universidade Nova de Lisboa
2023/2024

1 Introduction[1]

In this project you will build a "*taskbot that assist users in multi-step tasks, such as baking a birthday cake or fixing a scratch on a car — and adapt those instructions based on the resources and tools available to the customer. If, for example, a customer ran out of an ingredient halfway through a recipe or didn't have a specific tool for a DIY project, the taskbot had to adjust the plan and suggest possible solutions*."

The project will be divided into three parts:

- **Phase 1 (Task Retriever):** you will start by implementing a search index of complex manual tasks. This will allow your system to retrieve relevant tasks.
- **Phase 2 (V&L LLMs):** In the second phase, you will connect to the PlanLLM to answer and control the visual-language semantics.
- **Phase 3 (Dialog Manager):** In the final phase, you will implement an intent detector to understand what is the user objective and build a dialog manager based on the selected recipe.



[1] https://www.amazon.science/alexa-prize/taskbot-challenge/2022

2 Documents Indexing and Searching (15%)

Following a natural conversation, your final prototype will be able to engage in a simple conversation to ground the dialogue on a particular task. Like every human, your prototype will have its own KB of the different recipes, ingredients, task difficulty, and recipe steps.

In the first phase of the project, you will implement a Question-Answer search framework to answer Natural Questions. See the code examples provided during the laboratory classes.

2.1 Text-based search:

- Understand how to use the OpenSearch framework. Read the provided code and the documentation.
- Parse the JSON file containing the recipes and create the OpenSearch index mappings for the recipe title and description.
- Index the recipes and test the search functionality.
- You can try to optimize the search results.

2.2 Embeddings-based search

- Understand how to use the dual-encoders. Read the provided code and the documentation.
- Revise the index mappings to support k-nn vectors.
- Compute the recipe's embeddings and store them in a file.
- Add the embeddings to your index and test the search functionality (don't forget that you also need to compute the query embeddings).

2.3 Index mappings

Recipes are organized into different fields with text, keywords and integer data types. You should create your own set of mappings to index the recipes' data. The maximum set of mappings is the complete set of json fields, and a minimum set of mappings could be:

- recipe_id keyword
- title text
- title_embedding knn_vector
- ingredients keyword
- time integer

Both cases are sub-optimal and you should propose an optimal set of mappings that support your implementation functionalities.

2.4 Supported searches

In this phase you should implement different search methods to support searching with natural language questions and search with filters. This is intended search for recipes with specific traits, e.g. ingredients, servings, time, ratings, etc. Consider the following possibilities:

- Text based search, e.g., for keyword based search;
- Embeddings based search; e.g., for semantic search;
- Boolean filters alone, e.g., search by ingredients;
- Search with boolean filters, e.g. a combination of search methods;

2.5 Tips for persistent storage:

- Some tasks take a long time to process, e.g. computing the embeddings, hence, you may wish to put the embeddings in some persistent storage.
- Python pickles allow you to (de)serialize objects: https://docs.python.org/3/library/pickle.html
- HDF5 or Pandas DataFrames provide you a more structured solution you can use.
- If you wish to store JSON objects, you can do so in the index, however, remember that whenever you delete the index you will lose this data.

3 Large Language and Vision Models (20%)

Visual cues are an important aspect for human reasoning and decision processes. Visual information analysis algorithms allow the conversational assistant to provide visual depictions of a recipe and ingredients and to include the visual modality in the conversation.

3.1 Implementation

3.1.1 Cross-Modal Retrieval

The first objective for this phase of the project is to index all images of all tasks and support cross-modal retrieval, i.e. retrieve text with image queries, retrieve images with text queries, and retrieve images with image queries. To implement this functionality, you will use a CLIP encoder. CLIP allows you to compute the semantic similarity between a sentence and an image.

CLIP has a dual encoder architecture: the image encoder generates an embedding vector for images, and the text encoder generates an embedding vector for text. Both representations exist in the same embedding space and can be used to compute the

similarity text and image data in the visual space, i.e., text is always assumed to describe the contents of the image.

This will allow you to do a number of operations, e.g. compute the similarity between step A image and step B text, the similarity between user image A and all steps in the recipes, etc.

3.1.2 Large Language Model: PlanLLM

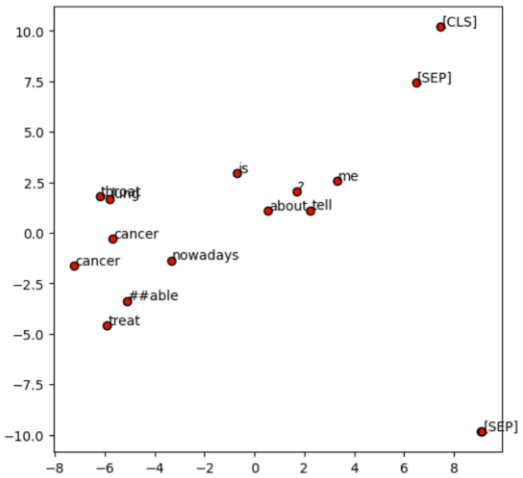
The second objective for this phase of the project is to follow the task plan and to support user requests and questions about the selected task. You will use a specialized LLM for following instructional manual tasks:

- **PlanLLM:** https://huggingface.co/NOVA-vision-language/PlanLLM

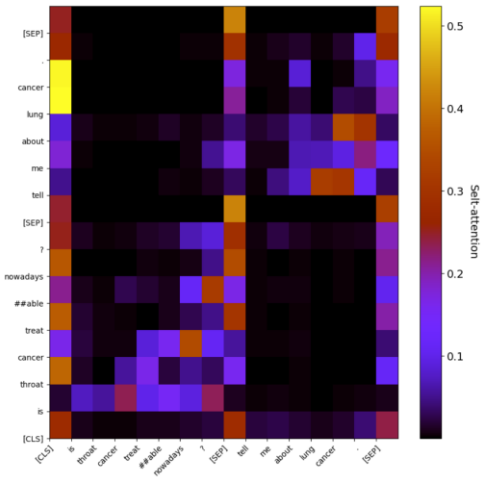
Analyse and study the provided code carefully and integrate this LLM in your project.

3.2 Contextual embeddings and self-attention

In this phase of the project you will show your understanding of the self-attention mechanism for vision-language web related tasks. You will also demonstrate your understanding of the Transformer architecture for different tasks.



Word embeddings visualization.



Single-head self-attention visualization.

You should discuss the following 5 points:

1. **Contextual embeddings.** Visualize the contextual word embeddings from layer 0 to layer 11. Observe how they change from layer to layer.
2. **Positional Embeddings.** Consider a simple BERT encoder. Insert a sequence of text with the same word repeated 20 times. Visualize and examine the embeddings and discuss what you observed.

3. **Sentence embeddings.** Consider the sentence embeddings algorithms. Examine the provided implementation (Lab 2) and explain the sentence embeddings in terms of output embeddings.

4. **Self-Attention.** Examine the self-attention mechanism of a transformer cross-encoder. Repeat with a dual encoder. Do critical analysis of your observations.

To address the above questions you should build on the provided tutorials to visualize word embeddings and self-attention matrices. See the two examples above.

4 Dialog manager

In the third phase you should use the all the available models and the provided intent detector to implement a simple Dialog Manager.

4.1 Dialog intent detector

4.2 The intent detector will detect the following intents: (1) Greetings, (2) Search recipe, (3) Out of scope, (4) Yes, (5) No, (6) Start task, (7) Next, (8) Stop (and others that you may use, ignore or merge). As a simple baseline you can use a dialog act detector as a general intent detector.

4.3 Dialog Manager (10%)

The behavior of the Dialog Manager should support conversations as the example below.

Task grounding:

TaskBot: Hi there, I am the Alexa Prize TaskBot. I can help you with a Cooking or Home Improvement Task, do you want to continue?

User: Sure, I would like to bake a cake.

TaskBot: Great! I would love to help you with that! Do you have a flavor in mind?

User: Chocolate, but I am not sure that it can be healthy.

TaskBot: I know lots of recipes for chocolate cake. Here are a few on the screen. Do any of them look good to you?

User: The third one looks good.

Task execution:

TaskBot: Alright, lets proceed! First, lets make the coconut pecan filling: please see the steps on the screen. Let me know if you would like me to read the steps, or if you have any questions.

User: Next step.

. . . . .

4.4 Group specific implementations (15%)

For the third phase you need to implement a part that is unique to your project as a group or individual. The proposed idea should be discussed with the lab instructor or selected from the following list of ideas:

Search/browsing:

1. Improved recipes search by fine-tuning sentence-transformers.
2. Improved recipes search with preferences (NLU + OpenSearch filters).
3. Recipes browsing by similarity.
4. Billion scale distributed search (OpenSearch).

Conversational agent:

5. Dialog manager with mixed initiative.
6. Dialog manager with improved intent detector.
7. Dialog manager with VQA fine-tuned to the domain (ViLT).
8. Dialog manager with support for chit-chat about recipes (DialogGPT / BART).

V&L Explainability:

9. VilT model explainability.
10. CLIP model explainability.

Gamification:

11. Gamification of recipes search.
12. Game: Guess the recipe title.

Literature reviews (min 6 pages, 15 references):

13. Energy and pollution concerns of large vision and language models
14. Ethical and moral risks of large vision and language models

Your own idea may also be suitable as the project independent part!!!

Propose it until May 18.

6 Project Grading, Submissions and Report

The progress accomplished on each phase should be submitted and detailed in a report.

- Phase 1: April 8 15%
- Phase 2: May 6 20% -> May 9
- Phase 3: June 3 10%
- Originality June 3 15%

The final report is limited to 12 pages written during the three phases. A suggestion is to write 4 pages on each phase. When you submit the report of each phase, you are allowed to update the text of the previous phase.

The suggested template is available on Overleaf. The suggested structure is as follows:

1. Introduction (Phase 1,2,3)
2. Algorithms and Implementation (Phase 1,2,3)
   - 2.1. Search and Indexing
   - 2.2. Large Vision and Language Models
   - 2.3. Dialog Manager
3. Evaluation
   - 3.1. Dataset description (Phase 1, 2)
   - 3.2. Baselines (Phase 3)
   - 3.3. Results analysis (Phase 3)
4. Critical discussion (Phase 3)
5. References

7 Readings

SentenceBERT: https://arxiv.org/abs/1908.10084

TWIZ: https://arxiv.org/abs/2310.02118

PlanLLM: https://aclanthology.org/2024.eacl-long.77/

VILT: https://github.com/dandelin/ViLT

InstructBLIP: https://github.com/salesforce/LAVIS/tree/main/projects/instructblip