Conversational Task Agent

Web Search and Data Mining
Project Guide

João Magalhães
(jmag@fct.unl.pt)

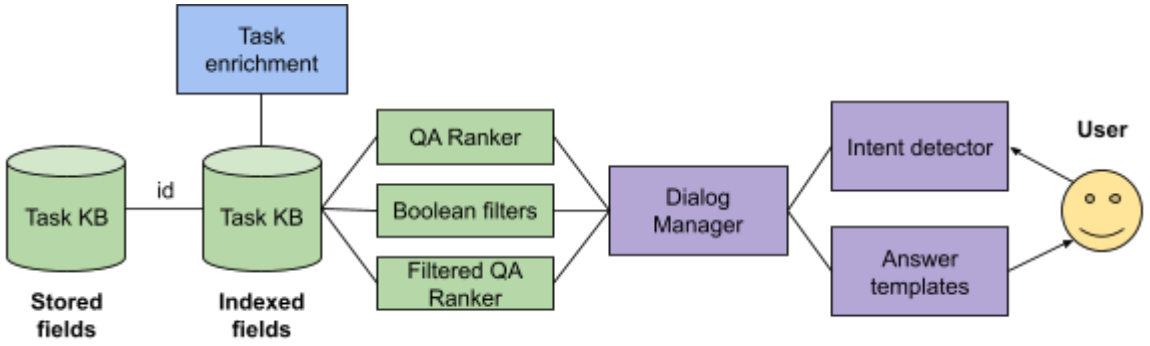
Universidade Nova de Lisboa
2021/2022

Introduction

In modern societies, there are many daily tasks that can be supported by conversational agents -- in upcoming years, it is expected a larger shift from the traditional search-and-browse paradigm to the conversational paradigm. The WSDM project is positioned within this context. You will implement a conversational assistant that will find the task that the user wishes to execute and will guide the user throughout the task.

The project will be divided into three parts, see the figure below:

- **Phase 1 (green part):** you will start by implementing a searchable index of a recipes Knowledge Base. This will allow your system to solve Question-Answer retrieval tasks.
- **Phase 2 (blue part):** In the second phase, you will enrich the KB by linking recipe images to other recipe steps that share the same visual-language semantics.
- **Phase 3 (purple part):** 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.



Phase 1: Answering natural questions (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 here:

https://wiki.novasearch.org/wiki/courses/WSDM_2022

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.

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).

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. A possible set of index mappings is:

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

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
- embeddings based search
- boolean filters
- search with boolean filters

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.

Phase 2: Multimodal tasks KB (20%)

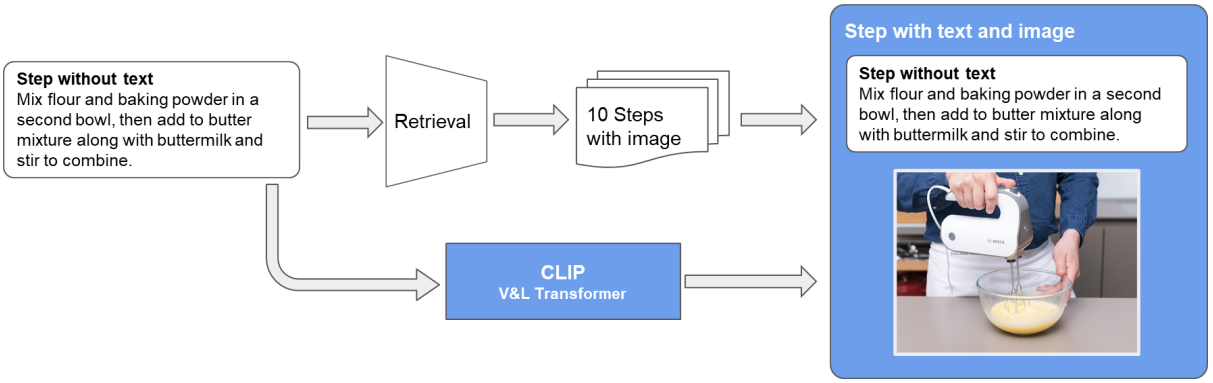
Visual cues are an important aspect for human reasoning and decision processes. Visual information analysis algorithms allow the conversational assistant to provide visual depictions of a recipe and ingredients and to include the visual modality in the conversation.

Recipes' illustration

In this phase of the project you will illustrate all steps of all recipes with images from other steps and/or recipes. To implement this functionality, you have two possible tools:

- CLIP allows you to compute the semantic similarity between a sentence and an image. For example, the similarity between step A image and step B text.
- Sentence transformers allows you to compute the semantic similarity between two sentences. For example the similarity between step A text and step B text.

Both methods can be used to assign an image from a step to another step that has no illustration images.



Implement your own solution. Do a failure analysis and propose a future solution to mitigate critical failures.

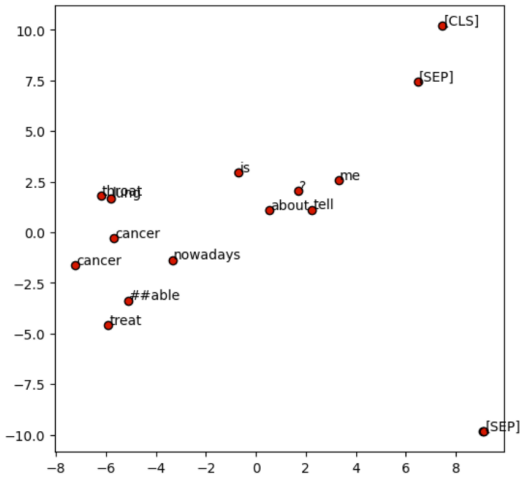
Contextual embeddings and self-attention

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.

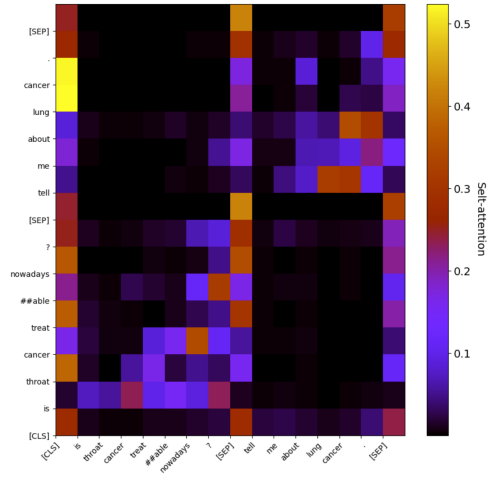
You should discuss the following 5 points:

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 algorithms. Examine the provided implementation 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.
5. **Vision and Language Self-Attention.** Visualize the self-attention with vision and language data. Do a 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. Below you can see two examples.



Word embeddings visualization.



Single-head self-attention visualization.

Phase 3: Dialog manager

In the third phase you should use the Multimodal Tasks KB and the intent detector to implement a simple Dialog Manager. You must also implement an group specific functionality.

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 visualization:

TaskBot: Ok, Lets take a look at that one. Check out on your screen "German Chocolate Cake" by Paleo Running Mama Blog. Would you like to see the ingredients?

User: Sure

TaskBot: Ok, here's the list of ingredients you will need on your screen.

User: Great! I have everything. Let's start the recipe.

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.

. . . . .

Dialog intent detector

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

Group specific implementations (15%)

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

1. Improved recipes search: Training the Neural Ranker
2. Improved intent detector: Training the Neural Ranker
3. Dialog manager with mixed initiative
4. Billion scale setup
5. Dialog manager with support for chit-chat about recipes
6. Dialog manager with VQA fine-tuned to the domain

Project Grading, Submissions and Report

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

- Phase 1: April 8th 15%
- Phase 2: May 6th 20%
- Phase 3: June 3rd 10%
    - +15% for the independent part of the project as demonstrated in the presentation.

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)
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

Course Grading

The WSDM course grading is as follow:

- 45% Project
- 15% Independent part of project as demonstrated in the presentation
- 40% Exam/test/project discussion