Castiglia Giovanni, Calò Federica

MILESTONE 1

Motivation

We want to develop a conversational model for food recommendations that allows multi-modal and natural user-system interaction. It also leads users towards healthier choices using recommendation explanations.
For this purpose, we would like to propose a pure textual (T) and multi-modal (MM) user-system interaction modality: the last one will consist in the communication by displaying the name and image of each item (dish on the menu) throughout the dialogue.

In this first milestone we propose the basic structure of our chatbot, developed by using the *python-telegram-bot* library. We also propose our dataset and describe its structure.

In further milestones, we will continue increasing the system complexity and providing all system states which compose the user system interaction aimed at the recommendation process (previously described in project acceptance document).

We will also conduct a simple user study and use acquired user preferences and data for analysis.

Dataset generation and structure

To reach food data, we started from the Allrecipes.com dataset, which was obtained in July 2015 by implementing a standard web crawler. It contains 60983 recipes published between the years 2000 and 2015 on the popular Allrecipes.com website. In particular, 58263 out of these recipes contain basic nutritional details to assess the healthiness of recipes.

We accessed the dataset for educational scopes under EULA (End User License Agreement) agreement and we extracted 2000 recipes.

We decided to consider four food categories:

- pasta (500 dishes)
- salad (500 dishes)
- dessert (500 dishes)
- snack (500 dishes)

The acquired information helped us creating a simpler dataset (composed of 4 smaller datasets), containing dishes characterized by the attributes reported here:

Field name	Description
id	URL of dish recipe on Allrecipes.com
category	Dish category in [Pasta, Salad, Dessert, Snack]
image_url	Dish image URL on Allrecipes.com
name	Dish recipe name
energy_Kcal	Calories of dish (Kcal)
energy_Kj	Calories of dish (Kj)
ingredients	List of ingredients of the dish
servings	Servings of the recipe
proteins	Proteins in the dish (g)
carbohydrates	Carbohydrates in the dish (g)
fibers	Fibers in the dish (g)
sugar	Sugar in the dish (g)
fats	Fats in the dish (g)
saturates	Saturates in the dish (g)
sodium	Sodium in the dish (g)
salt	Salt in the dish (g)
tot_grams_weight	Total weight of the dish (g)
serving_size	Weight of the serving (g)
proteins_100_grams	Proteins in 100 g of dish (g)
carbohydrates_100_grams	Carbohydrates in g of the dish (g)
fibers_100_grams	Fibers in 100 g of dish (g)
sugar_100_grams	Sugar in 100 g of dish (g)
fats_100_grams	Fats in 100 g of dish (g)

saturates_100_grams	Saturates in 100 g of dish (g)
sodium_100_grams	Sodium in 100 g of dish (g)
salt_100_grams	Salt in 100 g of dish (g)
nutri_score	Computed nutriscore
fsa_score	Computed fsa_score

Chatbot initial structure

To build our chatbot, we are using the *python-telegram-bot* library, which provides a pure python, asynchronous interface for the Telegram Bot API. It contains several classes that make writing conversation flow simple.

In order to run a conversational chatbot on Telegram, we have created a bot called FoodBot (**@mlfoodbot**) and obtained an Access Token, by using the BotFather standard bot.

One peculiarity of the *python-telegram-bot* library is the ease of defining the states of the conversation flow: it is possible to handle the messages received in each step and process user data during the interaction. Each state of the flow is represented by an identifier (return code), which is used as a pointer to next states. In each state it is possible to define different types of handler functions (to handle messages or commands) which take user input and pass it to specific user-designed callback functions.

The core component for designing a python-telegram-bot system are:

- Updater: this function takes the Access Token received from BotFather to specify the bot to which we are adding functionalities with the help of the python code.
- Update: this function helps us invoke a bot every time it receives an update, i.e., message or command, and send a message to the user.
- CommandHandler: this handler is used to handle any command sent by the user to the bot. The commands must begin with “/”. For example: “/start”, “/stop”.
- MessageHandler: this handler is used to handle any normal message sent by the user to the bot.

The main progresses in the project Milestone 1 are reported below:

- ☐ We have created a python class for representing the dishes on the menu. It is called Piatto and contains all previously mentioned attributes.
- ☐ We have created a global list representing the menu. It contains 2000 dishes, 500 per category (pasta, salad, dessert, snack). It is accessible globally by each user.
- ☐ To handle every data used during **each user** session, we have created a global dictionary: for each key representing a user (though its Telegram user id) we associate a dictionary of variables (e.g. for the user with id 123456 we have an entry containing a dictionary containing, for example, these fields: *startSessionDate=01 Jan 2015 04:59:59, userChoiceFoodCategory=Pasta*).
- ☐ We have created and loaded dataset in order to fill the menu of Piatto objects
- ☐ We have defined the behavior of the chatbot at the /start command: the chatbot greets the user and asks him about his food category of preference. Next step will set the user-system interaction mode (between T and MM)

Project work stakeholders

The students Castiglia and Calò are equally working on the project (code development, testing, documentation drafting) during daily meetings.

Project work GitHub repository

The repository of project work is private. Please tell us how to share it with you. Anyway, the code of Milestone 1 will be attached to this documentation.

Castiglia Giovanni, Calò Federica

MILESTONE 2

Motivation

In this second milestone we developed the constraints acquisition step, we provided translating functions (useful since we want italian speaking users to join our small user study) and we built a diseases and intolerances dataset which contains ingredients to avoid. In further milestones, we will continue with the processing steps towards recommendation one.

Diseases and intolerances dataset generation and structure

In order to filter the menu according to user constraints, we build a dataset containing for each well-known intolerance/disease a list of ingredients to avoid. This will help us remove dishes which have dangerous ingredients for a certain user. We started from the diseased and manually scan dishes from the menu and collected ingredients in the dataset. We also looked at online medical resources to investigate diseases and intolerances.

The simple dataset we created is composed by the attributes reported here:

Field name	Description
name	Name of the disease/intolerance
type	Disease or intolerance flag
ingredients_to_avoid	List of ingredients to avoid

An example row is reported here:
"**Alcohol**","**Intolerance**","sherry,vermouth,vodka,cognac,white wine,wine,red wine,beer,liquor,liqueur,whiskey,rum,peach schnapps,apricot brandy,bourbon,triple sec".

Interaction modality selection

We provided a section of the code where it is possible to set the interaction modality between pure textual (T) and multi-modal (MM).

Content translations

To allow italian speaking users to understand names and ingredients of dishes from the menu, we build two simple translating functions using the *googletrans* python library. This library allows translations from a source language to a destination language using a Translator object.

Constraints acquisition step

In this step, the system needs to acquire dietary constraints of the user in order to filter the menu according to user needs. This helps us prevent the recommendation of dishes that may cause damage to users. This operation consists of two main steps:

1. the user is asked to tap on a grid of well-known diseases and intolerances if she has any

2. the user is asked to write a list of ingredients she wants to avoid (in this case we avoid the possibility of recommending items that user dislikes/disgusts)

The core component used for designing the constraints acquisition phase are:

- InlineKeyboardButton: this object represents one button of an inline keyboard. We pass to each button a callback data useful to handle events on that specific button.
- CallbackQueryhandler: this handler is used to handle Telegram callback queries from callback buttons in an inline keyboard.

We created a grid of intolerances by using the InlineKeyboard feature of the *python-telegram-bot* library, by filling an array of InlineKeybordButtons. The click on each button calls a function which changes the button content and adds a tick symbol. After this sub-step, we asked the user to write a list of ingredients separated by comma.

Then, we created a list of constraints that will be used in further steps to filter the menu.

The main progresses in the project Milestone 2 are reported below:

- ☐ We have created a dataset of intolerances and diseases and their ingredients to avoid
- ☐ We have developed the second step of our conversation flow, i.e. the constraints acquisition step.
- ☐ We have introduced two function for content translations from english to italian and vice versa. We will have a textual processing that will work with english language (since the main dataset of dishes is written in english).
- ☐ We used callback query handle functions to handle inline keyboard buttons in order to show different states of the buttons (user clicks on a disease and the system lets her know about the present state)

Project work GitHub repository

The repository of project work is private. Please tell us how to share it with you. Anyway, the code of Milestone 1 and Milestone 2 will be attached to this documentation.

Castiglia Giovanni, Calò Federica

MILESTONE 3

Motivation

In this third milestone we filtered the menu according to user constraints (well-known diseases or user-typed ingredients). After that, we developed the preference elicitation step using the random Active Learning strategy. In further milestones, we will continue by building the user profile and looking at the representation of items. Then we will process recommendation based on similarity between user profile and items.

Menu filtering according to user constraints

In order to filter the menu according to user constraints, we used the list of diseases and ingredients provided by the user. We used the knowledge of the dataset of diseases and intolerances to remove from the menu all dishes (items) containing ingredients (features) that the user cannot eat due to her health problems. Then, we remove all dishes containing ingredients that the user has explicitly defined as not desired. In this way we avoid users to rate dishes that are dangerous according to their health problems. We also implemented a similarity function that allows the system to understand ingredients to avoid also when they are written by a user with

typos. The new filtered menu is stored for each user in the dictionary contained in the entry related to that user. The whole menu remains available to all users as a global variable.

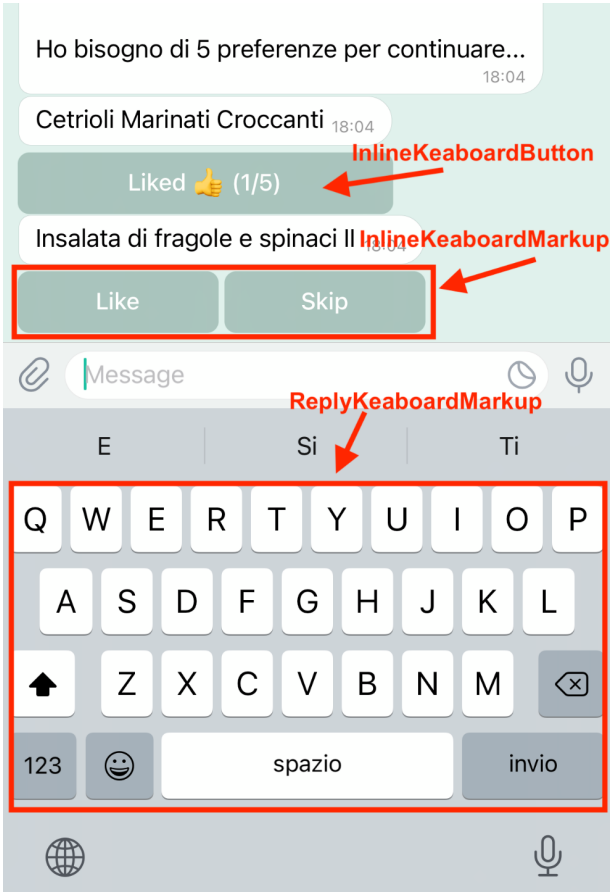
Preference elicitation

In this step, for each user, we randomly shuffle her filtered menu and we administer dishes in order to acquire user preferences. The system proposes one dish per time. Each dish is accompanied by two buttons: “Like” and “Skip”. The first one lets the user express a preference over the menu item, the second one tells the system to propose a new item. The preference elicitation process ends when the system gets 5 preferences. By using preference elicitation with Active Learning strategies like the random one, we face the problem of cold-start, typical in content based recommender systems like ours. During the item presentation step, we translate item names to allow italian speaking users to understand the content. In the multi-modal version of the system, we administer dishes by presenting their names and images at the same time. We believe that this interaction approach helps users better focus on their preferences.

To handle each item status (liked by user) we used inline keyboard buttons (InlineKeyboardButton) which have been handled using callback query functions. We are able to change the status (i.e. turn a Like button into a Liked (x/5) button) by identifying keyboards (inlineKeyboardMarkup) using personalized IDs and by changing their buttons' style and content.

When we want to send a text or photo message to users, we use the update.message.reply_text and update.message.reply_photo procedures.

We handle possible error conditions like a limited number of menu items after filtering.



The main progresses in the project Milestone 3 are reported below:

- ☐ We have filtered the menu for each user according to her constraints (well-known health problems, specified ingredients to avoid)
- ☐ We have designed and developed the preference elicitation step using Active Learning random strategy.
- ☐ We have handled some error conditions (e.g. end of menu for a certain user).

Project work GitHub repository

The repository of project work is private. Please tell us how to share it with you. Anyway, the code of Milestone 1, Milestone 2 and Milestone 3 will be attached to this documentation.

Castiglia Giovanni, Calò Federica

MILESTONE 4

Motivation

In this fourth milestone we designed the item representation using TF-IDF vectorization, we preprocessed TF-IDF vectors for computational reasons, we designed and build the user profile and we generated a first attempt of recommendation list based on the similarity of items (dishes of the menu) and user profile.

Items representation: TF-IDF vectorization

In order to represent items and their features, we decided to use the vector representation using TF-IDF (not normalized). Each dish has been represented by a vector of ingredients, represented by their feature score which represents the importance of that ingredient in that dish according to other dishes in the collection (and their ingredients). The TF-IDF score is performed as:

$$\textbf{tf-idf}(t_k, d_j) = \textbf{tf}(t_k, d_j) * \textbf{idf}(t_k)$$

$$\textbf{tf}(t_k,d_j) = f_{kj} / f_{zj}$$
$$\textbf{idf}(t_k) = \log(N / n_{tk})$$

with
z = most frequent term in doc. d_j
f_{kj} = freq. of term k in doc. d_j
f_{zj} = freq. of term z in doc. d_j
N = number of docs in the collection
n_{tk} = number of documents that contains tk

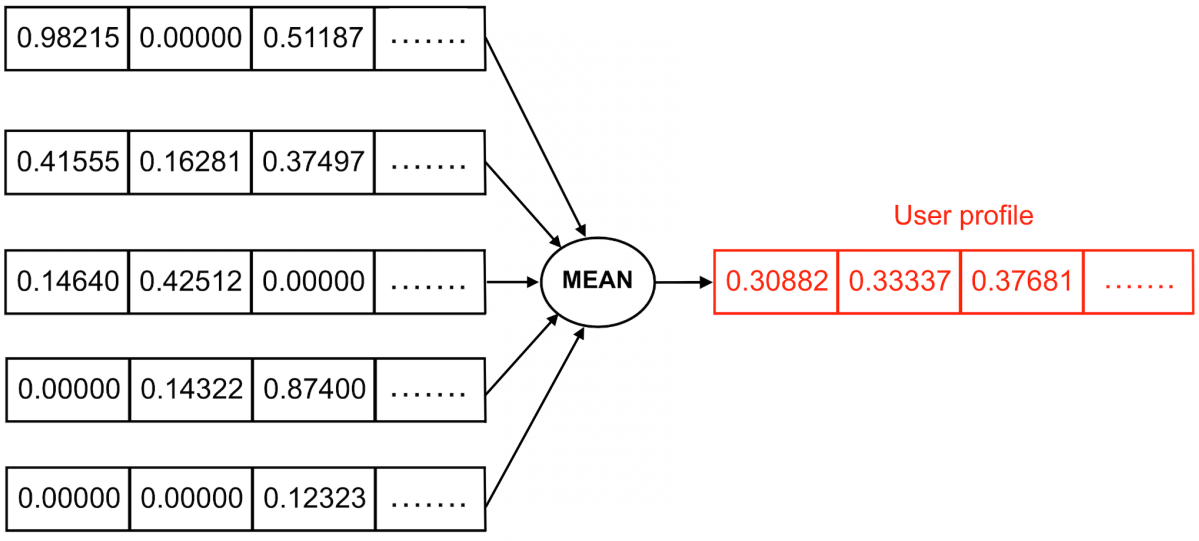
We used for this task the *TfIdfVectorizer* module of the *scikit-learn* library.
We preprocessed the TF-IDF vectors for each category of dish, in order to avoid that computation during the user-system conversation. This has been done by using the transform() method of the TfIdfVectorizer class. Then, we generated a dense representation of the output matrix using todense() python function. The system uses the matrix of vectors in order to compute similarity among dishes. We also stored names of features and names of dishes in external files.

We report the number of the dish features for each dish category after vectorization:

Dish item	Number of features
Pasta	317
Salad	287
Dessert	181
Snack	321

User profile building: a new vector compatible with menu items

In order to represent user preferences, we build each user profile by computing the mean of the vectors of dishes liked by each user. In this way, the score associated with each ingredient is the result of the average of scores of the vector representation of the dishes liked by the user (5 in optimal cases). A representation of this process is reported below:



TF-IDF vectors of five preferred dishes

The resulting vector has the same number of features of dish items of the menu for the category selected by the user. In this way it is easy to compare vectors of dishes with the user profile. This process is discussed in the next section.

Cosine Similarity: a way to compare user profile and menu items

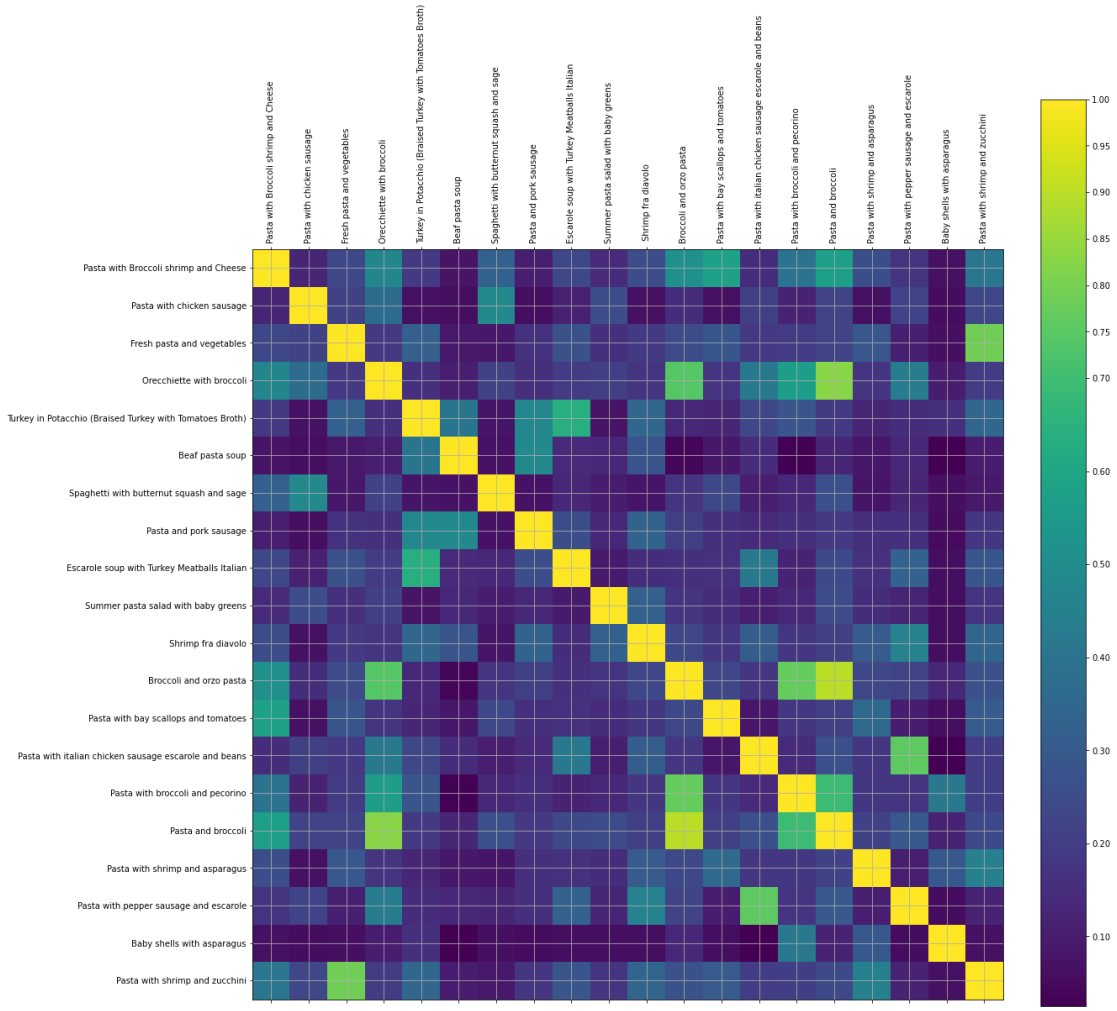
In order to compare the user profile with dish items to produce a recommendation list, we chose the Cosine similarity metric. It is computed as the normalized dot product of two vectors X and Y:

cosine_similarity(X, Y) = <X, Y> / (||X||*||Y||)

We performed the cosine similarity score among each pair of dishes of each category. To do so, we used the *cosine_similarity* module of *scikit-learn* library.

So, for each food category, we append the user profile vector to the TF-IDF matrix containing item representations and we perform cosine similarity. We obtain a new matrix of cosine scores representing the similarity evaluation between two corresponding items (by looking at their ingredients scores).
The last row of this matrix represents the similarity scores of each dish of the menu with the user profile. We extracted this last row and removed its last element (1, referring to the similarity score performed between user profile with itself).
Then, we matched each dish item with its similarity score with the user profile (using zip python function)
If we sort the menu according to each dish's cosine similarity score, we will obtain a recommendation list.

A solely representative cosine similarity matrix for 20 pasta category dishes is presented below:



The main progresses in the project Milestone 4 are reported below:

- ☐ We designed the item representation using TF-IDF vectorization and preprocessed TF-IDF vectors
- ☐ We designed and build the user profile and generated a first attempt of recommendation list based on the similarity of items (dishes of the menu) and user profile.

Project work GitHub repository

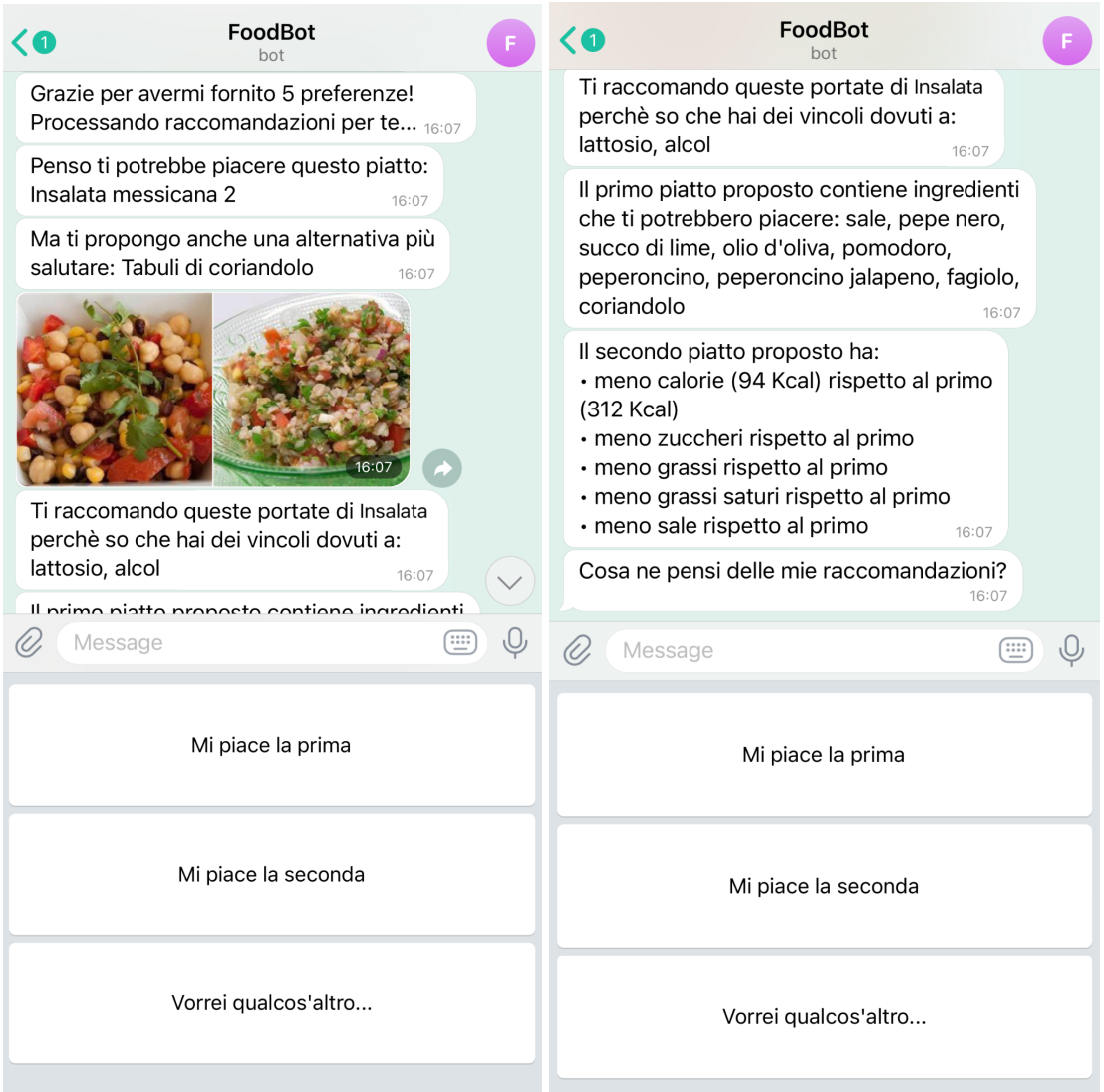
The repository of project work is private. Please tell us how to share it with you. Anyway, the code of Milestone 1, Milestone 2, Milestone 3 and Milestone 4 will be attached to this documentation.

Castiglia Giovanni, Calò Federica

MILESTONE 5

Motivation

In this fifth milestone we processed the recommendations to be shown to users. We started from sorting the recommendation list by decreasing cosine similarity of items with the user profile. Then we created a new list by re-ranking the recommendation list according to each dish's preprocessed FSA score, representing the healthiness of a recipe.
In this way we were able to recommend an element of the recommendation list and retrieve a healthier alternative, which is the most similar and healthier one. We handled a process that allows the user to ask for new comparisons until the preferred dish is found. We finally asked the user to rate the platform and we acquired useful information for our user study.



Recommendation lists generation

In order to generate the recommendation list, we sorted the menu according to decreasing cosine similarity with the user profile. To do so, we have attached to each dish the score of similarity (using the zip function) and we sorted the list looking at the score. To do so, we used the python sort procedure by setting the key parameter to the cosine similarity score using a lambda function and by setting the reverse parameter to true in order to have a list sorted by decreasing order: dishes that are more similar to the user profile have higher position in the list.
Then, we generate a new healthy list, by sorting the recommendation list according to each dish's FSA score. This metric is commonly used in the culinary recommendation scenario and helps evaluate the healthiness of a dish. We got this score from the original dataset.

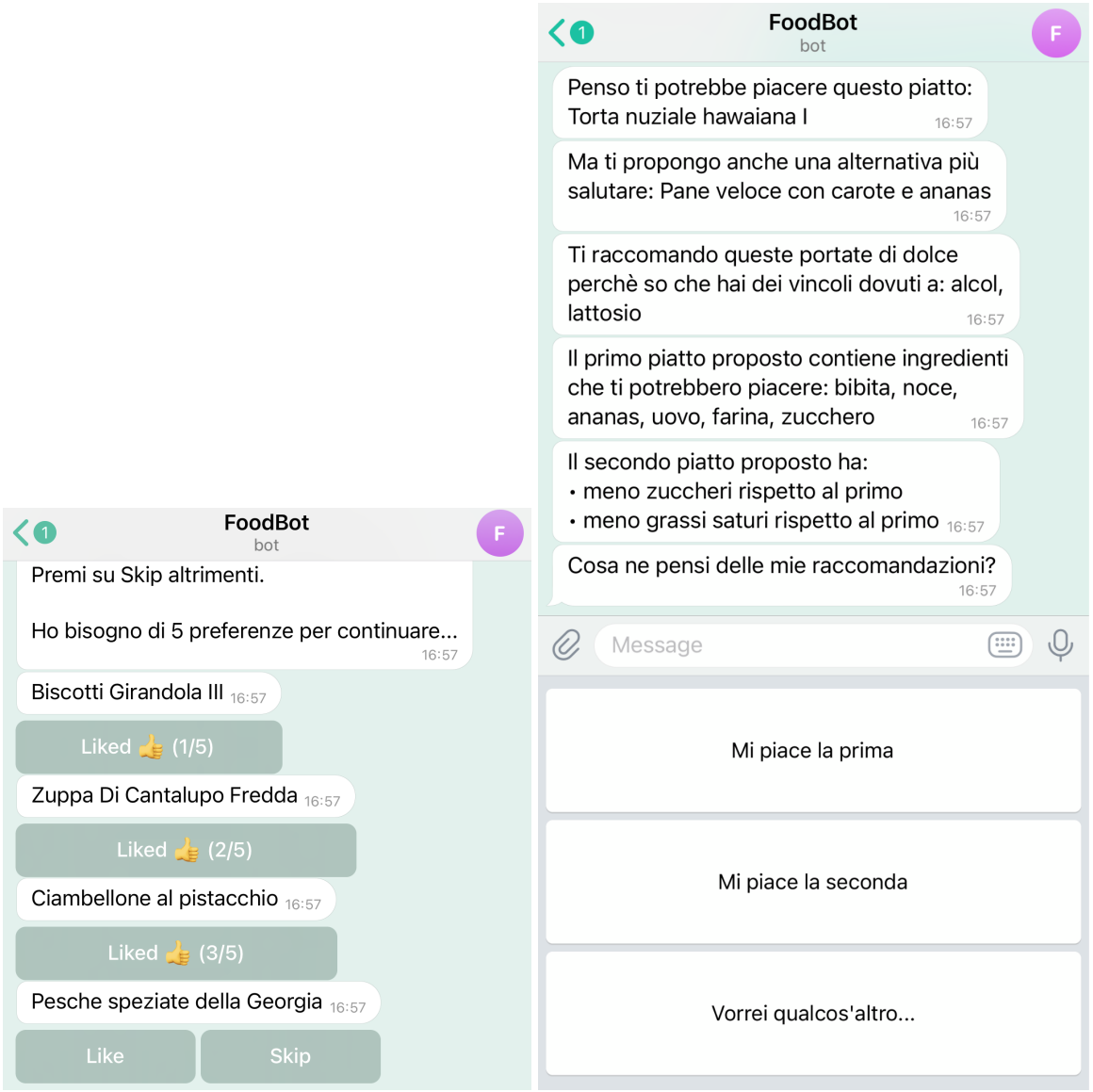
Recommendation and explanation phase

In this step, we extracted the first element of the recommendation list and we performed calculations to find the most similar and healthiest alternative. To do so, we got the item which has the higher ranking in both lists: the first, the recommendation list contains dishes sorted according to decreasing cosine similarity with the user profile, while the second one, the healthy list contains dishes sorted according to decreasing healthiness. We chose the best element by looking at the item with minimum value of:

rank_score = 0.4 * rank_rec_list + 0.6 * rank_health_list

We gave major importance to the healthiness of the alternative (60% of the score importance). We extracted the item with the minimum rank_score and presented it as a healthier alternative to the user.

In the textual modality (T), we show to user only textual names of the items and we provide a textual recommendation explanation. Instead, in the multi-modal modality (MM), we propose a pairwise comparison of dishes images and textual explanations.



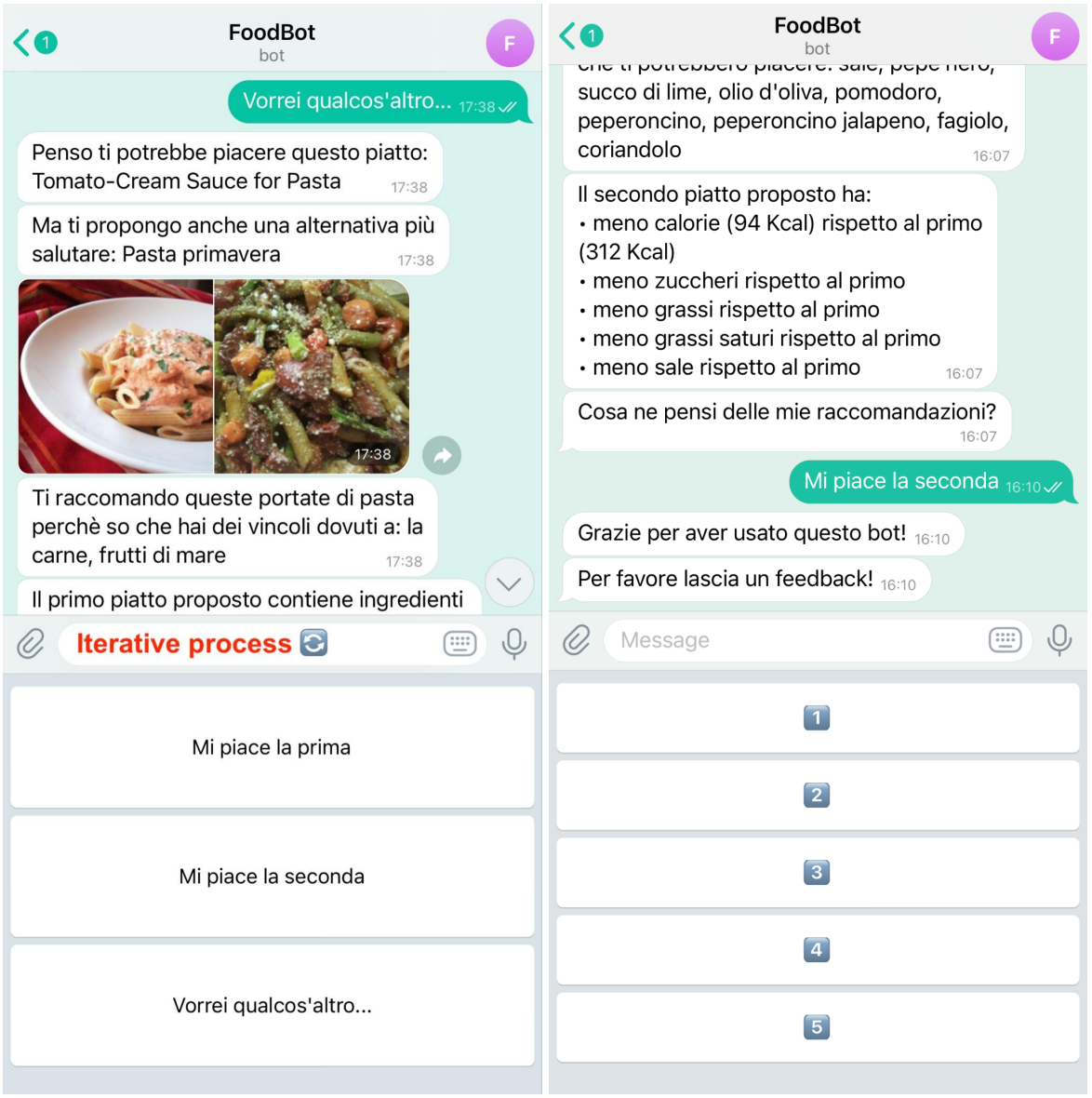
The recommendation explanation consisted in explaining why the second proposed alternative is healthier by looking at dietary metrics comparison.

We compared 5 dietary information for recommended item and healthier alternative:

- calories (Kcal)
- sugars (g)
- fat (g)
- saturates (g)
- salt (g)

We display the reasons why the second dish is better than first by looking at the comparison of these metrics. So, the user is informed about the less amount of sugars in an alternative dessert dish or a less kcal amount in a pasta dish. During the recommendation explanation step we also set a threshold for displaying ingredients that the user might like, by looking at the average TF-IDF score of the features in the user profile. In this way, the most important ingredients for each user are displayed to justify system choices during the recommendation process.

At each step, where user receives tow dishes (most similar to its preferences and healthier alternative), she is asked to choose one or to ask for a new comparison, by means of keyboard buttons (as done before by setting *ReplyKeyboardMarkup* features using *python-telegram-bot* library). If the user asks for a new comparison, the next element of the recommendation list is extracted and a new alternative is processed. Finally, we asked to users to evaluate the experience by giving a score from 1 to 5.



The main progresses in the project Milestone 5 are reported below:

- ☐ We produced two recommendation lists used to produce a recommendation of pairwise shown dishes (recommended one and healthier alternative).
- ☐ We designed an explanation recommendation process based on features liked by users and on dietary metrics.
- ☐ We asked for user system evaluation

Project work GitHub repository

The repository of project work is private. Please tell us how to share it with you. Anyway, the code of Milestone 1, Milestone 2 and Milestone 3, Milestone 4 and Milestone 5 will be attached to this documentation.

In the next milestone, we will run a within-subject user study (N = 15) with participants who will evaluate the experience of our chatbot with both interaction modalities (T, MM).

Castiglia Giovanni, Calò Federica

MILESTONE 6

Motivation

In this last milestone, we set up a small user study (N=30) with family-and-friends participants. We let them take part in this within-subject study which consisted in interacting with our two versions of the bot (different interactions modality) in turn: in our case we decided to present firstly the textual modality bot (T) and after that the multi-modal modality bot (MM). The users who completed the task of interaction with the two bot versions had to compile a short questionnaire with 5-point likert scales. We performed a simple analysis of variance (T-test) for investigating the impact of interaction modality (T, MM) on various acquired variables.

Insights from the questionnaire

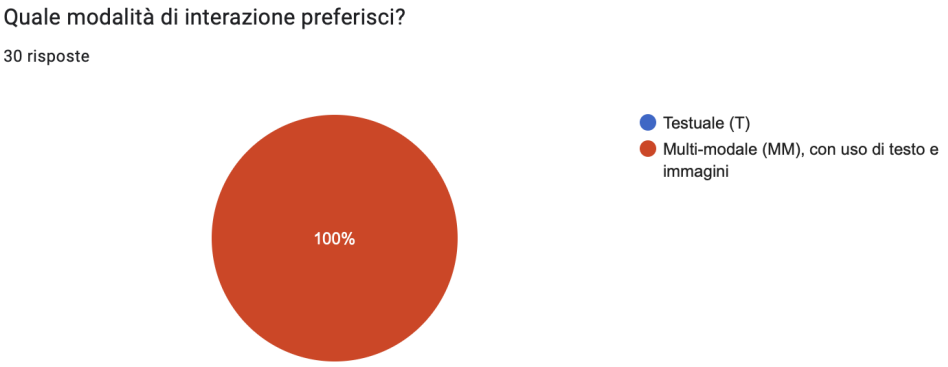
We designed a short Google Form questionnaire in which we asked user to specify the preferred modality and evaluate on 5 point scales the following metrics:

- satisfaction (what interaction modality is **more satisfying**?)
- effectiveness (what interaction modality is **easier to use**?)
- efficiency (what interaction modality is **faster to use**?)

We also asked users to express the reasons for their choice using a multiple checkboxes question.
Our form is available here: https://forms.gle/okQYPcGSLqMNPKky9.

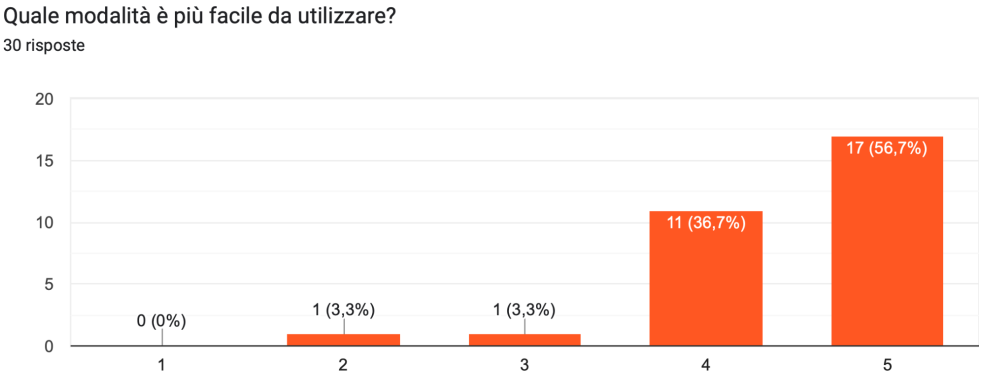
We retrieve the following insights:

1. All the users (100%) preferred the multi-modal modality.

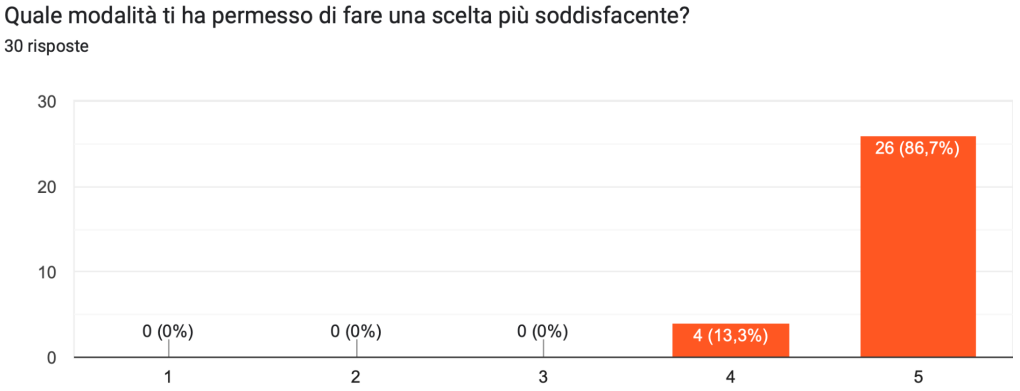


2. The main reasons for their choices is that MM representation allows them to identify clearly the dishes wrt the T representation (96.7%) and that image quality is good (20%).

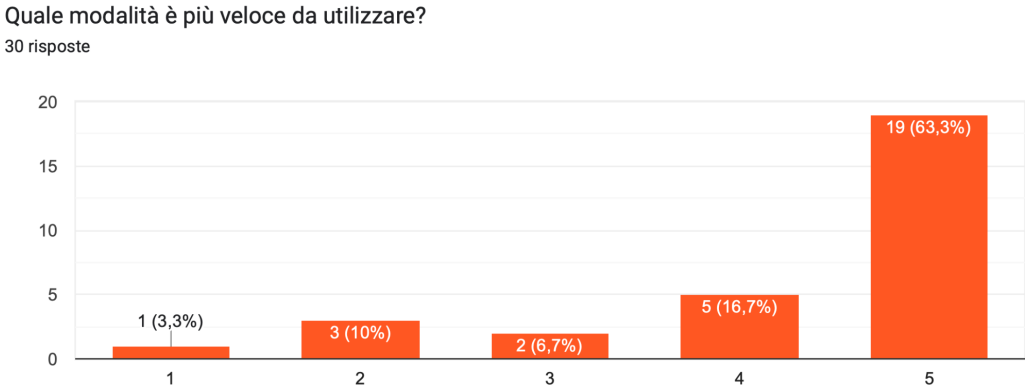
3. The majority of users strongly thinks MM modality is easier to use (56.7%) but also in a partially decisive way (36.7%).



4. The majority of users are completely satisfied with MM modality (86.7%) and also partially satisfied with MM modality (13.3%).



5. The majority of users think that the MM modality is faster to use (63.3%)



T-test analysis of variance

We performed a T-test to understand if two considered populations are statistically different from each other. Our aim is to understand if the change of modality (T, MM) directly impacts certain measured variables. T-test (like ANOVA with more than two groups) looks at the difference in means and the spread of distributions across **two groups**. It tests the so-called **null hypothesis**: it defines if two groups have the same population mean. In our case, if the null hypothesis is rejected, there is a good signal from the test, so the change of modality has a direct impact on the tested variable. T-test produces a test statistic value (T-statistic) which is then converted to a p-value. Generally a p-value lower than 0.05 allows us to reject the null hypothesis.

Example: the null hypothesis tells that the variable X = conversation duration is the same for all two groups (one per modality, T and MM)

We tested the following variables:

- conversation duration
- preference elicitation step duration
- recommendation step duration
- number of interactions (Note: it increases when user continues the recommendation step, by asking for new dish comparisons)
- has user performed a healthier choice (boolean flag)
- FSA score of selected dish
- number of skips during preference elicitation step

We grouped a dataframe (obtained from our measurements dataset) by the field scenario (Textual or Multi-modal) and we split it into two different dataframes. This helped us easily perform means and creating groups of values for each studied variable.

We obtained these results from the mean calculation of some fields:

Variable	Mean of values for T group	Mean of values for MM group
Conversation duration	148.46 s	132.03 s

Preference elicitation step duration	51.06 s	56.86 s
Recommendation step duration	37 s	28.13 s
Number of interactions	6.03	6.3
Number of skips	5.06	6.13

We obtained the following results from the T-test.

Variable	**T-statistic**	**p-value**	**Null Hyp. Rejected?**
Conversation duration	0.777	0.440	No
Preference elicitation step duration	-0.749	0.456	No
Recommendation step duration	1.880	0.065	**No***
Number of interactions	-1.705	0.093	**No***
Has user performed a healthier choice	-0.255	0.799	No
FSA score	-1.223	0.225	No
Number of skips	-0.872	0.386	No

* We obtained a good p-value but not sufficient for declaring rejected the null hypothesis. This could be caused by the relatively small number of participants. Since the p-value is greater than 0.05 so we cannot reject the null hypothesis of the

test. We don't have sufficient evidence to say that the mean duration time between the two modalities is different.

In a certain way, hypothetically we could assume that the recommendation step time is influenced by the modality proposed to the user (T,MM). This could tell us that the user spent less time when the recommendation is accompanied by images. They have the power of driving users faster to the desired item.

Moreover, the majority of users have performed a healthier choice using both interaction modalities. We can also state that the MM modality has slightly better influenced the user to perform healthier choices. Results, which could be not so significant due to the small user set, are displayed below.

Has the user performed a healthier choice?	True	False
T modality	16	14
MM modality	17	13

The main progresses in the project Milestone 6 are reported below:

- ☐ We performed a small user study with 30 participants
- ☐ We designed and administered a simple Google form used to ask users to evaluate which interaction modality they preferred during the user study (within-subject)
- ☐ We run a simple analysis of variance and of mean and reported insights

Project work GitHub repository

The repository of project work is private. Please tell us how to share it with you. Anyway, the code of Milestone 1, Milestone 2 and Milestone 3, Milestone 4, Milestone 5 and Milestone 6 will be attached to this documentation.