UNIVERSITÀ
DEGLI STUDI DI BARI
ALDO MORO

COMPUTER SCIENCE DEPARTMENT

MASTER’S DEGREE IN COMPUTER SCIENCE

THESIS IN

SEMANTICS IN INTELLIGENT INFORMATION ACCESS

“A Personal Assistant for Healthy and Sustainable Food Recommendations”

Thesis Supervisor:

Prof. Cataldo MUSTO

Candidate:

Giovanni Federico POLI

ACADEMIC YEAR

2022/2023

Table of Contents

1. Introduction

1.1. The context

We inhabit a hyperconnected world, where an increasing number of individuals have access to the Web, enabling us to share and access information, opinions, and content of various kinds. As early as 2017, half of the global population was online1, and by 2020 – a year marked by the COVID-19 pandemic that propelled the world online – every minute saw 52,083 users connecting on Microsoft Teams, while Netflix provided 404,444 hours of video streaming to its users.

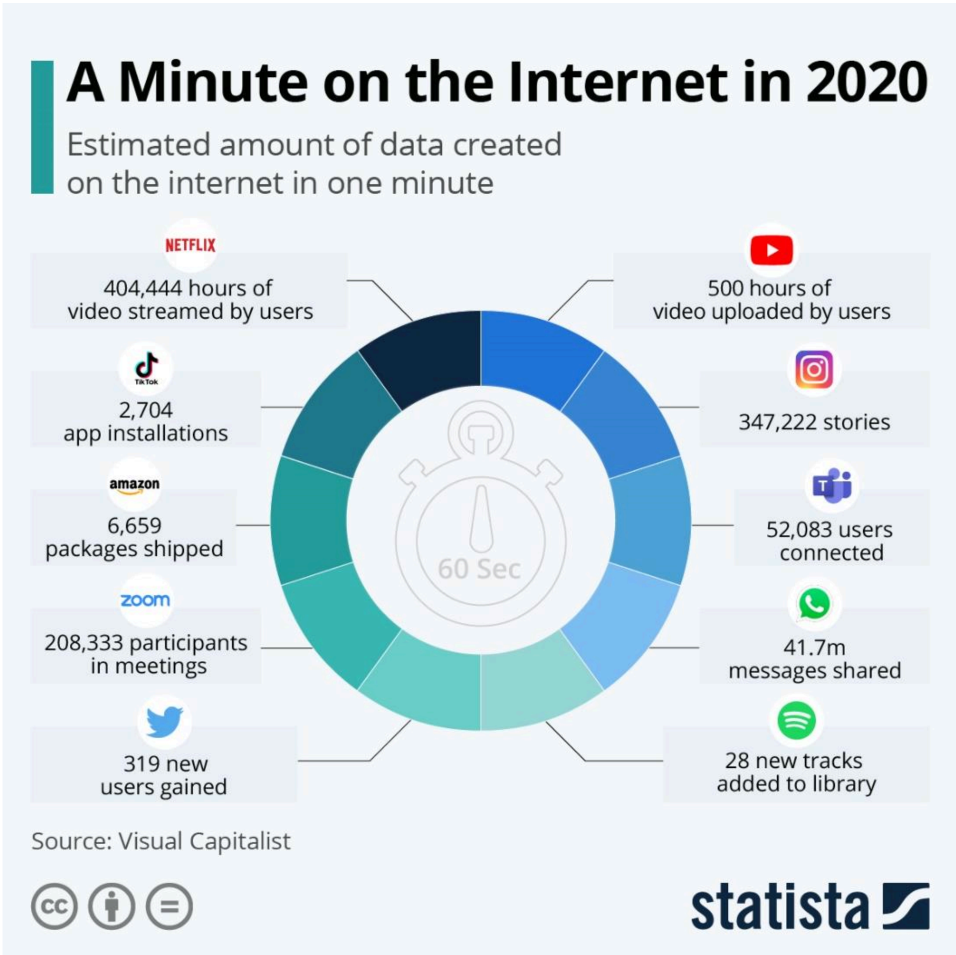


Figure 1: A minute on the internet, Statista.com

Our lives are increasingly intertwined with digital systems that facilitate various aspects of our daily routines and among these, recommender systems have emerged as indispensable tools, aiding us in navigating the vast expanses of information and choices available online.

[1] https://www.statista.com/chart/17518/data-created-in-an-internet-minute/

A recommender system, at its core, is an intelligent information filtering mechanism that predicts user preferences and provides personalized suggestions, thereby assisting users in making informed decisions. This technology has found widespread application in e-commerce platforms, content streaming services, and social media platforms, shaping our online experiences in profound ways.

Concurrently, the proliferation of conversational agents, commonly known as chatbots, has revolutionized the dynamics of human-computer interaction. Chatbots leverage natural language processing (NLP) techniques to engage in dialogue with users, offering assistance, answering queries, and executing tasks autonomously.

Telegram[2], a popular instant messaging platform, has emerged as a fertile ground for the integration of such chatbots, presenting a great tool for users to interact with automated systems.

Amidst the rapid evolution of both recommender systems and chatbots, the fusion of these technologies offers an compelling avenue for investigation and despite notable progress in both domains, a crucial aspect yet to be comprehensively explored is the persuasiveness of this kind of tools.

1.2. The aim

With a focus on promoting healthy dietary choices, the aim of this thesis is to explore the persuasiveness of a healthy food recommender system integrated into a chatbot.

In an environment saturated with dietary choices, the recommender system within the chatbot interface is tasked with not only suggesting nutritious recipes but also justifying these suggestions in a compelling manner. Recognizing the limitations inherent in promoting universally appealing yet not-so-healthy dishes, e.g. carbonara, the system is designed to prompt users with balanced recommendations, based on the user characteristics, while providing persuasive rationales behind each suggestion.

[2] https://telegram.org/

Through this approach, the thesis seeks to investigate the effectiveness of the chatbot in influencing user behaviour towards embracing healthier eating habits.

1.3. Thesis structure

The structure of this thesis follows a logical progression aimed at exploring the persuasiveness of the healthy food recommender system integrated into a Telegram chatbot. Beginning with the Introduction, the context of the research is established.

This sets the stage for delineating the aim of the thesis, which is to investigate the persuasiveness of the chatbot-recommender system in influencing user behaviour towards healthier eating habits.

Following this, the Fundamental Concepts section provides a grounding in essential terminology, including recommender systems, conversational agents, and chatbots, laying the foundation for understanding the subsequent discussions.

The State of the Art chapter surveys existing literature and advancements in the field, providing a comprehensive overview of relevant research and technological developments.

Moving into the core of the thesis, the @food_recsys_bot section delves into the specifics of the integrated system, exploring the architecture, technologies utilized, and the rationale behind design choices.

The subsequent Case Study chapter outlines the experimental methodology employed to evaluate the persuasiveness of the chatbot's recommendations, detailing the experiment setup and the metrics utilized for assessment.

The Results chapter presents the findings of the study, analyzing data collected from the experiment and drawing insights into the effectiveness of the chatbot in changing the user perspective towards healthy and sustainable recipes.

Finally, the Conclusions section summarizes key findings, discusses implications for future research, and suggests potential avenues for further exploration.

2. Fundamental concepts

2.1. The Artificial Intelligence

Artificial Intelligence (AI) represents a transformative field of study and innovation that seeks to emulate or replicate human-like intelligence in machines and systems. The overarching goals of AI are vast and multifaceted, including the reproduction of human cognitive functions such as learning, reasoning, perception, and decision-making. AI systems are designed to perform tasks that traditionally require human intelligence, ranging from basic pattern recognition to complex problem-solving and decision-making processes.

One of the fundamental techniques employed in AI is machine learning, which enables systems to learn from data and improve their performance over time without explicit programming.

Deep learning, a subset of machine learning, involves the use of artificial neural networks inspired by the structure and function of the human brain. These networks are capable of processing vast amounts of data and extracting intricate patterns and features, making them well-suited for tasks such as image recognition, natural language processing, and speech recognition.

Logical reasoning is another key aspect of AI, involving the use of formal rules and logical principles to derive conclusions and make inferences. This technique is often employed in areas such as knowledge representation, expert systems, and automated reasoning.

The applications of AI span across various domains, showcasing its versatility and impact on modern society.

- In the field of medicine, AI is used for diagnostic imaging, drug discovery, personalized medicine, and patient management.
- Industrial automation leverages AI technologies for process optimization, predictive maintenance, and quality control.
- Data analysis and analytics benefit from AI-powered algorithms for pattern recognition, anomaly detection, and predictive modelling.

- Also autonomous vehicles represent a prominent application of AI, where machine learning and computer vision algorithms enable vehicles to perceive and navigate their environment independently.
- Recommendation systems, commonly seen in e-commerce platforms and streaming services, utilize AI to personalize content and make tailored suggestions to users based on their preferences and behaviour.
- Virtual assistants like Siri[3], Alexa[4], and Google Assistant[5] exemplify AI applications in virtual assistance, employing natural language processing and machine learning techniques to understand and respond to user queries and commands[6].
- Robotics is another domain where AI plays a crucial role, enabling robots to perform complex tasks in industrial settings, healthcare, exploration, and entertainment.

Hence, it is safe to say that artificial Intelligence (AI) has permeated virtually every aspect of modern life, showcasing its ubiquitous presence and impact across diverse domains. From machine learning and deep learning algorithms powering advanced pattern recognition and decision-making systems to logical reasoning techniques facilitating knowledge representation and automated reasoning, AI technologies have become integral to numerous applications.

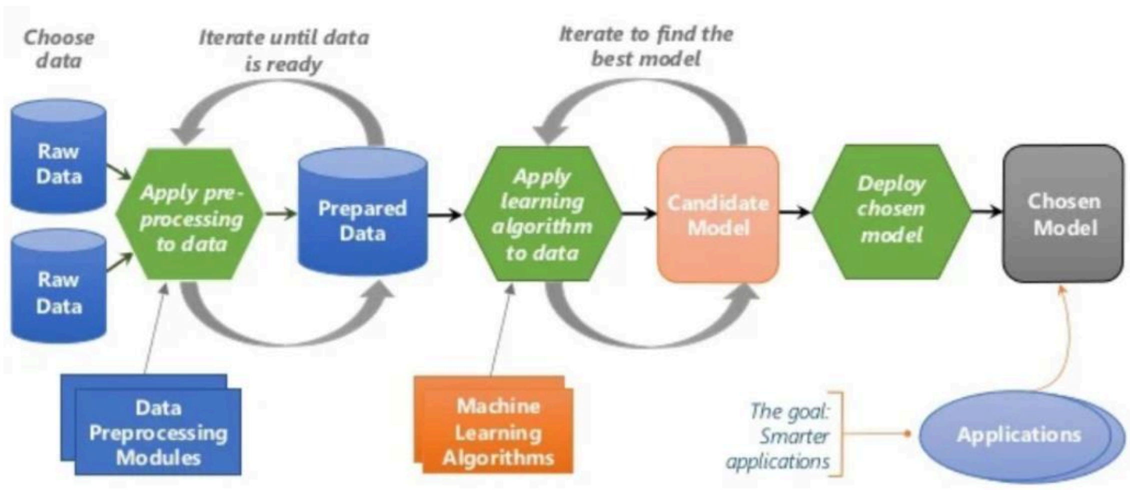


Figure 2: Machine Learning process

---

[3] https://www.apple.com/siri/
[4] https://www.alexa.com/
[5] https://assistant.google.com/
[6] V. Kumar, A. Dixit, R. R. G. Javalgi e M. Dass, «Research framework, strategies, and applications of intelligent agent technologies (IATs) in marketing,» Journal of the Academy of Marketing Science, vol. 44, n. 1, pp. 24-45, 2016.

The phases of AI - namely learning, reasoning, and action - collectively represent the iterative and dynamic nature of AI systems as they interact with data and respond to various stimuli.

1. Learning:

   The learning phase is fundamental to AI systems as it involves acquiring knowledge, skills, and patterns from data or experience. Machine learning algorithms, for instance, learn from labelled or unlabelled datasets to identify patterns and relationships, enabling the system to recognize and classify new data points. Deep learning models take this a step further by learning hierarchical representations of data, which are essential for tasks like image recognition and natural language processing. Reinforcement learning algorithms learn through trial and error, receiving feedback from the environment to refine their decision-making processes. So, the learning phase grants AI systems the ability to recognize patterns, make predictions, and adapt to new information.

2. Reasoning:

   The reasoning phase includes the processes of inference, decision-making, and problem-solving based on the knowledge acquired during the learning phase. Logical reasoning techniques, such as deductive and inductive reasoning, allow AI systems to draw conclusions and make inferences based on logical rules and patterns. Probabilistic reasoning techniques, such as Bayesian inference, enable systems to assess uncertainties and make decisions based on probabilities. Additionally, symbolic reasoning techniques, including knowledge representation and rule-based systems, facilitate complex problem-solving and decision-making in domains like expert systems and automated reasoning. Reasoning enables AI systems to analyze information, derive insights, and make informed decisions or recommendations.

3. Action:

   The action phase involves the execution of decisions or tasks derived from the learning and reasoning phases, resulting in tangible outcomes or responses. This phase is where AI systems interact with the external environment or users, applying their knowledge and reasoning capabilities to generate actions or responses. In autonomous systems like self-driving cars, the action phase involves navigation, control, and decision-making based on real-time sensory input and learned models. In conversational AI systems, the action phase entails generating appropriate responses or actions based on user inputs and the system's understanding of the context.

Overall, the action phase bridges the gap between AI systems' internal processes and their external interactions, manifesting in behaviours, decisions, or outputs that impact the surrounding environment.

In AI, the actors ("*who act*") are the so-called intelligent agents. These agents, acting as the primary actors in AI, interact with their environment through a process of perception and action. Equipped with sensors, they gather information from their surroundings, while actuators enable them to respond and act upon this data, whether in virtual or physical domains.

What distinguishes intelligent agents is their autonomy and goal-driven nature. They are designed to operate independently, pursuing specific objectives or tasks without constant human intervention. This autonomy is powered by a sophisticated process that involves collecting and analyzing data using AI algorithms, leading to informed decision-making and action.

The complexity of intelligent agents varies widely, from basic rule-based systems to advanced machine learning or reinforcement learning-based agents. While some agents rely on predefined rules and responses (reactive agents), others can learn from experience and improve their performance over time, Through continuous learning and feedback loops, they refine their capabilities, leading to improved performance and effectiveness over time. This adaptability and learning ability enable them to navigate complex scenarios and make informed decisions in dynamic environments.

In practical terms, intelligent agents find application across a range of domains and contexts. For instance, chatbots leverage natural language processing to provide user assistance, while autonomous driving systems use sensor data to make real-time decisions for safe navigation on roads. These applications showcase the versatility and utility of intelligent agents in modern AI ecosystems.

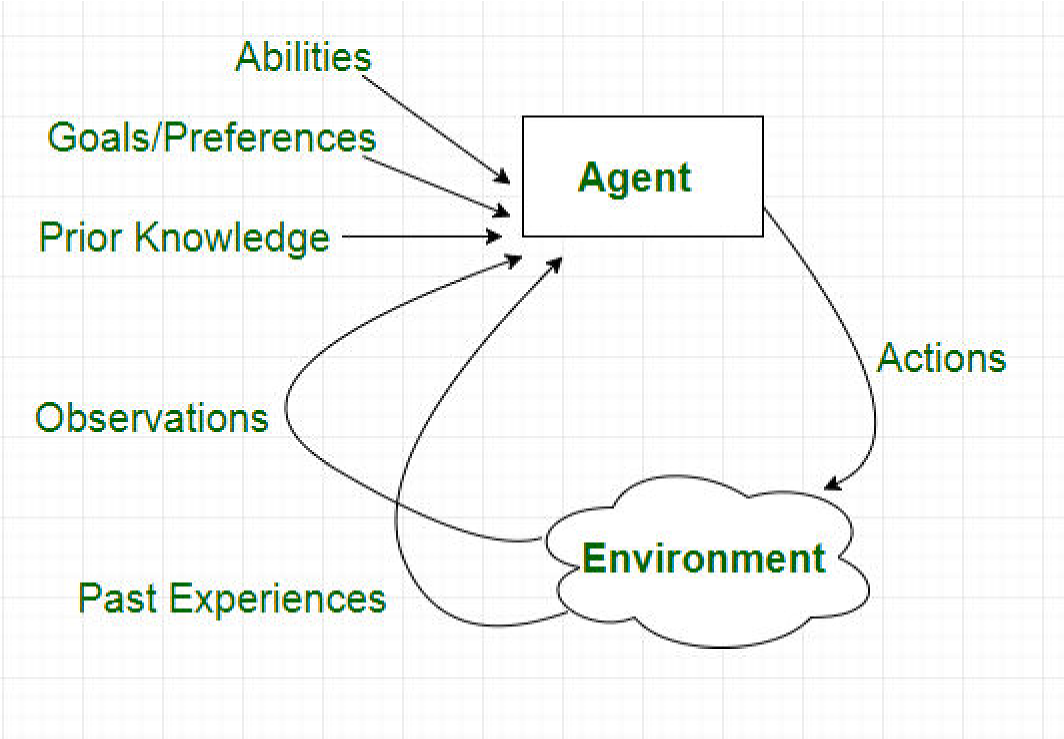


Figure 3: Intelligent agents actions schema

The behaviour of intelligent agents is influenced by a multitude of factors, both internal and external.

- Internally, these agents rely on accurate perception, defined goals, knowledge acquisition, reasoning processes, and decision-making mechanisms.
- Externally, factors such as feedback from the environment, user interactions, constraints, and rules shape their actions and responses.

More specifically, an agent's actions are influenced by a combination of internal and external factors that determine its behaviour and decisions.

Firstly, environmental perception plays a fundamental role: through its sensors, which can be cameras, microphones, motion sensors, and other devices, the Agent is able to gather information from the surrounding environment. This information, obtained through environmental perception, directly influences the decisions and actions taken by the Agent.

Another determining factor is the agent's goals and evaluation function. Each agent has specific goals to achieve and an evaluation function that determines how to assess the various possible actions in relation to these goals. In this way, the agent seeks to maximize its evaluation function by choosing actions that are expected to lead to the desired results.

Knowledge and information are further elements that influence the agent's behaviour. agents can have a predefined knowledge base or acquire knowledge from experience or learning. This knowledge, which may include rules, models, strategies, historical data, and more, impacts the actions taken by the agent.

Reasoning and decision-making processes are also integral to an agent's functioning. By using reasoning algorithms and decision-making processes, the agent can process the available information and make decisions based on that information. This can involve weighing different options, predicting the outcomes of actions, or applying optimization strategies.

Interactions and feedback are other important aspects that influence the agent's behaviour because agents can interact with the surrounding environment, other agents, or users, receiving feedback that will influence their future actions. For instance, an agent learning through reinforcement may receive rewards or penalties based on the actions taken, thus influencing its future behaviour.

Finally, constraints and rules may limit the actions that the agent can take. These constraints may be imposed by technical, ethical, legal, or security factors and influence the agent's decisions and actions.

Below are some of the main types of artificial intelligence (AI) systems that have evolved over time, taking into account their hybrid nature and combination of multiple techniques to address complex tasks. [7]

- Specialized artificial intelligence systems serve as a first example, focusing on specific tasks or narrow domains. These systems are optimized for a single activity and use dedicated algorithms to solve particular problems, as demonstrated by

[7] David L.Poole, Alan K. Mackworth «Artificial Intelligence: Foundations of Computational Agents».

DeepMind's AlphaGo[8], specialized in the game of Go and known for defeating world Go champion Lee Sedol in 2016.

Figure 4: AlphaGo logo

- In contrast, flexible artificial intelligence systems are designed to adapt to a wide range of tasks and domains, learning from diverse data and applying acquired knowledge to new contexts. These systems, like Google's TensorFlow[9], are based on machine learning and deep learning, allowing developers to create models for various applications.

Figure 5: TensorFlow logo

- Lastly, hybrid artificial intelligence systems combine different techniques and approaches, such as rule-based reasoning and machine learning, to achieve optimal performance in various situations, as demonstrated by IBM Watson[10], which integrates various technologies to solve complex problems across different sectors. It is important to note that these types of systems are not mutually exclusive and can be combined or integrated depending on the specific needs of the application. Artificial intelligence continues to evolve, introducing new approaches and paradigms that

[8] https://www.deepmind.com/research/highlighted-research/alphago/
[9] https://www.tensorflow.org/about
[10] https://www.ibm.com/watson

expand the possibilities and capabilities of AI systems, including intelligent agents that act within the context of artificial intelligence.

Figure 6: IBM Watson logo

In summary, AI represents a dynamic and evolving field with vast potential to revolutionize industries, enhance productivity, and improve human lives. By harnessing advanced techniques and technologies, AI systems continue to push the boundaries of what machines can achieve, forming the way for a future where intelligent systems collaborate seamlessly with humans to tackle complex challenges and drive innovation.

2.2. Recommender systems

In our current times, the proliferation of information and choices has become both a boon and a challenge for individuals seeking to navigate all the options available to them. Recommender systems emerge as invaluable tools in this context, designed to alleviate the burden of choice overload by providing personalized recommendations tailored to individual preferences and needs.

At their core, recommender systems serve the fundamental purpose of assisting users in making informed decisions among all the available options, whether in the domain of entertainment, shopping, or, also, dietary choices.

Central to the functioning of recommender systems are the entities that constitute their framework:

- objects;
- users;
- transactions;

Objects represent the items or entities being recommended, which could range from movies and books to products and recipes.

Users, on the other hand, are the individuals for whom recommendations are generated, each characterized by their unique preferences, behaviours, and past interactions with the system.

Transactions encapsulate the interactions between users and objects, providing valuable data that forms the basis for recommendation generation.

The operation of a recommender system typically unfolds through several distinct phases, each playing a crucial role in the recommendation process:

1. Training Phase:
   The training phase represents the foundations upon which the recommender system is built. During this phase, the system gathers and processes a vast amount of data, including user interactions, item attributes, and contextual information. This data is then used to construct a comprehensive model of user preferences and item characteristics. Techniques such as machine learning algorithms, data mining,

and statistical analysis are employed to extract patterns and insights from the data, enabling the system to understand the underlying relationships between users and items.

2. User Modeling Phase:
   With the main model in place, the recommender system proceeds to the user modeling phase. Here, the system refines its understanding of individual user preferences based on their interactions with the system over time. Furthermore, in the user modeling phase, ongoing research by Zhou et al.[11] focuses on enhancing user modeling through the incorporation of multimodal data, such as user preferences expressed through text, images, and social interactions. By leveraging multimodal information, recommender systems can better understand user preferences and provide more personalized recommendations.

3. Filtering and Recommendation Phase:
   The culmination of the recommendation process occurs in the filtering and recommendation phase. Drawing upon the insights obtained from the training and user modeling phases, the system employs filtering techniques to navigate through vast amounts of data and identify items that are most likely to appeal to individual users. These filtering techniques may include content-based analysis, collaborative filtering, or hybrid approaches that combine multiple recommendation strategies. By considering factors such as item relevance, user similarity, and contextual relevance, the system generates personalized recommendations tailored to each user's unique preferences and needs.

Several approaches have been developed to address the diverse needs and challenges encountered in recommendation tasks.

[11] Zhou, H., Zhou, X., Zeng, Z., Zhang, L., & Shen, Z. (2023, February 9). A Comprehensive Survey on Multimodal Recommender Systems: Taxonomy, Evaluation, and Future Directions.

- The content-based approach involves aligning information about an item's content with the user's profile, including the item's description, attributes, keywords, and labels. These details are compared with the user's profile, constructed by analyzing the items the user has engaged with during their navigation. Based on this comparison, the recommendation system suggests items to the user that align with their interests. Content-based approaches often employ reliable and probabilistic estimation techniques, such as Bayesian classifiers. An example application of the content-based approach is the "Recommended for You" section on Netflix, where the system offers movie suggestions based on content similarities to the user's watched films.

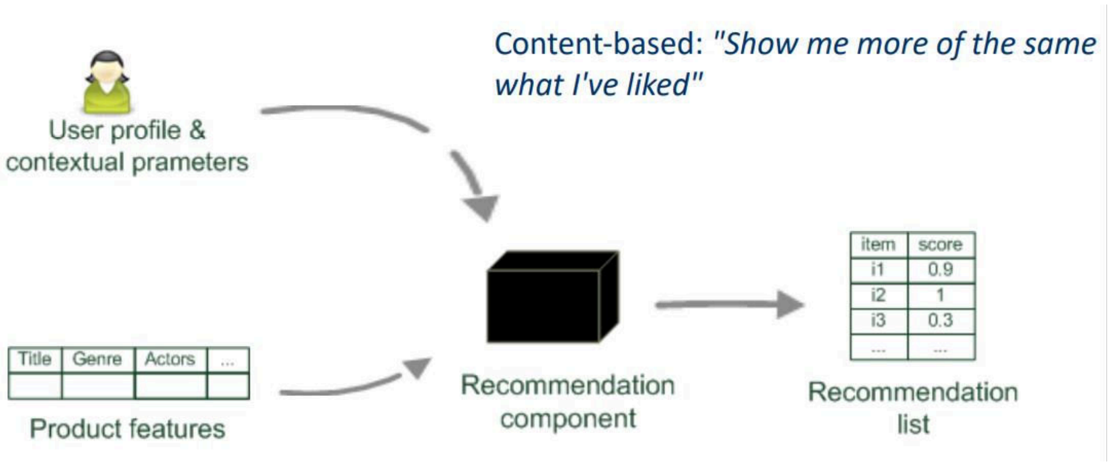


Figure 7: Content Based approach

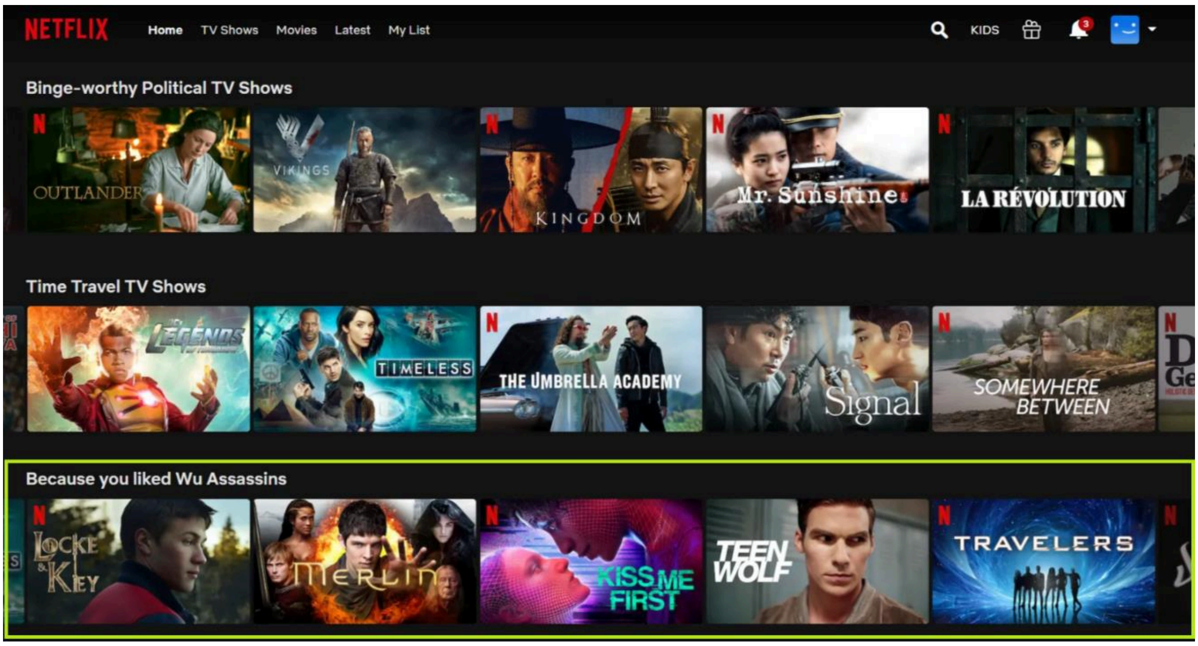


Figure 8: Netflix's content based approach

- On the other hand, the Collaborative Filtering approach utilizes collaborative algorithms to generate recommendations. This approach centers around the concept of "neighbourhood," representing users with similar tastes and preferences to the user for whom the recommendation is intended. In this process, users assign ratings to items, and based on these ratings, the active user's neighbourhood is identified. Subsequently, items that are highly regarded by the neighbourhood and have not been explored by the user are recommended. Collaborative approaches can be categorized into User-To-User and Item-to-Item methods.

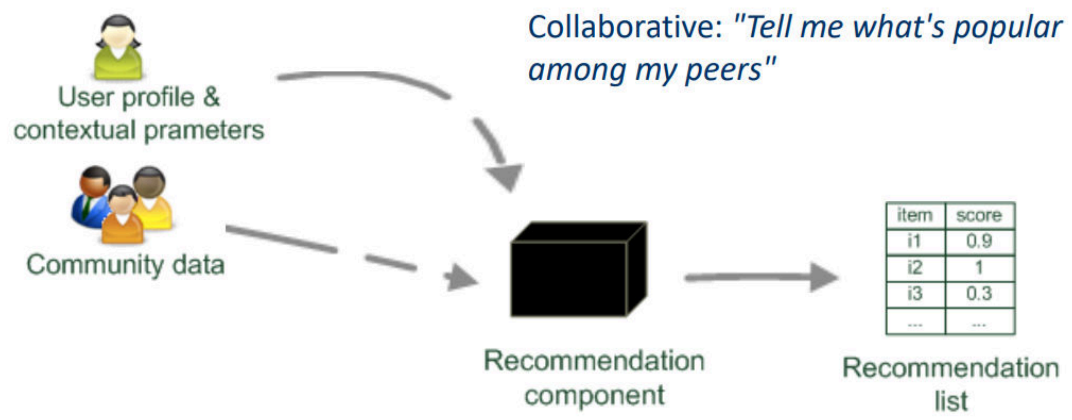


Figure 9: Collaborative Filtering approach

  - User-To-User: the goal here is to identify the neighbour of the active user who has the highest similarity. This is achieved by calculating similarity functions such as cosine similarity or Pearson similarity. Once the neighbour is identified, predictions are made for the values of items that the neighbour has rated but the active user has not. Finally, the item with the highest value is recommended.

    For instance, Amazon also uses collaborative filtering to suggest products based on what similar users have watched and rated.

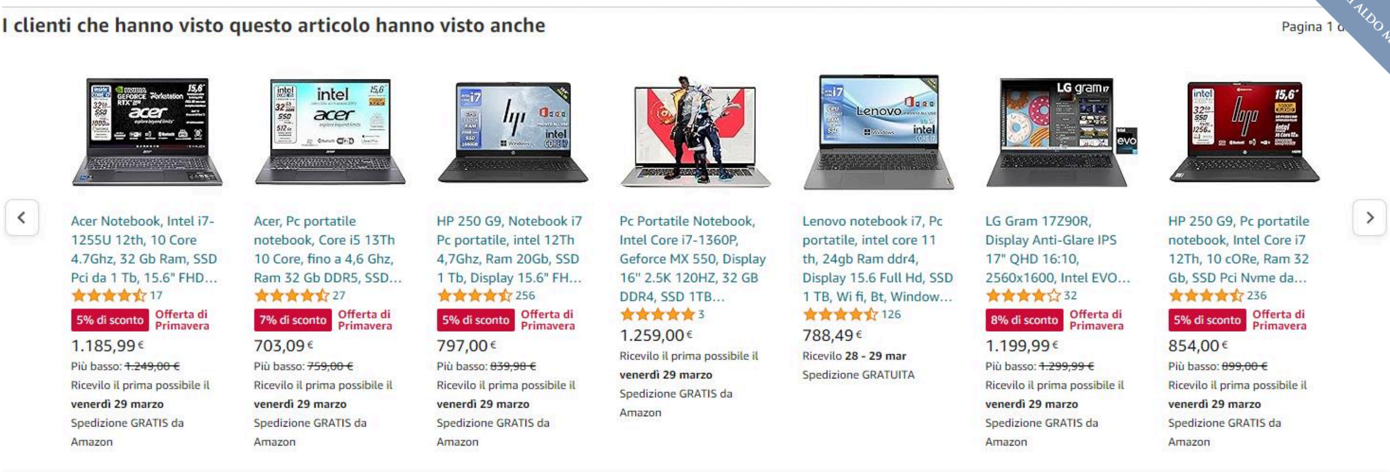


Figure 10: Amazon's Collaborative Filering approach User-To-User

- Item-to-Item: In this approach, the focus is on items known to appeal to the active user. Similarity is sought between the active user's items and all other items to recommend an item that might interest the user. This approach is used, for example, by Spotify to create the Discovery Weekly playlist every week.

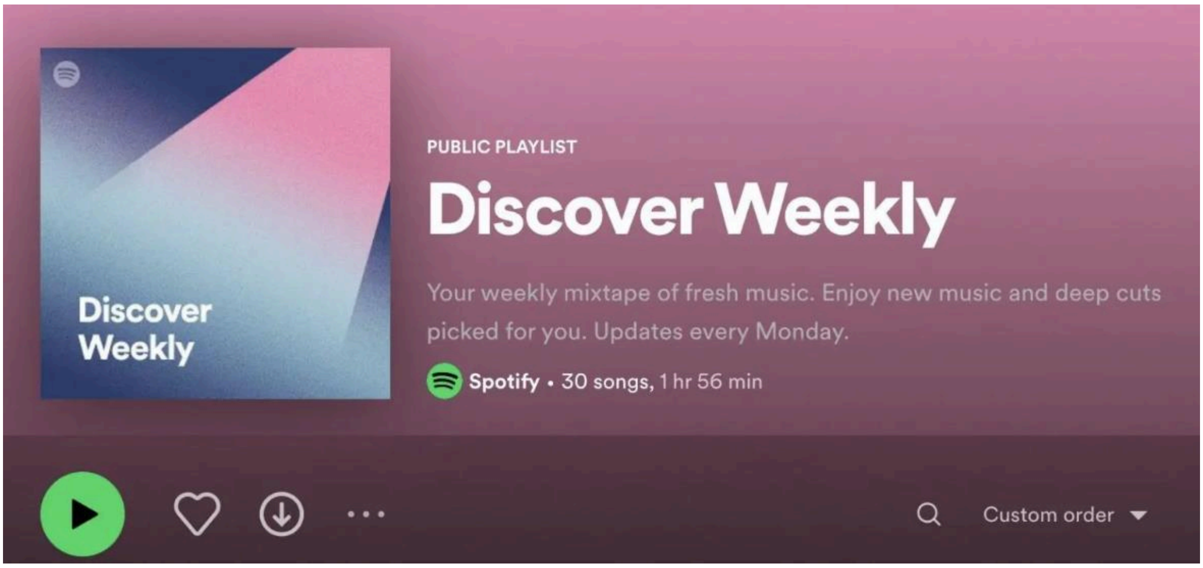


Figure 11: Spotify's Collaborative approach Item-To-Item

Despite its potential, the collaborative approach faces challenges such as data sparsity and the "cold start" problem. Data sparsity occurs when there are few ratings available for items, which can compromise the effectiveness of recommendation algorithms. Additionally, the "cold start" problem arises when new items lack user interactions and ratings, making it challenging for a collaborative algorithm to handle such situations.

Hybrid approaches combine elements of both the content-based and collaborative approaches.

These approaches can be categorized based on how they integrate these two methods, such as separate implementation followed by combining predictions, incorporating content-based features into the collaborative approach, incorporating collaborative features into the content-based approach, or building a unified model.

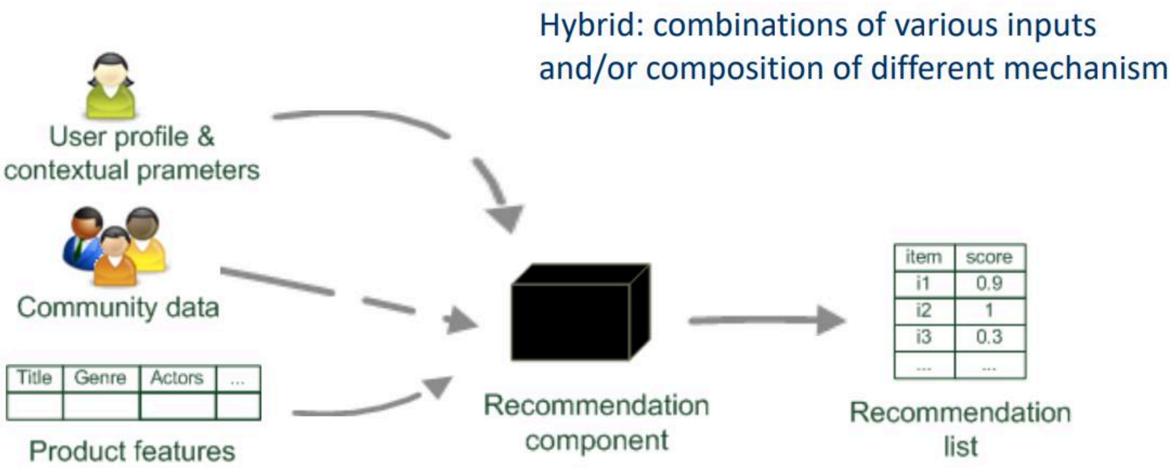


Figure 12: Hybrid approach

In the context of dietary choices, recommender systems hold significant potential to revolutionize how individuals approach nutrition and meal planning. However, food recommender systems face unique challenges that must be addressed to realize their full potential. One such challenge lies in the inherently subjective nature of food preferences, which can vary widely among individuals based on cultural, dietary, and personal factors. Additionally, the complex interplay of nutritional considerations, taste preferences, dietary restrictions, and meal variety further complicates the recommendation process.

Hence, unlike traditional recommender systems tailored for movies or products, food recommender systems needs to handle the complex relationship between taste preferences and nutritional considerations.

While user preferences certainly play a pivotal role in guiding recommendations, the overarching objective of promoting healthy recipes introduces an added layer of complexity. This requires moving away from the conventional approach of simply suggesting items based on user likes or ratings, since prioritizing taste alone may not align with the broader goal of fostering nutritious eating habits.

Despite these challenges, food recommender systems offer promising solutions to address the pressing need for personalized and health-conscious dietary guidance.

2.3. Conversational agents & Chatbots

Conversational agents, or conversational systems, have witnessed significant advancements since the inception of the ELIZA conversational agent by Weizenbaum in 1966[12].

ELIZA was one of the earliest examples of a natural language processing program designed to simulate conversation. The conversational agent operated by employing simple pattern-matching techniques to recognize and respond to user input, emulating the role of psychotherapist. Despite its rudimentary capabilities, ELIZA achieved remarkable success in engaging users in seemingly meaningful conversations by reflecting users' statements back to them in the form of questions and prompts.

What set ELIZA apart was its ability to elicit emotional responses from users, often leading them to anthropomorphize the program and attribute human-like qualities to it. By exploiting the human tendency to anthropomorphize, ELIZA demonstrated the potential for computers to simulate human-like conversational behaviour, laying the groundwork for subsequent developments in conversational AI.

```
  Welcome to
                    EEEEEE  LL      IIII   ZZZZZZ   AAAAA
                    EE      LL       II       ZZ   AA   AA
                    EEEEE   LL       II     ZZZ    AAAAAAA
                    EE      LL       II    ZZ      AA   AA
                    EEEEEE  LLLLLL  IIII ZZZZZZ    AA   AA

  Eliza is a mock Rogerian psychotherapist.
  The original program was described by Joseph Weizenbaum in 1966.
  This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:
```

Figure 13: Conversational agent ELIZA

[12] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1): 36–45, 1966.

Hence, conversational agents have a longstanding history, and their resurgence in recent times can be attributed to a confluence of practical and technological motivations. From a practical standpoint, there is a growing demand for more natural interaction strategies that mimic human conversation. This is particularly evident in scenarios such as driving, where users require hands-free and intuitive interfaces. Additionally, there is a need to automate certain tasks, such as customer relationship management (CRM), to enhance efficiency and productivity.

Figure 14: Conversational agent Clippy

On the technological front, advancements in algorithms have played a pivotal role in revitalizing conversational agents. Improved algorithms for processing voice and audio, understanding user input, and handling natural language both in input and output have significantly enhanced the capabilities of conversational agents. These advancements have paved the way for more seamless and context-aware interactions between users and conversational agents.

Figure 15: Several conversational agents

In terms of input, conversational agents consider the dialogue history, which includes the last few utterances exchanged, and optionally, background knowledge to enrich the conversation context. As for output, the agent generates the next utterance to interact with the user in each turn, along with the possibility of performing specific actions, such as recommending items or controlling devices like lights or music players.

Conversational agents can be broadly categorized into open-domain and goal-oriented agents.

- Open-domain agents are designed for generic chit-chat conversations and can handle a wide variety of topics.

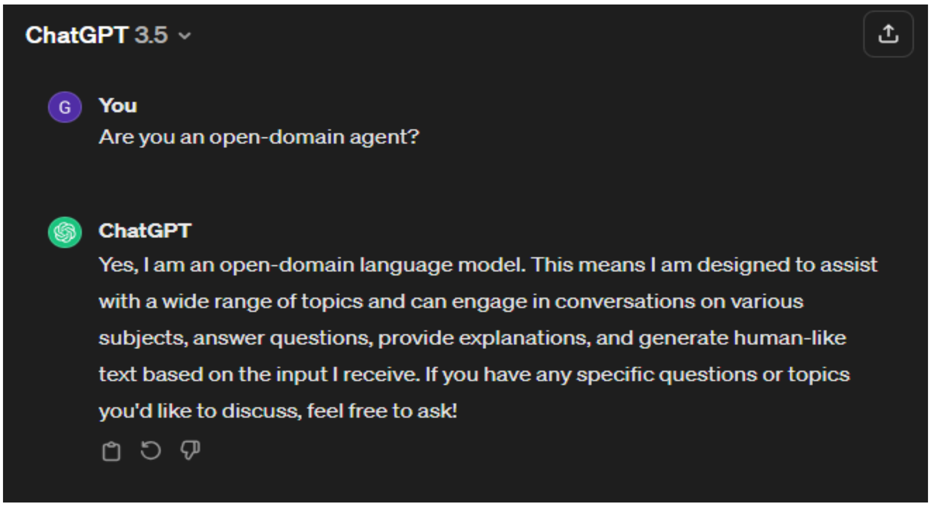


Figure 16: Open-domain agent Chat-GPT

- Goal-oriented agents are tailored for specific domains and are adept at guiding conversations to fulfil user tasks, such as booking flights or recommending movies.

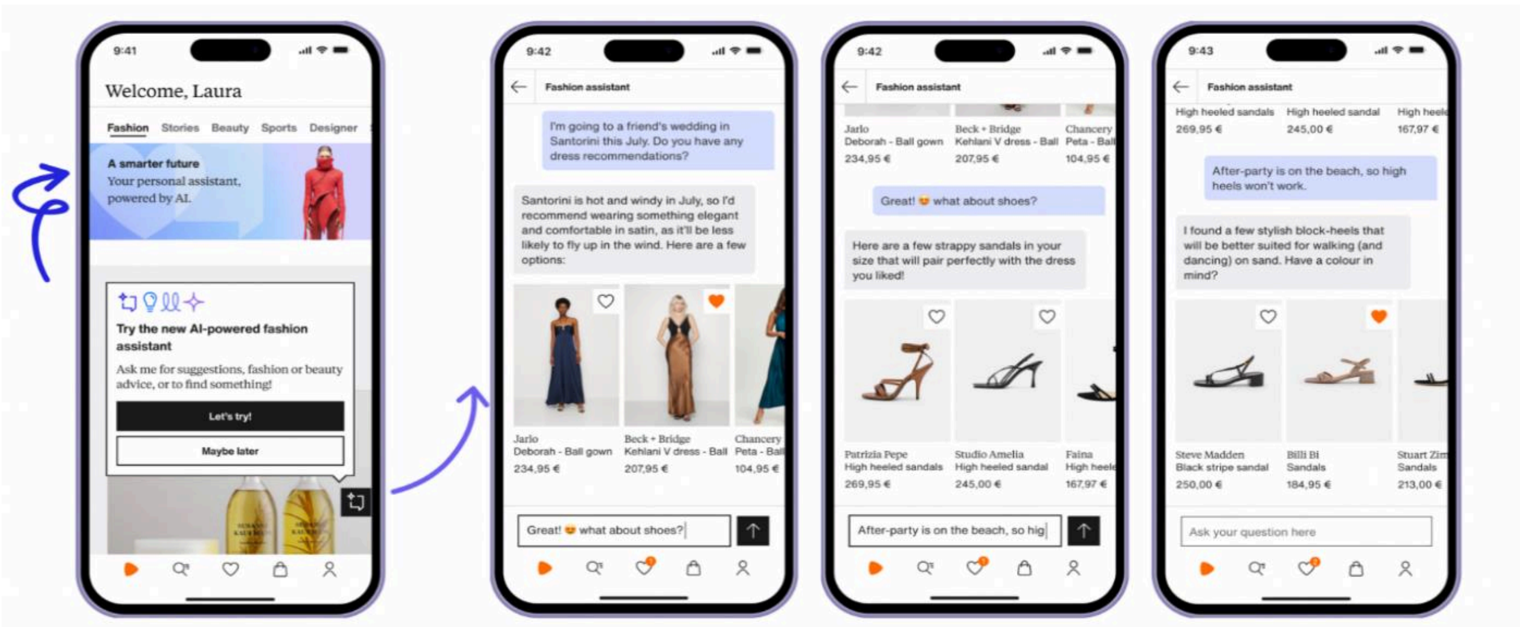


Figure 17: Goal-oriented agent Zalando helper

One key distinction in conversational systems lies in their architecture, with systems categorized as either modular or end-to-end.

- Modular conversational systems include distinct components, each responsible for a specific aspect of the conversation, such as automatic speech recognition, natural language interpretation, dialog management, natural language generation, and text-to-speech synthesis.
- In contrast, end-to-end systems integrate all functionalities into a single framework, offering a streamlined approach to conversation processing.

In this context, conversational recommender systems (CRSs) have emerged as a specialized application aimed at facilitating personalized recommendations through conversational interactions. "*A CRS is a software system that supports its users in achieving recommendation-related goals through a multi-turn dialogue*"[13], they integrate recommendation capabilities into the conversational flow, enabling users to receive tailored recommendations during dialogue exchanges. They focus on guiding the users through a natural conversation to collect their preferences[14] instead of asking them to list all at once.

Input processing for conversational recommender systems (CRSs) involves determining the most suitable interaction modes and strategies.

In particular, CRSs must support various interaction types, including natural language, buttons, or a mix of both, depending on user preferences and the context of use.

The choice of interaction strategy, such as voice, text, or other forms like handwritten input, impacts the user experience significantly and should align with the system's capabilities and user expectations.

Understanding and processing user inputs it's not a trivial task and require robust intent recognition mechanisms to extract the user's underlying needs and intentions accurately.

[13] Jannach, Dietmar, et al. "A survey on conversational recommender systems." ACM Computing Surveys (CSUR) 54.5 (2021): 1-36.
[14] Michael Jugovac and Dietmar Jannach. Interacting with recommenders – overview and research directions. TiiS, 7(3):10:1–10:46, 2017. doi: 10.1145/3001837. URL https://doi.org/10.1145/3001837.

User modeling in CRSs is crucial for personalizing recommendations based on user preferences and informative needs. This involves modeling not only objective features, such as user demographics and past interactions, but also subjective features like user preferences, interests, and contextual factors. Then, the recommendation process involves managing the preference elicitation phase and transitioning to the recommendation phase seamlessly. Effective dialogue state management is essential to track the user's current preferences and guide the recommendation process accordingly.

Output generation instead, focuses on managing user feedback, continuing the dialogue flow, and presenting recommendations in a user-friendly manner.

Returning recommendations in a clear and understandable format is crucial, and explanations can enhance user trust and comprehension.

Semantics also plays a fundamental role in understanding user queries and generating appropriate responses[15], for all the reasons listed so far, the design of CRS involves several key components, such as:

- dialog manager;
- intent recognizer;
- entity recognizer;
- sentiment analyzer;

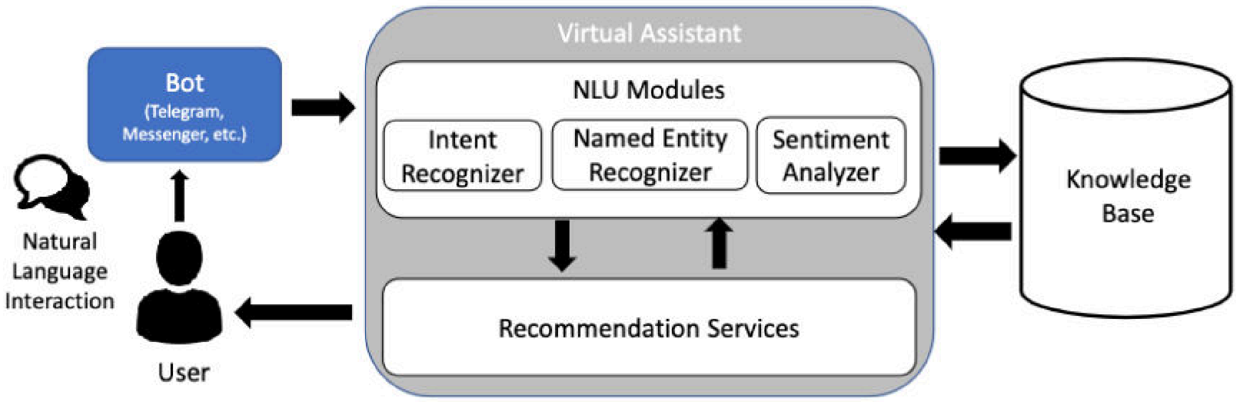


Figure 18: CRS general architecture

[15] Lops Pasquale, Musto Cataldo, Narducci Fedelucio, Semeraro Giovanni, "Semantics in Adaptive and Personalised Systems", Springer.

The dialog manager orchestrates the conversation flow. It manages the sequencing of dialogue turns, maintains context across exchanges, and coordinates the handover between different system components.

The intent recognizer identifies the user's intent or request. By identifying the user's underlying intent, it enables the system to tailor its responses and recommendations accordingly, ensuring relevance and effectiveness in meeting user needs.

The entity recognizer extracts relevant entities or parameters from user input, facilitating context-aware recommendation generation. By parsing user input and identifying relevant entities, the entity recognizer facilitates the creation of personalized recommendations that align with user preferences and requirements.

Additionally, the sentiment analyzer assesses user sentiment or feedback, enabling the system to adapt recommendations based on user preferences and satisfaction.

CRSs are proved to be more effective for more complex recommendations with information overload. For example, planning a trip where multiple agents with different goals are required, or recommend a book/movie where the agent queries are mostly relevant to the system's current context.

An essential aspect of CRSs and, in particular, of this thesis experiment, is also granting clear explanations to users regarding the rationale behind recommendations. Clear explanations not only enhance user understanding and trust but also enable users to make informed decisions.

In our endeavour, we'll leverage Python[16] as programming language and integrate the CRS in the Dialogflow platform[17] by Google. Python offers a versatile and powerful programming environment, equipped with libraries and frameworks for natural language processing and machine learning. Dialogflow offers features such as natural language understanding, intent recognition, and context management, thereby streamlining the development and deployment process of our conversational recommender system.

[16] https://www.python.org/
[17] https://cloud.google.com/dialogflow

2.4. State of the art

Currently, there are several food recommender systems available in the market or open to the public. Among these, two notable ones are:

- Yummly[18]:

  Yummly is a well-known food recommendation system accessible through both an app and website. Its recommendation algorithm incorporates various factors to provide personalized suggestions to users:
  - User Preferences: Yummly allows users to input their food preferences such as favourite or avoided ingredients, dietary restrictions (e.g., vegetarian, vegan, gluten-free, etc.), and personal tastes.
  - Recipe Reactions: Users can interact with recipes in different ways, like adding them to favourites, saving for later, or flagging favourite recipes. These actions impact future recommendations.
  - Popular Recipes: Yummly also considers recipes that are popular and highly rated by users when generating recommendations.

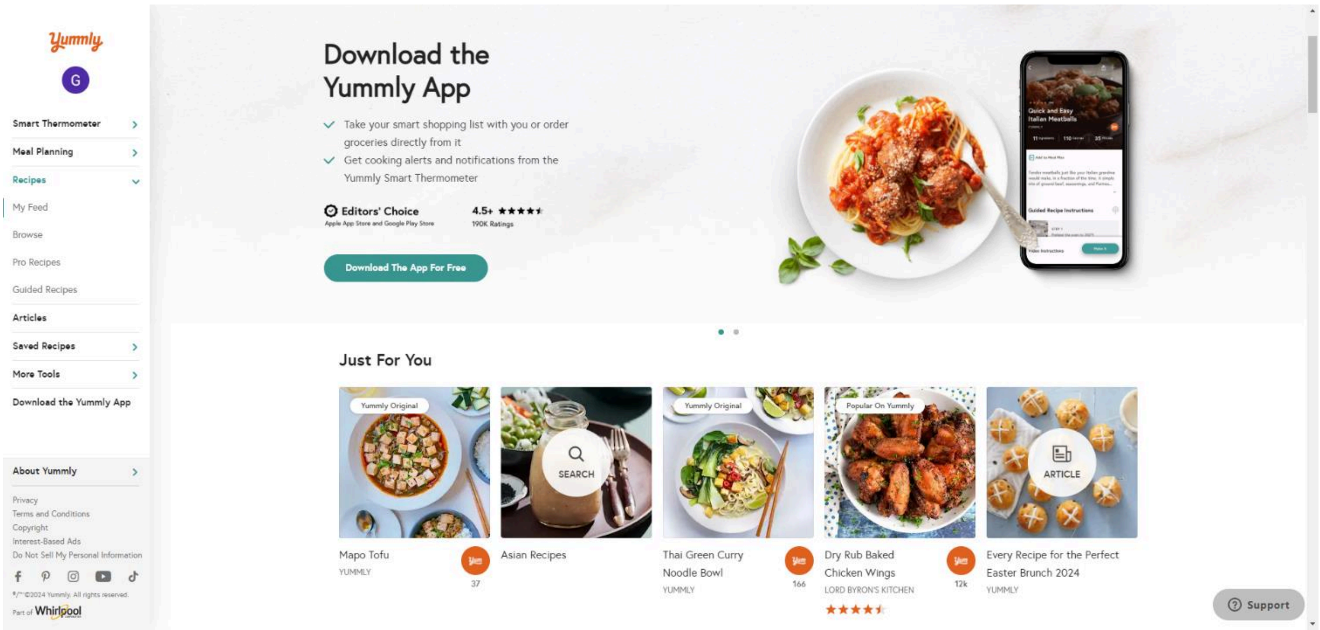


Figure 19: Yummly homepage

[18] https://www.yummly.com/

- AllRecipes[19]:

  AllRecipes is another platform for recipe collection and sharing that employs a recommendation system to propose new recipes to users. Key features of AllRecipes' recommendation system include:
  - User-Favorite Recipes: AllRecipes tracks recipes favoured by users, using them to suggest other similar or related recipes.
  - Reviews and Ratings: User reviews and ratings of recipes are factored into the recommendation algorithm. Well-reviewed and positively rated recipes are more likely to be recommended.
  - Cuisine Style: AllRecipes considers users' preferred cuisine styles such as Italian, Mexican, Asian, etc., and suggests recipes based on these preferences.
  - Trends and Popularity: AllRecipes' recommendation algorithm also takes into account culinary trends and popular recipes of the moment to provide updated and relevant suggestions.

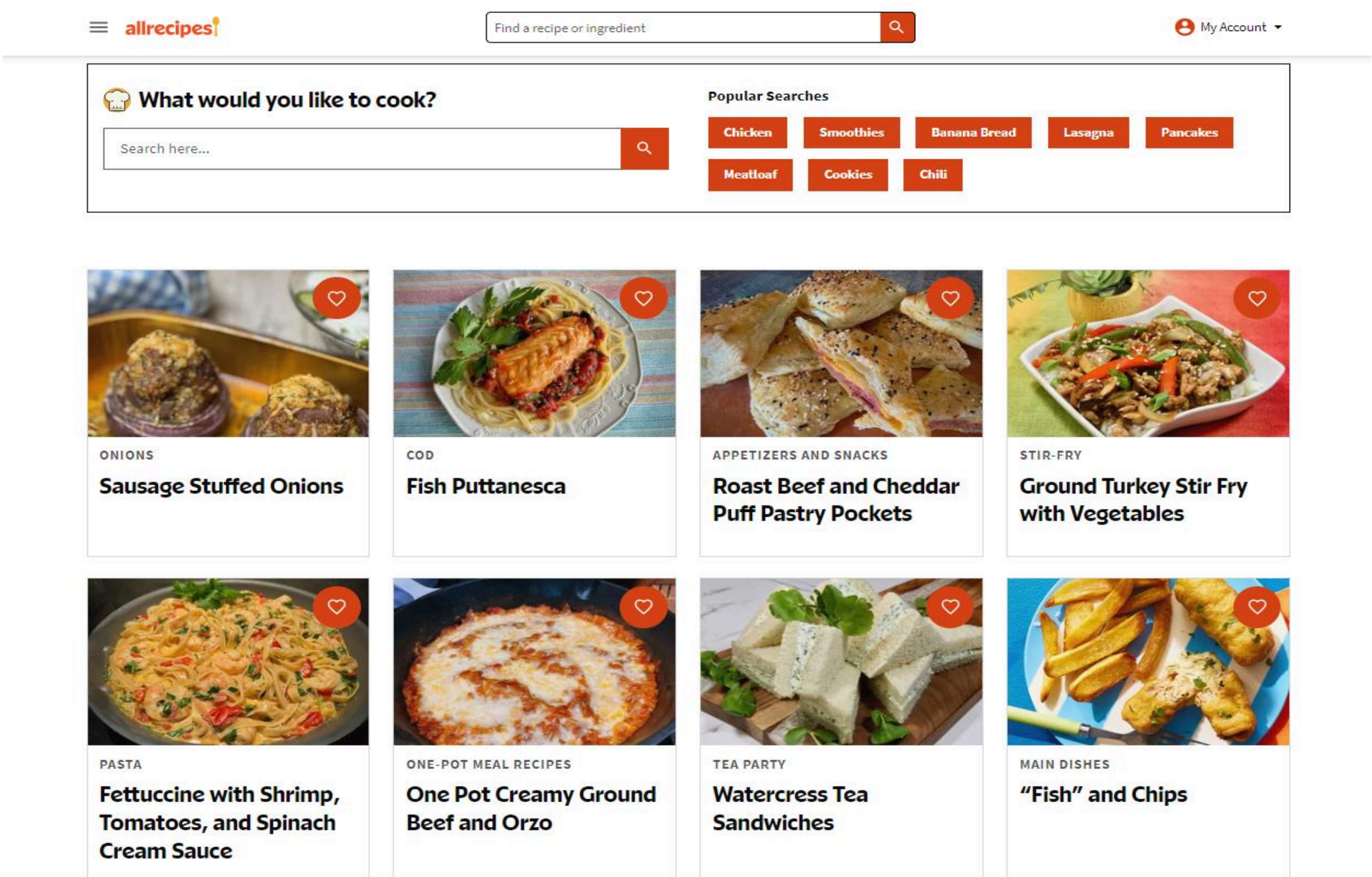


Figure 20: AllRecipes homepage

[19] https://www.allrecipes.com/

However, despite these features, neither of these platforms currently emphasizes macros or calories as primary recipe features, nor do they address user goals related to nutrition. Instead, they focus on suggesting recipes that users might enjoy without incorporating detailed nutritional information about the recipes.

Fewer conversational food recommender systems are available compared to traditional platforms, and none of them delve deeply into the unique characteristics of users as comprehensively as our bot. For instance, many of these systems, such as Mealime [20]- a meal planning app that includes a conversational chatbot feature. Users can chat with the chatbot to receive personalized recipe recommendations based on their dietary preferences, cooking habits, and available ingredients - do not prioritize macros or calories as primary recipe features, nor do they address user goals related to nutrition in a detailed manner. Instead, they primarily focus on suggesting recipes that users might enjoy based on general preferences without incorporating comprehensive nutritional information into their recommendations.

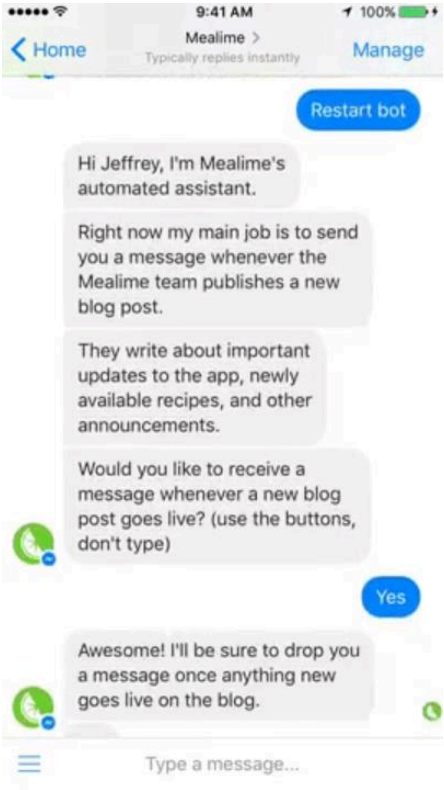


Figure 21: Mealime's automated Facebook assistant

Our bot has been specifically designed to bridge this gap by introducing advanced features analysis that takes into account users' personal preferences, including specific nutritional goals such as macros, calories, nutrients and much more.

[20] https://www.mealime.com/

3. @food_recsys_bot

3.1. The Overview

Food recommendation systems play a significant role in providing personalized suggestions and recommendations to users based on their dietary habits, recipes of interest, and food preferences.

These systems are designed to enhance user experience by offering tailored food choices that align with their individual needs and tastes.

Figure 22: Food Recommender System's Idea

However, despite their potential benefits, current food recommendation systems face several limitations that hinder their effectiveness and reliability.

One of the primary limitations of current food recommendation systems is the lack of a guarantee that the recommended foods are truly healthy for users. While these systems may take into account certain nutritional parameters, they often do not comprehensively assess the overall healthiness of the recommended foods, which can lead to suboptimal dietary choices for users.

Another significant limitation is the assumption that users' past food preferences will remain unchanged over time. This static approach ignores the potential for changes in users' diets, lifestyles, and food preferences, which can significantly impact the relevance and accuracy of the recommendations provided by the system.

Furthermore, food recommendation systems face a data scarcity issue, as users typically evaluate only a small portion of the available foods[21]. This limited data pool makes it challenging to find similar users or foods, thereby reducing the system's ability to generate relevant and diverse recommendations.

To overcome these limitations, leveraging natural language processing (NLP) techniques has emerged as a promising approach. NLP techniques, such as conversational agents, enable food recommendation systems to acquire and filter data differently based on the user's real-time situation and evolving preferences. By incorporating NLP techniques, food recommendation systems can facilitate impartial food evaluations by selecting foods based solely on personal information extracted from the user.

This includes considering dietary restrictions, current food preferences, nutritional requirements, available time for cooking and many others, thereby enhancing the relevance and suitability of the recommended food choices.

In this context chatbots serve as indispensable allies in creating a personalized and interactive experience for users. These intelligent conversational agents, powered by advanced natural language processing capabilities and artificial intelligence algorithms, engage users in meaningful dialogues to understand their unique dietary preferences, restrictions, and goals. Unlike traditional static interfaces, chatbots add a human-like touch by asking relevant questions, providing context-aware recommendations, and adapting their responses based on real-time user feedback.

This dynamic interaction not only enhances user engagement but also enables the system to continually learn and improve its recommendations over time.

Moreover, to enhancing user engagement and improving recommendation accuracy over time, chatbots also play a crucial role in promoting education and awareness regarding healthy eating habits and sustainable food choices.

[21] Rostami, M., Farrahi, V., Ahmadian, S., Jalali, S. M. J., Oussalah, M. (2023). A novel healthy and time-aware food recommender system using attributed community detection. Expert Systems with Applications, 221, 119719. https://doi.org/10.1016/j.eswa.2023.119719

By leveraging chatbots to deliver educational content in a personalized and interactive manner, food recommendation systems can empower users to make informed decisions that align with their health and sustainability goals. This educational aspect not only adds value to the user experience but also contributes to promoting healthier and more sustainable lifestyles on a broader scale.

In line with this mission, our bot, @food_recsys_bot, is designed to not only recommend healthy recipes but also to educate and raise awareness about nutrition and sustainable food practices and its purpose is to serve as a recommendation system for healthy recipes with a focus on providing persuasive explanations for the suggestions.

The bot aims to encourage users to adopt healthier eating habits by offering personalized recommendations based on their preferences, restrictions, dietary goals, nutritional needs and several other characteristics.

Some of the functionalities offered by the bot are:

- Providing personalized recipe recommendations:
  The bot takes into account users' dietary preferences, restrictions, goals, activity levels and many other factors to suggest healthy recipes that align with their preferences, but most importantly with their needs.

- Offering persuasive explanations:
  The bot provides detailed explanations for its recommendations, including comparisons between recipes and insights into their nutritional values. These explanations are designed to educate users and motivate them to make healthier food choices.

- User profiling and customization:
  The bot gathers essential information about users through a streamlined profiling process, allowing it to offer more accurate and relevant recommendations. Users can also modify their profiles to update their preferences and goals whenever they please.

Overall, @food_recsys_bot, empower users to make informed and healthier food choices by providing customized recipe recommendations and meaningful explanations tailored to their individual needs and preferences.

Three are its main components:

1. The bot itself:
   The @food_recsys_bot is the core component of this integrated system, serving as the interface through which users interact with the food recommendation system. The bot is designed to operate within the Telegram platform through Google Dialogflow, providing users with personalized recipe recommendations and persuasive explanations to enhance their experience. It handles user inputs, processes requests, and delivers relevant information and suggestions based on the user's profile preferences.

2. The recommendations system:
   This is responsible for generating personalized recipe recommendations for users. It generates customized recipe suggestions by analyzing user input and profiles, incorporating data from the Italian site giallozafferano.it[22] . It considers factors like dietary restrictions, caloric goals, and nutritional preferences to provide tailored recommendations promoting healthier eating habits.

3. The explanation system:
   The explanation system plays a vital role in providing users with detailed insights and explanations regarding the recommended recipes. It enhances user understanding and awareness by delivering persuasive explanations that highlight the healthiness, nutritional benefits, and sustainability of the recommended recipes. The explanation system employs different explanation types.
   In particular for this thesis we will focus on the explanation methods foodGoals, foodMacros, sustainability and seasonality to provide insights into daily caloric intake, macronutrient distribution, ideal nutritional ratios and information about the

[22] https://www.giallozafferano.it/

seasonality of ingredients as well as their level of sustainability. It also facilitates comparisons between recipes, allowing users to make informed decisions after receiving information about both recipes compared. The explanation system aims to change users' perspectives on food choices, encouraging them to adopt healthier eating habits and make informed dietary decisions.

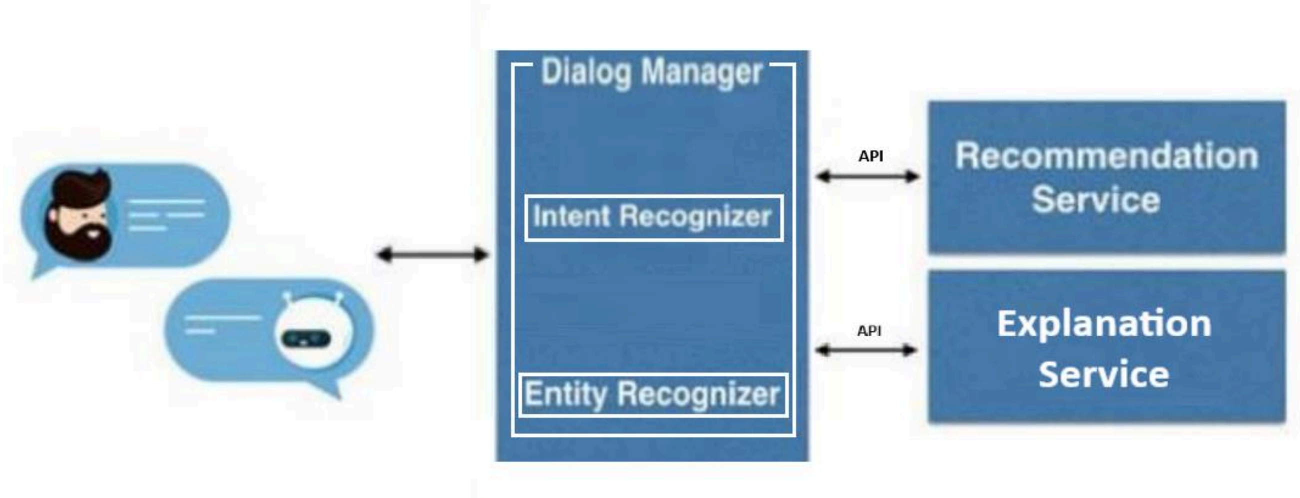


Figure 23: @food_recsys_bot structure

3.2. The implementation

3.2.1 The recommendation system

The recommendation system's task is to provide users with coherent suggestions for dishes and recipes based on their characteristics. The core of this recommendation process lies in the information provided by the user through conversation with the chatbot, which is then processed as parameters for the recommendations. Specifically, this information includes the user's physical state, such as height, weight, and dietary goals, socio-demographic factors like age, lifestyle, or how much they are willing to spend on the dish they are seeking recommendations for.

These insights are then processed in reference to the recipe dataset used for recommendations, structured so that each recipe is associated with various parameters that represent them, aligning with the information gathered from users. Given the extensive nature of the dataset, a recalculation of the score for each recipe based on all parameters is utilized, achieved through a set of rules created based on Food Knowledge[23]. This ensures that recommendations are in line with users' preferences and needs, providing accurate and personalized suggestions.

The recommendation service has been implemented in Python using the Flask framework, suitable for web application development. The recipes recommended by the chatbot are sourced from a dataset containing thousands of recipes from the GialloZafferano.it website. Various libraries were employed during development, such as "Pandas" for dataset handling and the "python-telegram-bot" library for suggestion requests.
The parameters of the request include:

- n: Specifies the number of recipes to extract.
- category: Specifies the categories of recipes requested (i.e., "First courses," "Main courses," "Desserts").
- isLowNickel, isVegetarian, isLactoseFree, isGlutenFree, isLight: Boolean values indicating any dietary restrictions specified by the user, specified separately.
- difficulty: Indicates the difficulty level of the recipe to recommend.
- goal: Refers to the user's dietary goal.
- user_cost: Indicates the user's preferred recipe cost.
- user_time: Indicates the user's preferred recipe preparation time.
- age: Refers to the user's age.
- sex: Indicates the user's gender.
- mood, activity, stress, sleep, depression: Integer values (0=yes, 1=no) related to the user's mood, activity, stress, sleep, and depression status.
- fatclass: User classification based on BMI.

The service primarily consists of two files for recipe recommendation:

1. food_rs_webservice.py:

   This is the main file implemented as a Flask app. It reads information from the database, handles HTTPS requests, and creates recommendations in JSON format

[23] I. Paparella, «Progettazione e Implementazione di un Food Recommender System Basato su Holistic User Model» 2020.

considering the parameters passed by the user. The file also includes Flask app initialization, the definition of the endpoint for the '/’ route, and the main "score" function, which calculates a score for each recipe based on the "ratingValue" and "ratingCount" columns.

2. recommender_script.py:
   This file is crucial for the connection between the recommender system and the Conversational Agent. It contains three classes: Recommendation, Recommendation_due and Recommendation_tre. They all have a static method for generating recipe suggestions based on user information. The Recommendation class handles a single recipe suggestion, while Recommendation_due and Recommendation_tre handle, respectively, two and three different suggestions, in order to prompt the user with a second or a third suggestion whenever the first one is not suitable. The classes process requests forwarded to the server, build the request URL, retrieve recipe data, and send a response containing the title and URL of the suggested recipe.

In addition to this information, the code includes various lists such as RichIn.json, which contains foods categorized on their level of certain substance such as calcium, magnesium, iron and so on, or Seasonality.json and Sustainability.json that asses the levels of seasonality and sustainability of the recipes. These lists can be used to further filter the DataFrame based on user preferences or needs.

This approach allows the chatbot to provide personalized suggestions based on user preferences and the availability of recipes in the dataset, ensuring an interactive and useful experience.

3.2.2 The Explanation system

The explanation system is responsible for formulating and providing users with explanations regarding the recipes recommended by the recommendation system.

When a person seeks new recipes to enrich their culinary knowledge, they are often skeptical and doubtful about whether the recipes align with their interests, intolerances, health conditions, or dietary restrictions.
Therefore, it is crucial that, in addition to recipe suggestions, users can ask questions about a wide range of information, such as nutritional details, potential risks and benefits associated with nutrient intake, costs, preparation times, and adherence to dietary restrictions.

The fact that users can interact in natural language with an entity like the chatbot stimulates their curiosity and cognitive ability regarding food, encouraging them to discover more recipes and information. The explanation system associated with the chatbot enables this by providing several different types of explanations (discussed later in this work) related to the recommended recipes, with options for single styles or comparisons.

The explanations can be grouped into macro-categories, including nutritional aspects, personal factors (such as cost, preparation time, culinary skills, user's goals), health considerations (risks, benefits, and user's age), and recipe popularity. These explanations are available for both individual recipes and comparisons between two recipes.

The explanation system is designed with dedicated modules for generating explanations in natural language based on the parameters received from the recommendation system, transferring explanations to the chatbot, and a main module that coordinates these operations.

The explanation service has been implemented in Python, also utilizing the Flask micro-framework. Unlike the recommendation system, the URL used for the request contains three mandatory parameters, concerning the style of explanation (whether the explanation is singular or comparative), the type of explanation (there are 19 provided explanation types), and the img_url of the recipe recommended by the recommendation service.

There are three Python files forming the explanation service:

1. web_expl.py:

   This is the main file representing the web app written using Flask. The main function in the file is get_expl, which is associated with the path "/expl" in the application through the use of the @app.route('/expl') decorator.

   Upon application startup, paths to JSON files containing various information, including data on nutrients, dietary restrictions, foods rich in certain nutrients, food sustainability, food seasonality, and their relation to dopamine, are defined.

```
@app.route('/expl')
#class Explain(Resource):
def get_expl():
        nutrientsPath = 'Nutrient.json'
        restrictionsPath = 'Restrictions.json'
        richInPath = 'RichIn.json'
        sustainabilityPath = 'Sustainability.json'
        seasonalityPath = 'Seasonality.json'
        dopaminePath = 'Dopamine.json'

        recipeA_url = request.args.get('imgurl1')
        recipeB_url = request.args.get('imgurl2')

        url_dataset_en = 'dataset_en.csv'
```

   Figure 24: /expl route

   Subsequently, within the get_expl function, the contents of the JSON files are read and loaded into corresponding variables. The path to a CSV file representing the recipe dataset is also specified.

   Recipes to compare are then identified through the URLs provided as parameters in the GET request.

   A scan of the rows of the CSV file is performed to find information corresponding to the URLs of the provided recipes. The recipe information is then stored in two variables, recipeA_values and recipeB_values.

   Following this, an empty dictionary user is created containing various user parameters provided as parameters in the GET request.

   Subsequently, two lists of experiments are defined, one for explanations related to a single recipe and the other for comparative explanations between two recipes. These lists contain strings representing the types of experiments that can be executed to generate explanations.

   An empty dictionary explanations is then created to store the generated explanations. The index of the requested experiment and the index of the desired explanation style in the GET request are checked. If the indices are valid, the corresponding explanations are generated using the get_str_exp function defined in the external module called expl_types. The explanations are then added to the explanations dictionary. Finally, the explanations are converted to JSON format using Python's json library, and the resulting JSON is returned as the client's request response.

2. expl_types.py:
   Its main function is get_str_exp, which is called by web_expl.py and coordinates the creation of justifications by selecting the specific type requested.
   The get_str_exp function accepts several parameters:
   - exp_type: a string representing the desired type of explanation.
   - recipeA_values and recipeB_values: the values of properties of the two recipes.
   - user: the dictionary containing user parameters created in web_expl.py.

   Within the get_str_exp function, a check is performed on the value of exp_type to determine which type of explanation should be generated. Based on the value of exp_type, the corresponding function is called to create the specific experiment.
   Other functions in the module contain implementations for creating specific types of explanations that will be later discussed.

3. The third and final file related to the explanation service is expl_script, a file very similar to the previously described recommender_script. It is also an essential auxiliary file for the connection between the chatbot and the Recommender system.
   In this file, a single class named Explanation is defined, comprising 33 static methods, each related to the explanation and comparison types. The structure of all methods is the same; therefore, we will explain the general structure of a function related to explaining a single recipe and one related to comparing two recipes, which are then applicable to all functions.

As a general example of explaining a single recipe, we take the "spiegazione_piatto" method, which is called when the user requests a general explanation of the recommended recipe.

This method, like all others, accepts two parameters: update and context, which are specific objects for interaction with the Telegram bot. The method structure is as follows:

1. Initially, an empty list named restr_list is initialized, which will be used later to store dietary restrictions.
2. The value of restrictions within the context.user_data dictionary, representing the recipe recommended by the recommendation service, is then checked. If a restriction is active (value equals 1), the corresponding identifier is added to the restr_list.
3. Next, a parameter named restr is created, to which the restriction strings are concatenated, separated by a comma, using the join method. If restr_list is empty, restr will be set to None.
4. An URL towards the external web service handling explanationsis then defined.
5. Similarly, a dictionary named params is created, containing the parameters necessary for the web service request. These parameters include the type of explanation, the style, and other specific user information.
6. The img_url parameter of the Recommendation class, is used to refer precisely to the just recommended recipe and perform explanations on it.
7. A complete URL is created by including the parameters using the urlencode function from the urllib.parse library. The first three are mandatory parameters, concerning the type of explanation, the explanation style (0 represents the style of explaining a single recipe, 1 represents the style of and, -1 represents both), and the URL of the image of the recipe recommended by the recommendation service.

```python
url = "http://127.0.0.1:5000/expl?"
params = {
    "type": 18,
    "style":0,
    "imgurl1": imgurl,
    "difficulty": context.user_data["cook_exp"],
    "goal": context.user_data["goals"],
    "user_cost": context.user_data["max_cost_rec"],
    "user_time": context.user_data["time_cook"],
    "user_age": context.user_data["age"],
    "sex": context.user_data["gender"],
    "mood": context.user_data["mood"],
    "bmi": context.user_data["weight"],
    "activity": context.user_data["ph_activity"],
    "stress": context.user_data["stress"],
    "health_style": context.user_data["ht_lifestyle"],
    "health_condition": context.user_data["ht_lifestyle_importance"],
    "sleep": context.user_data["sleep"],
    "depression": context.user_data["depress"],
    "restr": restr,
}
full_url = url + urlencode(params)
```

Figure 25: Example of the created URL

8. A GET request is made to the complete URL using the requests library, and the response is obtained as JSON.
9. This JSON explanation is then processed, particularly from the "explanations" field.
10. If an explanation is present, a process of splitting the explanation into segments of a maximum length of 500 characters is performed to later translate the message.
11. A response message containing the text is sent to the update sender.

3.2.3 The Bot

In the original version[24] of the bot, the process of creation was divided into several phases:

- User Interface Design: First and foremost, the organization of the chatbot's interface on Telegram was planned, using the "Python-Telegram-bot" library in the Python environment to manage conversations and interact with the Telegram API.

- Conversation Flow Definition: The chatbot was programmed to initiate conversations with users following a conversation flow model. This model guides users through

[24] Lopedota, F. (2022/2023) "FoodRecSysBot": Progettazione e sviluppo di un Agente conversazionale per il supporto personalizzato nella scelta di cibo e ricette.

structured questions and answers, adapting the flow based on the received information and storing important details.

- Integration with the Food Recommender System: Several Python libraries were used to connect the chatbot to the Food Recommender System. This system generates personalized culinary recommendations based on users' responses to personal questions.

- Response Management: To efficiently handle the chatbot's responses, a Design Model based on Intents was adopted[25]. This model focuses on identifying user Intents and providing appropriate responses based on these Intents. The Dialogflow platform was used to train the chatbot to recognize user Intents using machine learning algorithms and provide coherent and relevant responses.

The main file for the bot implementation is FoodRecommenderSys.py. In this file, the creation and structure of the Telegram bot that provides suggestions and explanations generated by the food recommender system are implemented. The bot interacts with users through a series of questions about their personal information and then uses the provided answers to offer personalized suggestions based on them. To set up a properly functioning and effective system that allows user interaction using natural language, various libraries and different techniques were used. The main ones are python-telegram-bot, an open-source library for developing Telegram bots using Python, and Dialogflow, which plays a fundamental role in processing user requests in natural language and linking to the Food RS.

[25] Amir Shevat «Designing Chatbots: Creating Conversational Experiences».

First, to create a bot on Telegram, you need to search for "BotFather" within the app itself and follow its instructions to create a new bot. The bot creation process involves obtaining a unique token from BotFather, which identifies the bot. Only after this can the bot code be written, importing the Telegram and Telegram.ext libraries that provide the necessary functionalities to interact with the Telegram API.

Figure 23: BotFather logo

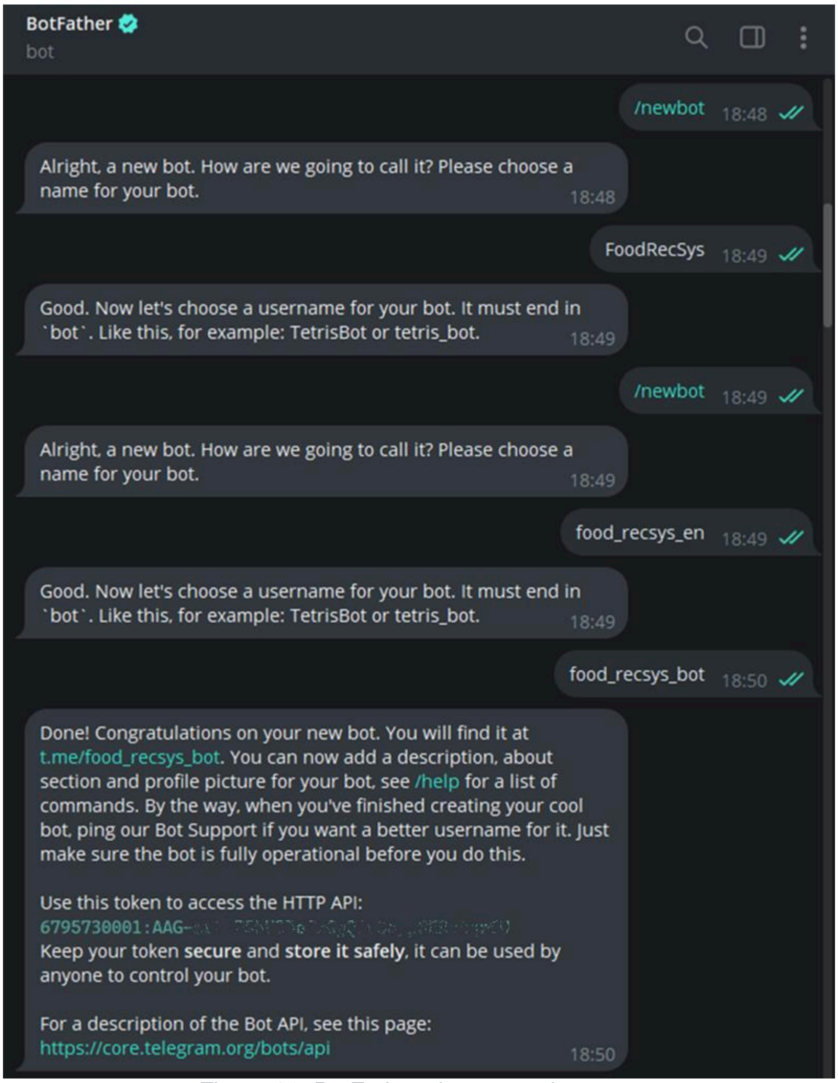


Figure 26: BotFather chat example

In the specific case of this thesis, the first part of the FoodRecommenderSys.py file declares a series of constants to define a conversation state management schema. These constants, such as "GENDER," "AGE," "MOOD," and others, are used as identifiers to indicate the current state of the user during interaction with the bot. Each state represents a specific question asked to the user regarding their information related to that constant. This is done through a ConversationHandler object, which defines all states and specifies which functions should be executed when the user is in each of these states.

For instance, if the user is in the GENDER state, it means that the user is currently responding to the gender question (and the gender function will be executed), or AGE if the chatbot is waiting for the user to provide their age (which will be processed with the age function). After defining these constants, all the functions that allow the bot's operation are set up.

The "start" function serves as the bot's initial function when a user begins a conversation using the "/start" command which it the default command when initiating a new conversation with a telegram chatbot.
This function prompts the user to choose their gender by typing "Man" or "Woman".
Following this, the chatbot presents a series of questions with corresponding buttons for user selection, allowing for information gathering. The user's gender response is stored in their information, and the subsequent "gender" function verifies and records this data. If the user's response is invalid, the chatbot remains in the gender selection state until a correct response is provided. Similar button-based interactions are used for subsequent questions regarding age, weight, and other preferences, maintaining a consistent structure throughout the conversation flow.
In each of these functions, the user's responses are stored in context objects, which will be the parameters of the request sent to the recommendation and explanation servers (an example of a request can be seen in the previous paragraph).

The process is repeated until all questions posed to the user are completed, and the last function returns a ConversationHandler.END to declare the end of the conversation states.
Before moving on to the main part, it is worth further discussing the "dialogflow_mode" function.
It uses the Google Cloud Dialogflow service to interpret the user's message and provide a response based on the detected Intent. Depending on the detected Intent (e.g., "suggestion" or "explanation goal" etc.), the bot calls the appropriate functions belonging to the

recommender_script.py and expl_script.py files described earlier to provide the requested information to the user.
We thus come to the end of the implementation part with the main program "main()." Here, the necessary objects are initialized to start the Telegram bot and manage user interactions. First, the logger is initialized, which is an object that records program events during execution, such as information log messages, warnings, or errors. In the case of this project, it is used to format and display log messages in a specific format specified by the format string '*%(asctime)s - %(name)s - %(levelname)s - %(message)s*', and it also confirms the bot's correct startup.

Next, an Updater object is created. The Updater is the main component of the python-telegram-bot library that handles interactions with the Telegram API. It requires the Telegram API access token to identify the bot, which was provided by BotFather.

Once the Updater is created, its dispatcher (event handler) is obtained. The dispatcher is responsible for routing messages from Telegram to the corresponding message or command handlers you will define later. In practice, it listens for and routes user requests to the appropriate code. Subsequently, a ConversationHandler object is created to manage the conversation flow with users. So, among the main components of the bot there are the conversational and command handlers used to manage user interactions and execute specific actions within the bot, in particular:

1. Conversational Handlers:
   These handlers manage conversations with users by defining entry points, states, and fallbacks.
   Each conversational handler has entry points that are typically triggered by command handlers or specific user actions.
   States represent different stages or steps within a conversation, and they are associated with message handlers that handle user input during those states. Fallbacks are used to handle unexpected or invalid user input during a conversation.

2. Command Handlers:

These handlers execute actions based on user commands or inputs. Each command handler is associated with a specific command (e.g., "/create", "/modify", "/clear_session") that users can input to trigger a particular action. Command handlers are typically used for tasks like initiating profile creation or modification, clearing session data, or starting specific conversation paths.

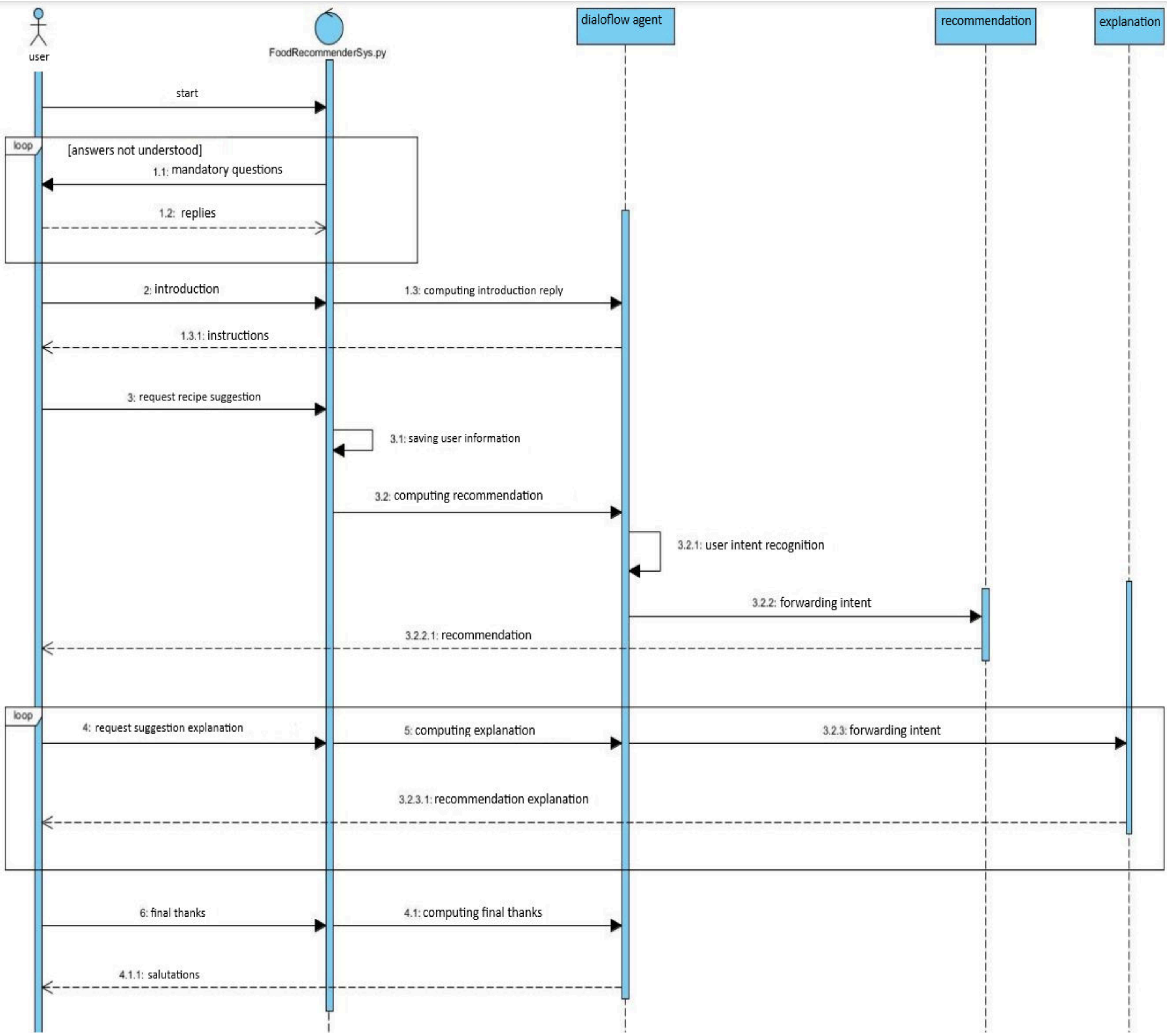


Figure 27: Sequence diagram

That being said, the bot was re-implemented for this thesis due to some issues with the previous version. In particular reimplementing the bot involved updating project requirements to address the deprecated python-telegram-bot v13.4 library, necessitating an upgrade to version 20.7[26]. This update brought significant changes to the main function logic, replacing the dispatcher with the Application component that encapsulates crucial elements for bot execution, such as the updater and dispatcher itself.

Additionally, the new version of the library required the adoption of the async-await paradigm that facilitated handling responses asynchronously. Hence, within the main module, all functions were modified to adhere to this paradigm, enabling response management to occur asynchronously by awaiting the resolution of message updates on the chat-bot side before concluding.

These modifications were essential to ensure the bot's functionality and responsiveness, optimizing user interactions and overall performance. Then also some features were enhanced, let's delve into the details of the modifications:

- The explicit comparison between recipes has been improved significantly enhancing user experience and decision-making within the system. Previously, users were presented with a mere list of facts without a clear preference between recipes, leaving the final decision solely to the user's responsibility. However, with the updated explanation module, users are now provided with a clearer preference in comparative explanations, offering a real suggestion in the comparisons. This means that the system now evaluates and highlights one of the recipes as better than the other, facilitating users in making informed choices.

[26] https://pypi.org/project/python-telegram-bot/20.7/

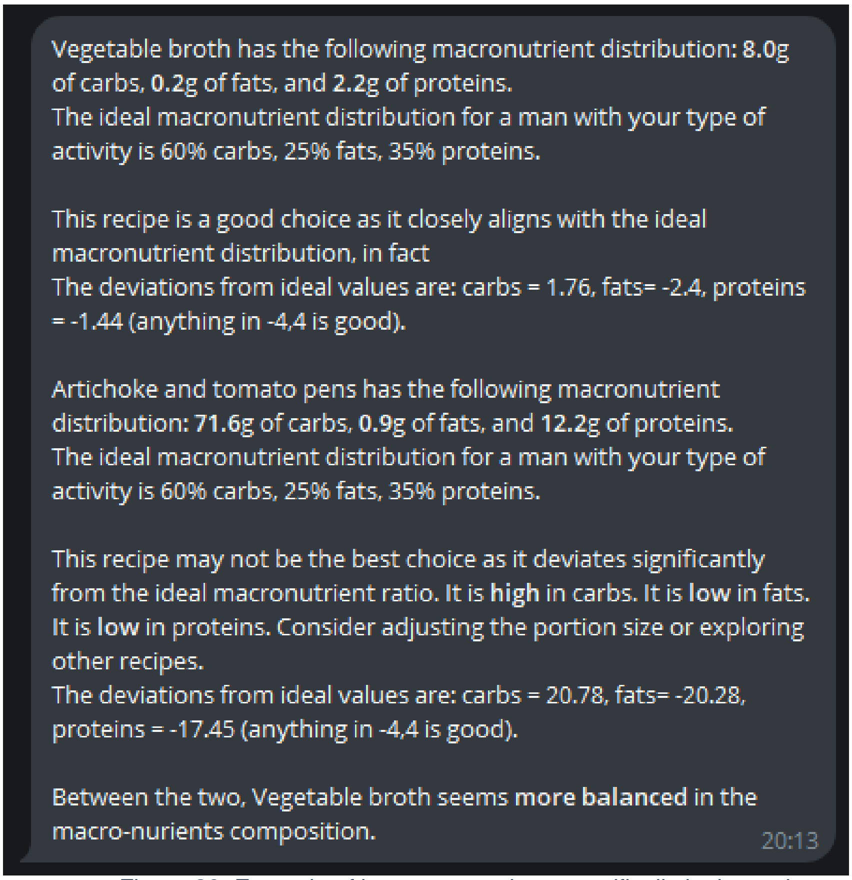


Figure 28: Example of better comparion - specifically in the end

- In the previous project, there was no provision for users to modify their data. However, the /modify command was introduced, managed through three functions and a dedicated conversational handler. In particular:
  - modify_profile(update: Update, context): It checks if the "gender" attribute is present in the user's data. If not, it replies with a message indicating that the profile hasn't been created yet and suggests using the "/create" command first. If the profile exists, it constructs a message displaying the current profile attributes and asks the user what attribute they want to modify. The function ends by returning the ATTRIBUTE state.
  - choose_attribute(update: Update, context): This function handles the user's choice of attribute to modify in their profile. It takes the user's input, converts it to lowercase, and checks if it matches any predefined attributes in the attribute_options dictionary. If there's a match, it constructs a message with options related to that attribute and sends it to the user with a keyboard for selection.

    If the user selects "none," it ends the conversation. If the input doesn't match any options, it asks the user to repeat the input. The function returns

TO_CHOICES after sending the attribute options message to the user, indicating a transition to the next step in the conversation.

- change_attribute_value(update: Update, context): this function handles various attribute changes based on user input. It retrieves the user's input text and converts it to lowercase then checks the value against a series of if-elif conditions to determine which attribute the user wants to change and performs the corresponding action. For each valid input value, it prints a debug message, sends a reply confirming the attribute change, updates the corresponding attribute in the context.user_data dictionary, and ends the conversation handler (ConversationHandler.END).

Hence, this command enables users to adjust their preferences by initiating profile modification. Through a guided process, users can select and modify specific parameters, culminating in a confirmation message to confirm the changes.

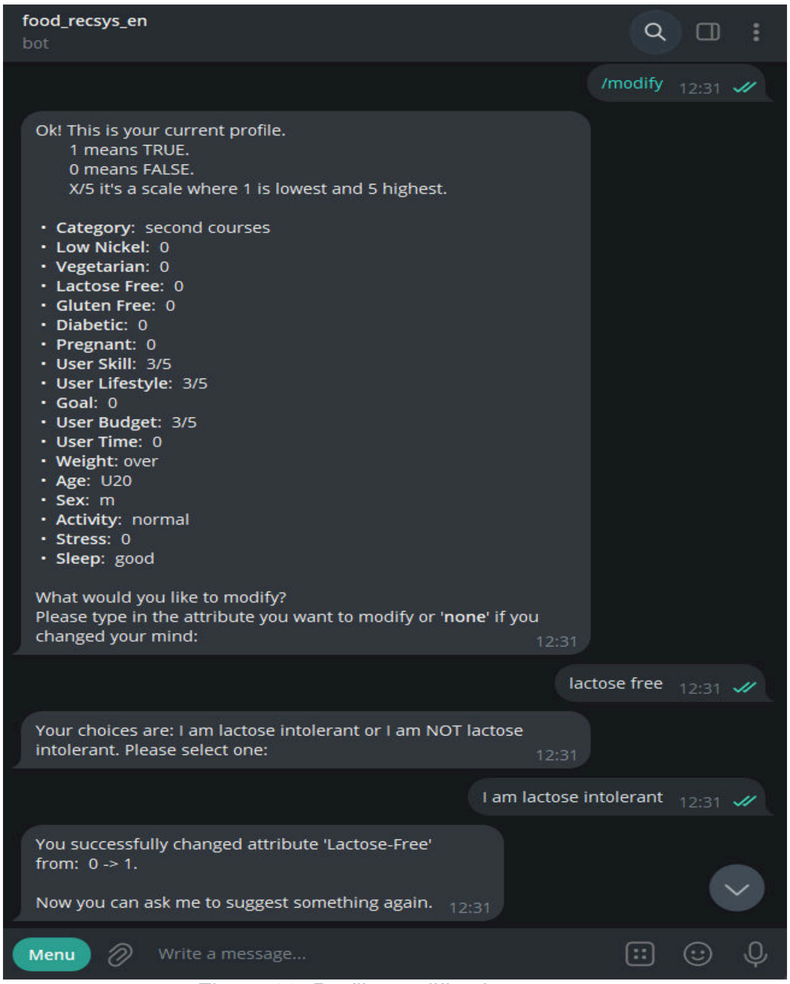


Figure 29; Profile modification process

2. The User Profiling process has been simplified by reducing the initial set of questions from 22 to around 10, focusing on gathering essential information for recommendations while minimizing user inconvenience. Default values are assigned to parameters not set by the user, typically set to their mean value. Users are informed of this defaulting practice and are given the option to modify all profile values using the /modify command. This streamlined approach ensures efficient data collection and user control over their profile settings.

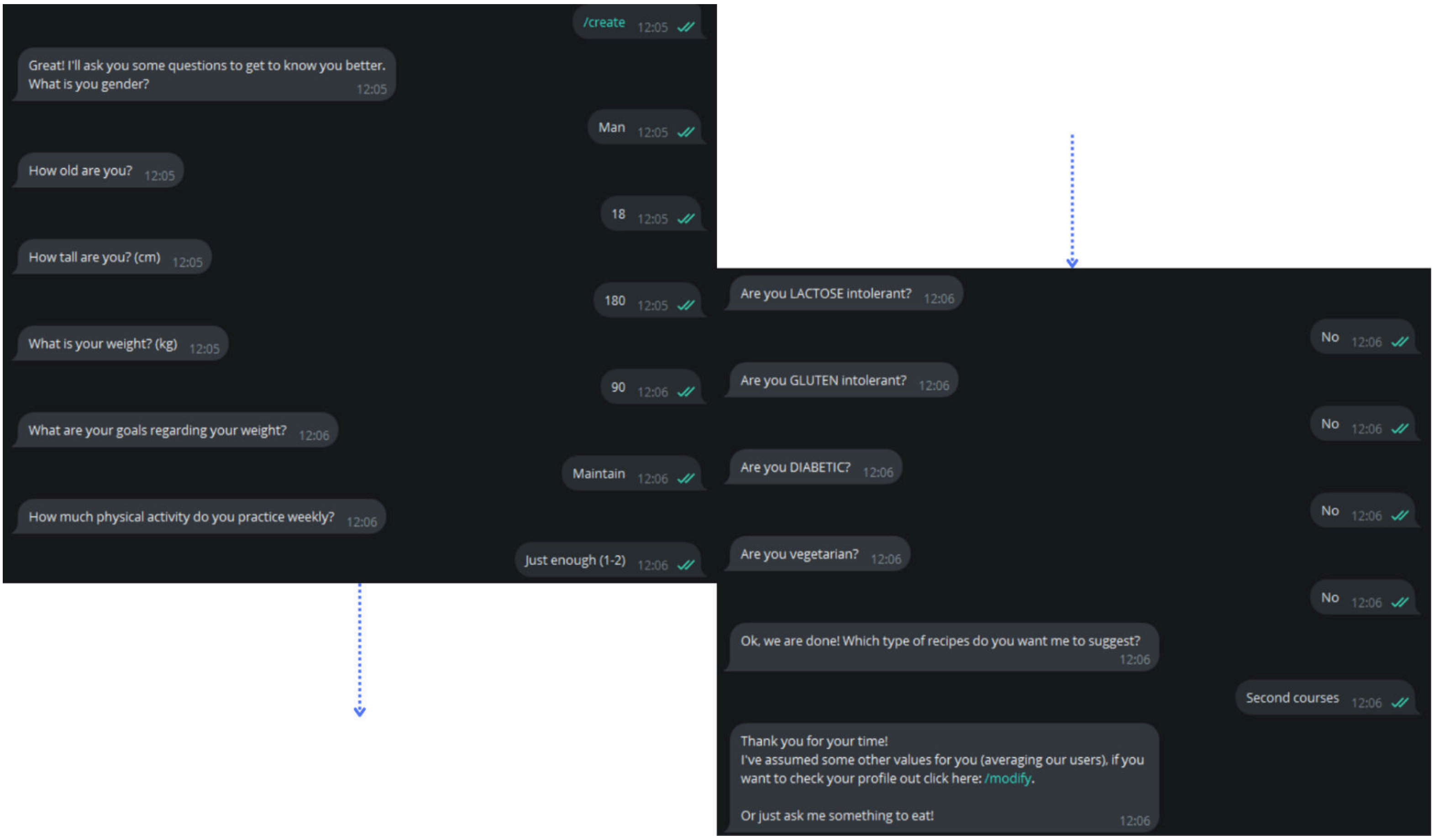


Figure 30: Streamlined profile creation process

3. In addition, a new approach has been implemented for the foodGoals explanation type, particularly focusing on the computation of daily caloric intake. In the previous version, the calculation was based on the assumption that a single meal constitutes about 40% of the daily caloric intake, which seemed somewhat inaccurate. To address this, the function has been completely reworked to calculate daily caloric intake based on gender, goals (such as losing, gaining, or maintaining weight), and

activity level. This refined approach is applied in both single explanations and comparisons, ensuring accuracy and relevance in nutritional recommendations.

Specifically: The function now assesses a recipe's suitability for a user based on their nutritional goals and activity levels. It calculates the daily calorie intake based on the user's sex, goal (lose, gain, maintain weight), and activity level (low, normal, high).

The function provides positive language and commitment in acknowledging the user's actions towards their goals. It includes social proof by mentioning that many users with similar goals and activity levels have enjoyed the recipe.

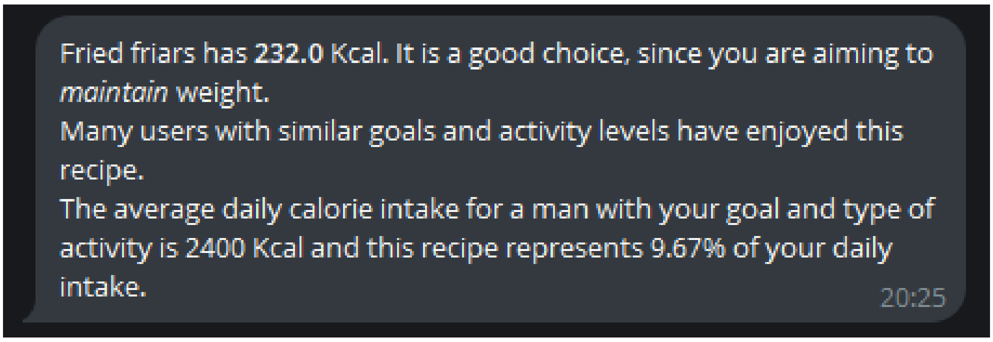


Figure 31: Example of new foodGoals

4. Furthermore, to improve user safety and sensitivity, the dopamine-based explanation and comparison type has been removed. This kind of explanations focused on the dopamine production based on the consumption of certain foods, but the decision was made after discussions with Professor Musto, the thesis supervisor, considering the potential risks involved, especially for users affected by depression or related conditions.

5. Instead, a novel method called foodMacros has been introduced to enhance explanation and comparison processes. This approach focuses specifically on the macronutrients found in recipes, namely carbohydrates, fats, and proteins, and it establishes an ideal ratio derived from relevant nutritional research. By meticulously analyzing deviations from this ideal distribution, foodMacros provides personalized recommendations based on nutritional values. This ensures a more tailored and effective approach in guiding users towards healthier eating habits. Specifically, the foodMacros method initiates with a conversion to decimals using the formula

ideal_macros["carbs|fat|proteins"] / 100. This step simplifies subsequent calculations and comparisons. The term "total_recipe_macros" denotes the sum of actual values of carbohydrates, fats, and proteins present in the recipe. Subsequently, the expected nutrient content is defined through the formula *total_recipe_macros * (ideal_macros["carbs|fat|proteins"] / 100)*. By multiplying the total recipe macros by the ideal distribution percentage, this formula computes the hypothetical nutrient content that would be present if the recipe precisely matched the ideal distribution.

Moving forward, the method defines deviation formulas as follows:

recipe_macros["carbs|fat|proteins"] -

*(total_recipe_macros * (ideal_macros["carbs|fat|proteins"] / 100)).*

These formulas calculate the deviation, which represents the disparity between the actual amount of carbohydrates, fats, or proteins in the recipe and the expected amount based on the ideal distribution. The deviation formula subtracts the expected nutrient amount from the actual content in the recipe. The resulting deviation value indicates the extent to which the actual content deviates from the expected content under the ideal distribution. Given the practical challenge of achieving a perfect match with the ideal ratio, a deviation range of [-4, 4] is considered healthy and appropriate, providing users with a realistic guideline for nutritional intake.

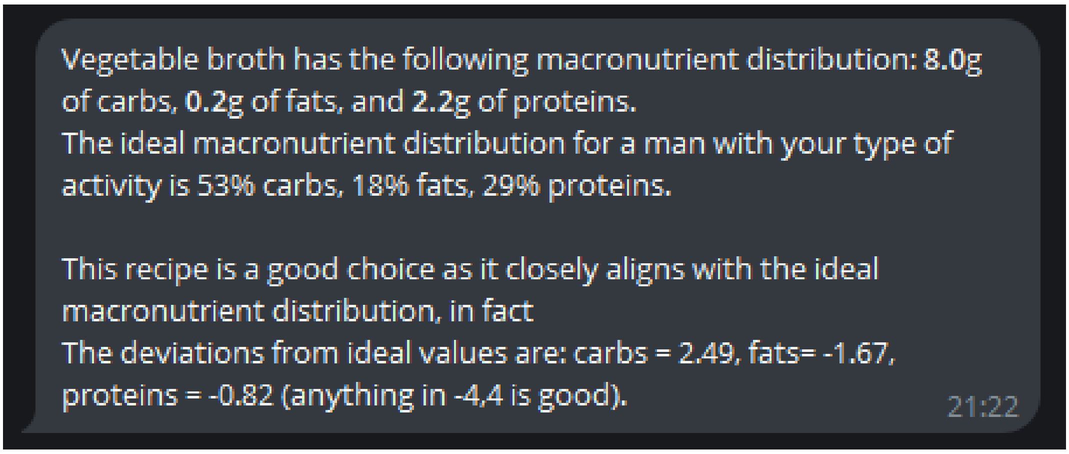


Figure 32: Example of foodMacros

6. Finally, since there was no provision for users to express a preference for the ingredients of the recipe to suggest. It was implemented a new feature: the User-specific Request, in which users can ask the bot for a recommendation based on a specific ingredient or type of dish they provide. This is implemented just through a

new intent that takes into account the Dialogflow entity that represents the desired ingredient (or type of dish). The bot tries to fulfil the request by searching through the top 15 ranked suggestions. Upon receiving the request, the bot identifies the Dialogflow entity representing the ingredient or type of dish and transforms it into lowercase and singular form using the library inflect[27]. This transformation is also applied to the ingredients of each recipe checked along to their titles. If the bot finds a match, either in the title or ingredients of a recipe, it will return it as a suggestion to the user. However, if no suitable recipe is found based on the specified parameters, the bot will inform the user accordingly.

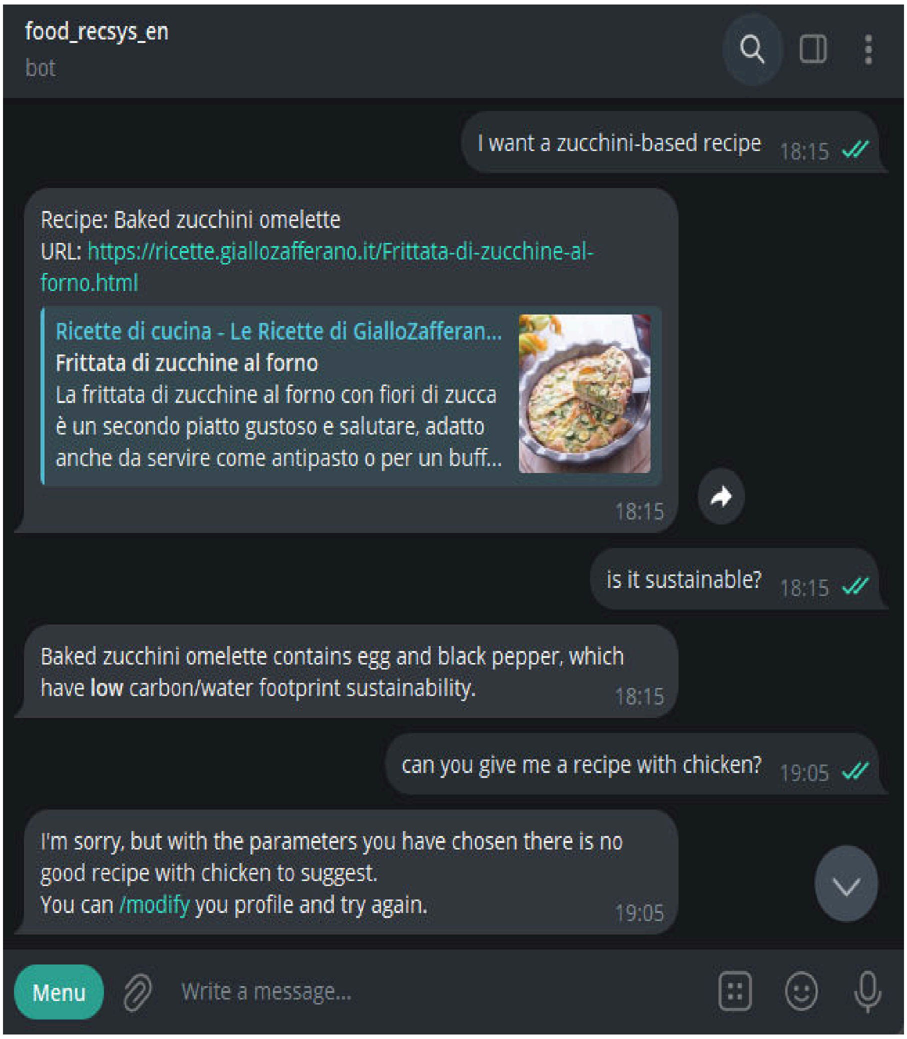


Figure 33: User-specific request

[27] https://pypi.org/project/inflect/

3.2.4. DialogFlow

Google Dialogflow is a natural language understanding platform that allows developers to build conversational interfaces such as chatbots. It utilizes machine learning and AI techniques to understand and process user input in natural language, enabling human-like interactions between users and applications.

Figure 34: Dialogflow logo

In our endeavour, DialogFlow has been used for:

1. Intent Recognition:

   Dialogflow can recognize user intents based on input text or voice, allowing developers to define how the system should respond to different user requests. The new DialogFlow agent not only maintains basic chitchat capabilities and welcoming features but also introduces a more organized structure by categorizing intents into five specific areas. These areas include:

   - Introduction:

     Provides general information about the bot and directs users to commands like /create for profile creation or /modify for profile modification.
   - Suggestion:

     Initializes the recommendation process after the user creates a profile, sending parameters to the recommendation system to generate recipe suggestions.
   - Change Suggestion:

     Allows users to request a change in the recommended recipe, providing an alternative if the first suggestion is not suitable.
   - Explanation:

     Enables users to inquire about the reasons behind a recommendation, nutritional values, cooking difficulty, and other intrinsic characteristics of the suggested recipe.
   - Comparison:

     Provides users with the option to compare two different recipes based on various criteria, helping them make informed decisions about their choices.

2. Entity Recognition:

   Entity recognition plays a crucial role in enhancing the functionality of the bot, particularly in understanding and responding to user-specific requests. One key aspect of entity recognition is its ability to identify and extract various types of entities from user input. In the context of the bot's functionality related to user-specific requests for ingredient-based recipes, entity recognition becomes especially important. The bot utilizes entity recognition to identify and extract the specific ingredients mentioned by the user in their request.

3. Context Management:

   Dialogflow maintains context during conversations, enabling more contextually relevant responses and better handling of follow-up questions.

To be able to use Dialogflow, Google Cloud is fundamental. Google Cloud is a suite of cloud computing services provided by Google, offering a wide range of solutions for building, deploying, and managing applications and data in the cloud. It provides infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) offerings, enabling businesses and developers to leverage scalable and flexible cloud resources.

In our case, through Google Cloud, we created a service account to generate credentials specifically for the Python connection between Dialogflow and the bot. To maintain confidentiality and privacy, new credentials were implemented and carefully managed. They were added to the .gitignore file, which prevents them from being tracked and uploaded to version control systems. This steps ensure secure and authenticated access, allowing the bot to interact with Dialogflow's APIs and services effectively.

3.2.5. Explanations & Comparisons

In discussing explanations and comparisons, it's essential to first understand the intent recognizer of Dialogflow. The intent recognizer, powered by machine learning models and a list of training phrases, plays a crucial role in interpreting user intents specifically designed for the project. When a user sends a request to the chatbot, Dialogflow's Intent Recognizer analyzes the text and matches it with the existing intent's training phrases. Once the corresponding intent is identified, the system proceeds to the next module to process the request and provide a suitable response.

Responses can be directly generated by the Dialogflow platform using the dedicated Responses module. Developers can create a list of responses within the specific Intent section. Alternatively, as in this project's case, the correct intent is identified to respond to the user by forwarding the request to an external server. The request is then processed, the result is structured into natural language, and finally sent to the user in text format.

This workflow showcases the seamless integration of intent recognition and response generation in Dialogflow. By leveraging machine learning algorithms and a well-defined training dataset, the platform efficiently handles user queries, ensuring accurate interpretation of intents and timely delivery of relevant responses. This approach not only enhances user experience by providing personalized interactions but also offers flexibility for developers to customize responses based on specific project requirements.

Every explanation and comparison is associated to a specific intent.In this way it is possible to accurately identify what the users are asking for and provide it to them. In particular we have:

Intent	Description	Triggering messages
Introduction	Provides some general information about the bot.	hi, hello, who are you, what do you do, etc.
Suggestion	Initialize the suggestion process, providing the user with a suitable recipe	Suggest something to eat
Change suggestion	It can be called whenever the user does not find suitable the first suggestion (up to two times)	give me another one, I don't like this, do you have another one?, can you change it?
Stop suggestion	Ends the change suggestion process	*automatically triggered*
Specific suggestion	Provides the users with a recipe with an ingredient they request	I want a *ingredient*-based recipe, Can you give me a recipe with *ingredient*?
Expl age	Explanation about the healthiness and	is it adequate for my age?

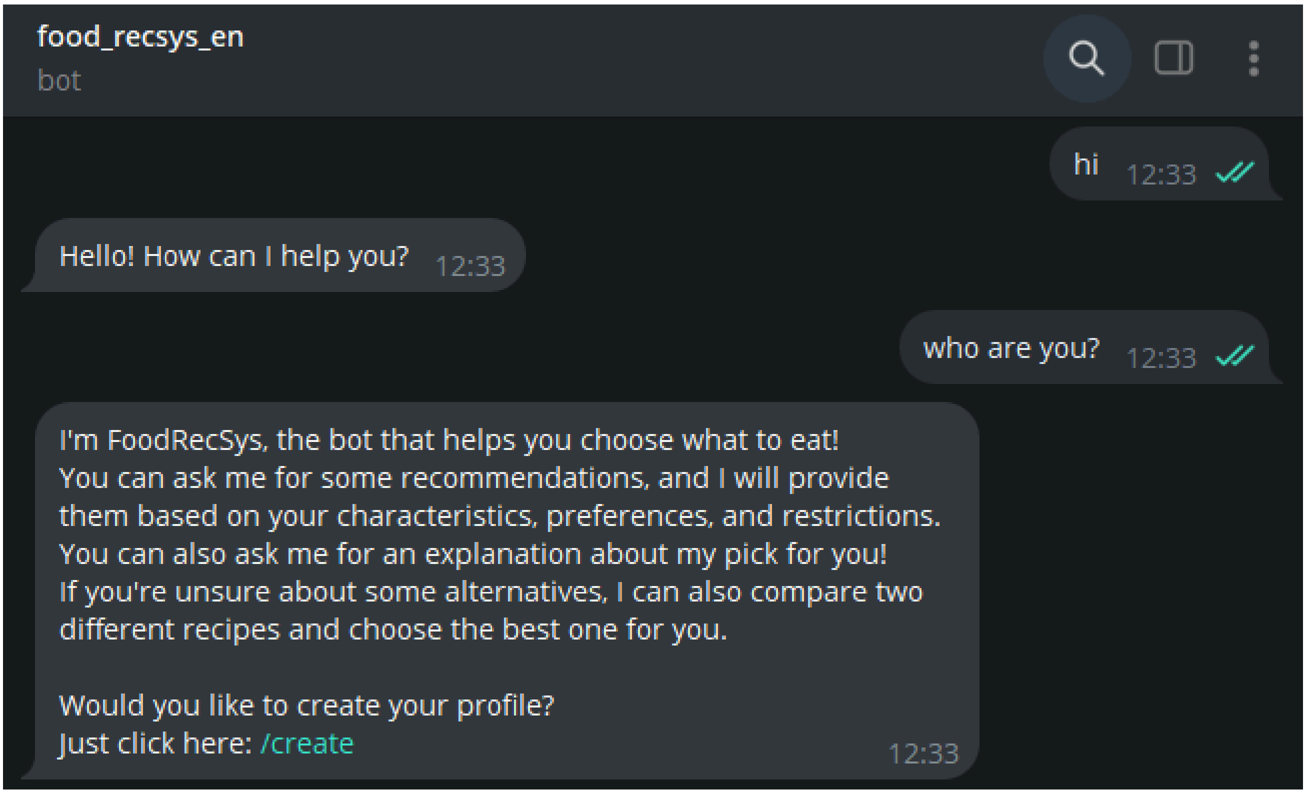
	appropriateness of the recipe based on the age group the user belongs to	
Expl cost	Explanation about the appropriateness of the recipe based on its cost	how much does it cost?
Expl goal	Explanation about the healthiness and appropriateness of the recipe based on the user goals	is it in line with my goals?
Expl health-benefits	Explanation concerning the possible benefits of the recipe	does it have some healthy benefits?
Expl health-risks	Explanation concerning the possible risks of the recipe	what are its health risks?
Expl lifestyle	Explanation about the appropriateness of the recipe with respect to the user's lifestyle	is it good for my lifestyle?
Expl macros	Explanation about the healthiness of the recipe with respect to an ideal macros distribution	are its macros good?
Expl popularity	Explanation about the popularity of the recipe with respect to the ratings on giallozafferano.it	how popular is it?
Expl restriction	Explanation about the appropriateness of the recipe with respect to the user's restrictions	is it ok for my restrictions?
Expl skill	Explanation about the appropriateness of the recipe with respect to the user's cooking skills	is it easy to cook?
Expl seasonality	Explanation about the level of seasonality of the recipe ingredients	is it seasonal?
Expl sustainability	Explanation about the level of sustainability of the recipe ingredients	is it sustainable?
Expl time	Explanation about the appropriateness of the recipe with respect to the user available time	do I have enough time?
Expl meal	Plain description of the recipe	what are its characteristics?
Expl meal check	Provides a comparison between various nutrients and 40% of their respective	what are it nutritional values?

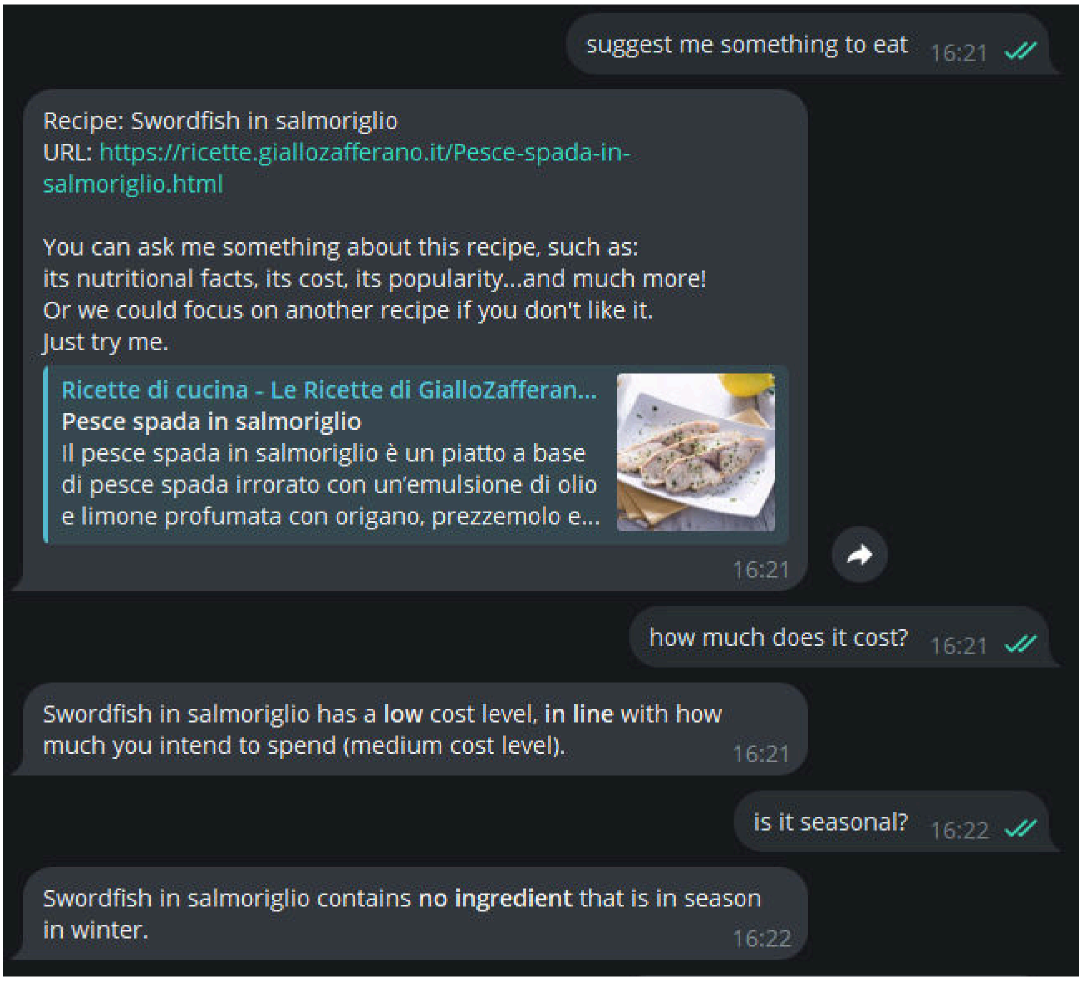
	daily reference intake	
Comp age	Comparison between two recipes taking into account the users' age	compare them according to my age
Comp cost	Comparison between two recipes taking into account the users' cost restriction	compare them according to their costs
Comp goal	Comparison between two recipes taking into account the users' goals	compare them according to my goals
Comp health-benefits	Comparison between two recipes taking into account the health-benefits of the recipes	compare them according to their health benefits
Comp health-risks	Comparison between two recipes taking into account the health-risks of the recipes	compare them according to their health risks
Comp lifestyle	Comparison between two recipes taking into account the users' lifestyle	compare them according to my lifestyle
Comp macros	Comparison between two recipes taking into account the macros distributions of the two recipes	compare them according to their macros
Comp popularity	Comparison between two recipes taking into account the recipes' ratings	compare them according to their popularity\rating
Comp restriction	Comparison between two recipes taking into account the users' restriction	compare them according to my restrictions
Comp skill	Comparison between two recipes taking into account the users' cooking skill	compare them according to the skill required to cook them
Comp seasonality	Comparison between two recipes taking into account the seasonality of the recipes' ingredients	compare them according to their seasonality
Comp sustainability	Comparison between two recipes taking into account the sustainability of the recipes' ingredients	compare them according to their sustainability
Comp time	Comparison between two recipes taking into account the users' time restrictions	compare them according to cooking time
Comp meal	Plain descriptions of the two recipes	compare them according to their nutritional values
Compl meal check	Comparison between recipes taking into account various nutrients and 40% of their respective daily reference intake	compare them according to their nutritional values

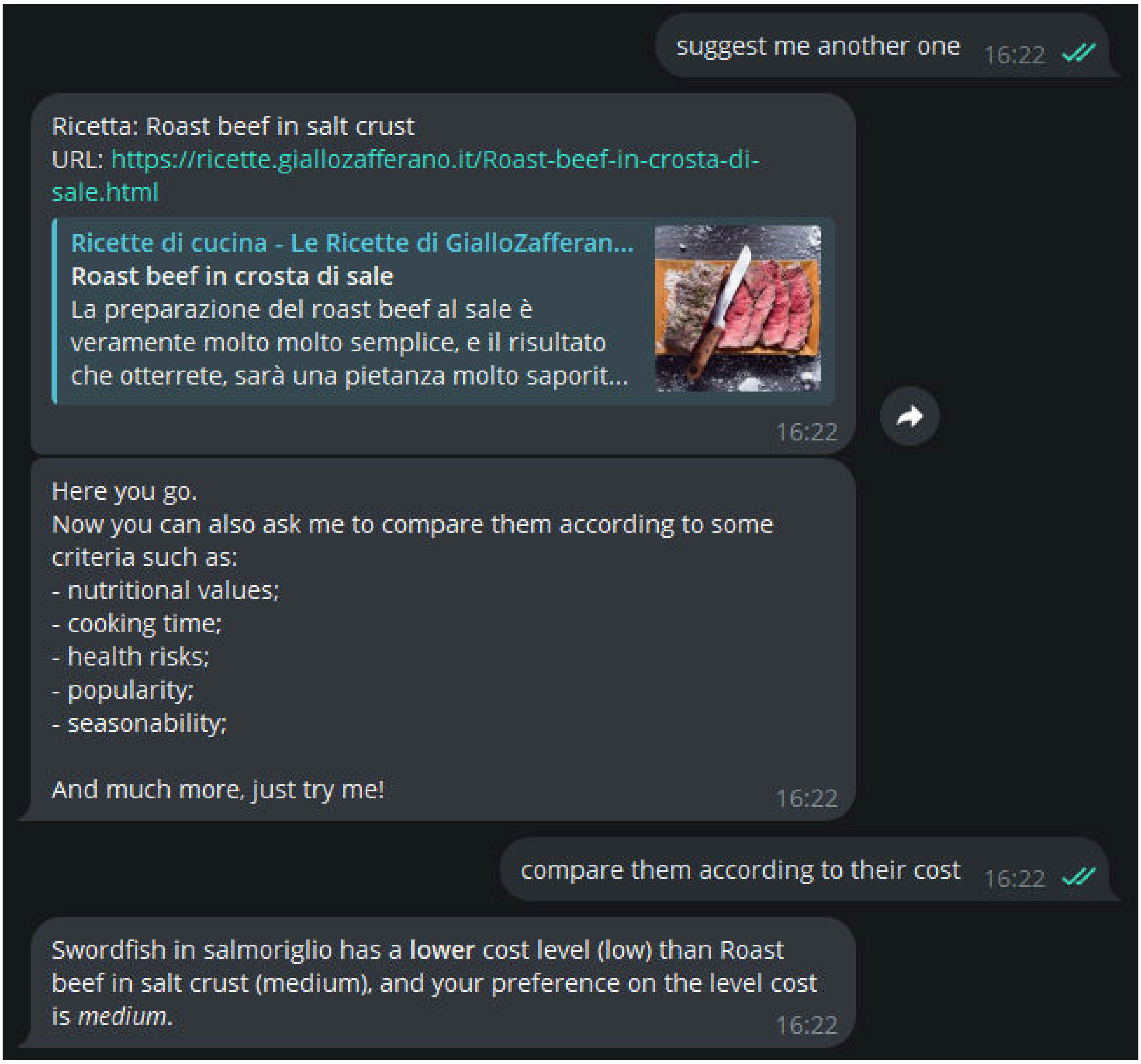
```python
if intent == "Suggestion":
    await Recommendation.suggerimento(update, context)
if intent == "Change suggestion 1":
    await Recommendation_due.altro_suggerimento2(update, context)
if intent == "Change suggestion 2":
    await Recommendation_tre.altro_suggerimento3(update, context)
if intent == "Explanation check":
   await Spiegazione.controllo_piatto(update, context)
if intent == "Explanation popularity":
   await Spiegazione.spiegazione_popolarita(update, context)
if intent == "Explanation meal":
    await Spiegazione.spiegazione_piatto(update, context)
if intent == "Explanation skill":
    await Spiegazione.spiegazione_skill_cucina(update, context)
if intent == "Explanation goals":
    await Spiegazione.spiegazione_obiettivo(update, context)
if intent == "Explanation health-benefit":
    await Spiegazione.spiegazione_benefici_salute(update, context)
if intent == "Explanation health-risk":
    await Spiegazione.spiegazione_rischi_salute(update, context)
if intent == "Explanation cost":
    await Spiegazione.spiegazione_costo(update, context)
if intent == "Explanation age":
    await Spiegazione.spiegazione_eta(update, context)
if intent == "Explanation restriction":
    await Spiegazione.spiegazione_restrizioni(update, context)
if intent == "Explanation lifestyle":
    await Spiegazione.spiegazione_lifestyle(update, context)
if intent == "Explanation time":
    await Spiegazione.spiegazione_tempo(update, context)
if intent == "Explanation macros":
    await Spiegazione.spiegazione_macros(update, context)
if intent == "Explanation sustainability":
    await Spiegazione.spiegazione_sustainability(update, context)
if intent == "Explanation seasonability":
    await Spiegazione.spiegazione_seasonability(update, context)

if intent == "Comparison two meals":
   await Spiegazione.controllo_piatto_due_piatti(update, context)
if intent == "Comparison popularity":
    await Spiegazione.spiegazione_popolarita_due_piatti(update, context)
if intent == "Comparison meal checks":
    await Spiegazione.spiegazione_piatto_due_piatti(update, context)
if intent == "Comparison skill":
    await Spiegazione.spiegazione_skill_cucina_due_piatti(update, context)
if intent == "Comparison goal":
    await Spiegazione.spiegazione_obiettivi_due_piatti(update, context)
if intent == "Comparison health-benefit":
    await Spiegazione.spiegazione_benefici_salute_due_piatti(update, context)
if intent == "Comparison health-risk":
    await Spiegazione.spiegazione_rischi_salute_due_piatti(update, context)
if intent == "Comparison cost":
    await Spiegazione.spiegazione_costo_due_piatti(update, context)
if intent == "Comparison age":
    await Spiegazione.spiegazione_eta_due_piatti(update, context)
if intent == "Comparison restriction":
    await Spiegazione.spiegazione_restrizioni_due_piatti(update, context)
if intent == "Comparison lifestyle":
    await Spiegazione.spiegazione_lifestyle_due_piatti(update, context)
if intent == "Comparison time":
    await Spiegazione.spiegazione_tempo_due_piatti(update, context)
if intent == "Comparison macros":
    await Spiegazione.spiegazione_macros_due_piatti(update, context)
if intent == "Comparison sustainability":
    await Spiegazione.spiegazione_sustainability_due_piatti(update, context)
if intent == "Comparison seasonability":
    await Spiegazione.spiegazione_seasonability_due_piatti(update, context)
```

Figure 35: Intent-function list

3.2.6. Usage examples






suggest me another one 16:22
Ricetta: Roast beef in salt crust
URL: https://ricette.giallozafferano.it/Roast-beef-in-crosta-di-sale.html
Ricette di cucina - Le Ricette di GialloZafferan...
Roast beef in crosta di sale
La preparazione del roast beef al sale è veramente molto molto semplice, e il risultato che otterrete, sarà una pietanza molto saporit...
16:22
Here you go.
Now you can also ask me to compare them according to some criteria such as:
- nutritional values;
- cooking time;
- health risks;
- popularity;
- seasonability;
And much more, just try me!
16:22
compare them according to their cost 16:22
Swordfish in salmoriglio has a **lower** cost level (low) than Roast beef in salt crust (medium), and your preference on the level cost is *medium*.
16:22

4. Experimental Evaluation

The experiment for this thesis is structured with the same framework as the bot, but a simplified version is implemented. Most intents have been removed, leaving only the default fallback and welcome intents along with the introduction.

Additionally, a new commandHandler "/get_suggestions" has been added, which will initiate the actual experiment process.

Our research question is: "Can we influence our users' perception of recipes to encourage healthier eating habits by providing them with more information about the recommended recipes?"

To find an answer to this question, the user will follow the following process during the experiment:

1. To access the bot, users can click on the following link: https://t.me/food_recsys_bot. This will take them directly to the bot on Telegram, where they will receive a welcome message.

2. After entering the bot, users can create their user profile using the command /create. A brief questionnaire consisting of 10 mandatory questions will guide them in providing their basic information.
   If users wish to make changes to their profile, they can use the command /modify after completing the mandatory questions. By typing the name of the attribute they wish to modify, the bot will display the available options. Alternatively, they can type "none" if no changes are desired.

3. Healthiness assessment: Users will receive recommendations for healthy recipes and will be asked to express their opinion on the healthiness of each recipe. The bot will then provide additional explanations, and users will need to reconsider their opinion based on the information provided.

4. Sustainability assessment: The process focuses on the sustainability of recipes rather than healthiness, following similar steps as described above.

5. Upon completing the recommendations process, the bot will display the user's unique ID and provide a link to the final questionnaire. Users are encouraged to participate in the questionnaire to share their opinions and contribute to the research.

4.1. Amazon Web Services

The experiment is currently available and hosted on an AWS machine.

Figure 36: Amazon Web Services logo

Amazon Web Services (AWS)[28] is a comprehensive cloud computing platform provided by Amazon, offering a wide range of services that enable businesses and individuals to build, deploy, and manage applications and infrastructure in the cloud. In particular, AWS offers a vast array of cloud services across categories such as compute, storage, databases, networking, machine learning, artificial intelligence, analytics, security, and more. Some popular services include Amazon EC2 (Elastic Compute Cloud) for virtual servers, Amazon S3 (Simple Storage Service) for object storage, Amazon RDS (Relational Database Service) for managed databases.

Our machine on Amazon Web Services (AWS) is an Amazon EC2 instance, specifically using the t2.micro instance type and running the Ubuntu operating system with 1GB of RAM and 30GB of storage.

An Amazon EC2 (Elastic Compute Cloud)[29] instance is a virtual server provided by AWS that allows users to run applications and services in the cloud. EC2 instances are highly customizable, allowing users to choose specifications such as CPU, memory, storage, and networking capabilities based on their workload requirements.

[28] https://aws.amazon.com/
[29] https://aws.amazon.com/it/ec2/

The t2.micro instance type is one of the many instance types available on AWS and it is designed for general-purpose computing and is suitable for a wide range of applications, including development and testing environments, small-scale applications, and low-traffic websites.

4.2. The Experiment

In addition to the questions previously addressed, this experiment will also collect data on users' levels of knowledge and interest regarding the healthiness and sustainability of the recipes. These metrics will serve as a basis for comparison in interpreting the results. The experiment is structured as follows:

After obtaining mandatory user information and completing profile creation, the main experiment commences. Users will initially receive 6 suggestions, comprising a first course, a second course, and a dessert in both the healthiness and sustainability categories.

This process was implemented through a new conversationHandler that takes care of the entire flow, specifically:

The conversation handler defines a structured flow for a chatbot or conversational interface related to suggesting and evaluating recipes based on their healthiness and sustainability:

- entry points: The conversation starts when the user triggers the "get_suggestions" command, which is handled by the "healthiness_initialiation" function.
- states: Defines different states or stages of the conversation, each corresponding to a particular point in the interaction flow. Here are the states and their corresponding message handlers:
    - PRE_FIRST_COURSE_HEALTINESS: User is expected to provide unconditioned input for the first course's healthiness.
    - POST_FIRST_COURSE_HEALTINESS: After receiving the user's input for the first course's healthiness, the conversation moves to the explanation for the first course and now expects a conditioned input from the user.

- PRE_SECOND_COURSE_HEALTINESS: User provides unconditioned input for the second course's healthiness.
- POST_SECOND_COURSE_HEALTINESS: After receiving the user's input for the second course's healthiness, the conversation moves to the explanation for the second course and now expects a conditioned input from the user.
- PRE_DESSERT_HEALTINESS: User provides unconditioned input for the dessert's healthiness.
- POST_DESSERT_HEALTINESS: After receiving the dessert's healthiness input the conversation moves to the explanation for the second course and now expects a conditioned input from the user.
- PRE_FIRST_COURSE_SUSTAINABILITY: User provides unconditioned input for the first course's sustainability.
- POST_FIRST_COURSE_SUSTAINABILITY: After the first course's sustainability input, the conversation moves to the explanation for the first course and now expects a conditioned input from the user.
- PRE_SECOND_COURSE_SUSTAINABILITY: User provides unconditioned input for the second course's sustainability.
- POST_SECOND_COURSE_SUSTAINABILITY: After the second course's sustainability input, the conversation moves to the explanation for the first course and now expects a conditioned input from the user.
- PRE_DESSERT_SUSTAINABILITY: User provides unconditioned input for the dessert's sustainability.
- POST_DESSERT_SUSTAINABILITY: After receiving the dessert's sustainability input, the conversation reaches the end of the experiment.

- fallbacks: In case a user input doesn't match any of the expected states or handlers the users will be prompted with an error message.

Hence, upon receiving each suggestion, users are prompted to rate its healthiness/sustainability on a scale from very unhealthy/unsustainable to very healthy/sustainable. Subsequently, users are presented with an explanation elaborating on the recipe, providing additional information.

After reviewing the explanation, users are asked to rate the recipe again, taking into account the new information. This process is repeated for all 6 suggestions (3 assessing healthiness, 3 sustainability).

Regarding healthiness, the bot will randomly provide one of 3 types of explanations:

- Expl Macros,
- Expl Goals,
- Expl Meal Check.

In the sustainability section, the bot will either present Expl Sustainability or a new explanation listing only the recipe ingredients.

```
path_handler = ConversationHandler(
    entry_points=[CommandHandler("get_suggestions", healthiness_initialiation)],
    states={
        PRE_FIRST_COURSE_HEALTINESS: [MessageHandler(filters.TEXT, first_suggestion_healthiness_explanation)],
        POST_FIRST_COURSE_HEALTINESS: [MessageHandler(filters.TEXT, second_suggestion)],
        PRE_SECOND_COURSE_HEALTINESS: [MessageHandler(filters.TEXT, second_suggestion_healthiness_explanation)],
        POST_SECOND_COURSE_HEALTINESS: [MessageHandler(filters.TEXT, third_suggestion)],
        PRE_DESSERT_HEALTINESS: [MessageHandler(filters.TEXT, third_suggestion_healthiness_explanation)],
        POST_DESSERT_HEALTINESS: [MessageHandler(filters.TEXT, fourth_suggestion)],

        PRE_FIRST_COURSE_SUSTAINABILITY: [MessageHandler(filters.TEXT, fourth_suggestion_sustainability_explanation)],
        POST_FIRST_COURSE_SUSTAINABILITY: [MessageHandler(filters.TEXT, fifth_suggestion)],
        PRE_SECOND_COURSE_SUSTAINABILITY: [MessageHandler(filters.TEXT, fifth_suggestion_sustainability_explanation)],
        POST_SECOND_COURSE_SUSTAINABILITY: [MessageHandler(filters.TEXT, sixth_suggestion)],
        PRE_DESSERT_SUSTAINABILITY: [MessageHandler(filters.TEXT, sixth_suggestion_sustainability_explanation)],
        POST_DESSERT_SUSTAINABILITY: [MessageHandler(filters.TEXT, end_of_experiment)],
    },
    fallbacks=[MessageHandler(filters.TEXT, unknown)],
)
```

Figure 37: ConversationalHandler for the experiment

The objective is to evaluate whether the bot's explanations influence users' perceptions of the recipes. All user input and selections are recorded in a CSV file for later analysis.

Upon completion of the process, users receive their unique Telegram ID and a link to a Google Form hosting the final questionnaire, structured as follows[30]:

Question	construct	answers
1. Please choose your education level:	Demographic	• Primary School • High School • Bachelor's degree • Master's degree

30 Roberto Polillo «Facile da usare, una moderna introduzione all'ingegneria dell'usabilità», 2010.

		• Ph.D. or more
2. How would you rate yourself as a computer user?	Demographic	• No experience • Beginner • Average • Advanced
3. Have you ever used a recommender system before?	Demographic	• Yes • No • Maybe
4. How frequently have you used conversational agents and digital assistants?	Demographic	• Never • Very infrequently (a few times overall) • Infrequently (a few times a month) • Moderately (1-3 times a week) • Regularly (Daily)
5. The items recommended to me match my interests:	Recommendation accuracy	• Strongly disagree • Disagree • Neutral • Agree • Strongly Agree
6. This recommender system helped me discover new products:	Novelty	• Strongly disagree • Disagree • Neutral • Agree • Strongly Agree
7. This recommender system provided me with unexpected but useful suggestions:	Serendipity	• Strongly disagree • Disagree • Neutral • Agree • Strongly Agree
8. The information provided for the recommended items is sufficient for me to make a decision:	Interface sufficiency	• Strongly disagree • Disagree • Neutral • Agree • Strongly Agree
10. The recommender can be trusted	Trust and reliability	• Strongly disagree • Disagree • Neutral

		• Agree • Strongly Agree
11. Overall, I am satisfied with the recommender:	Overall satisfaction	• Strongly disagree • Disagree • Neutral • Agree • Strongly Agree
12. After reading the explanations about the healthiness and sustainability of the recipe, I am more informed about these aspects:	Information Clarity and Effectiveness	• Strongly disagree • Disagree • Neutral • Agree • Strongly Agree
13. Do you think the explanations provided by the system were clear and understandable regarding the healthiness and sustainability of the recipe?	Explanation Clarity	• Yes, very clear and understandable • Yes, moderately clear and understandable • Neutral • No, not very clear and understandable • No, not clear and understandable at all
14. After receiving the explanations, would you change your eating or consumption habits to promote healthiness and sustainability?	Behavioural Intentions	• Yes, I would certainly give it a try • I am open to making changes following the explanations • I am not sure • I might not make any changes • No, I don't think so
15. After receiving the explanations, do you believe that the recipes recommended by the system have a different healthiness and sustainability value than you initially thought?	Perceived Impact of Information	• Yes, I am convinced • I think there might be some differences • I don't know • I am not convinced by the differences • No, my opinion remains the same

5. Results

In this paragraph, all the data collected during the experimentation phase will be analyzed, particularly the responses provided by users in the final questionnaire and other relevant data derived from the actual usage of the bot. The final questionnaire was designed to assess users' experiences in interacting with the bot, encompassing aspects such as ease of use, overall satisfaction, but more importantly explanation clarity and perceived impact of information.

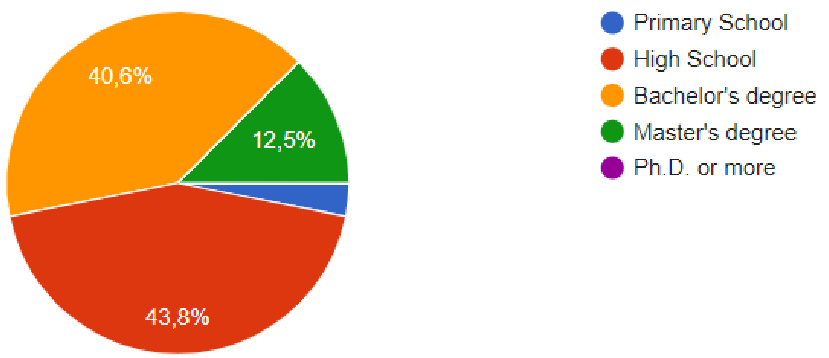
Hence, in addition to the quantitative and demographic responses gathered through the questionnaire, qualitative feedback provided by users has also been examined to gain a deeper understanding of their experiences and perceptions.

Furthermore, bot usage data, including the users' profiles and opinions, have been considered to evaluate the effectiveness and efficiency of the implemented system.

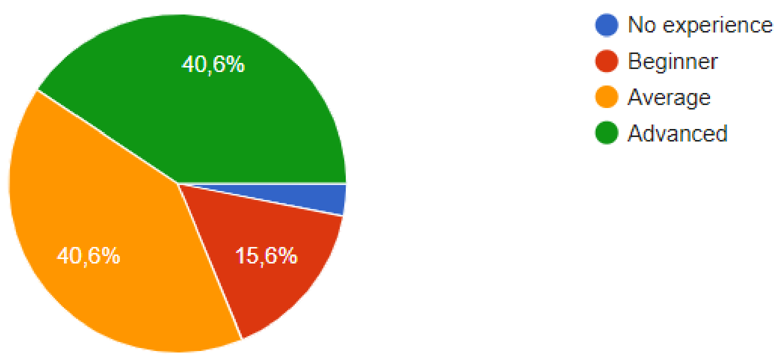
5.1 Questionnaire

The questionnaire was submitted to 32 users and the initial questions provide insight into their demographics.

1. Please choose your education level:

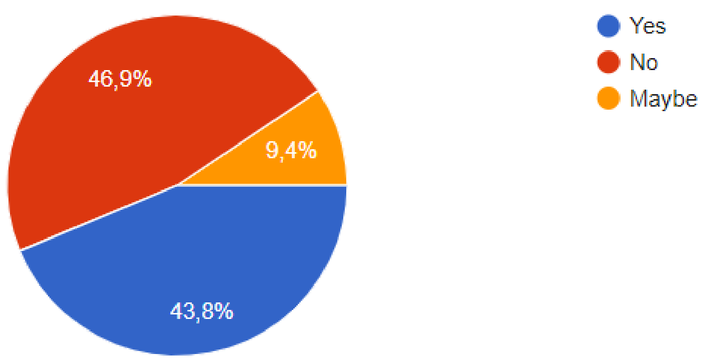


2. How would you rate yourself as a computer user?



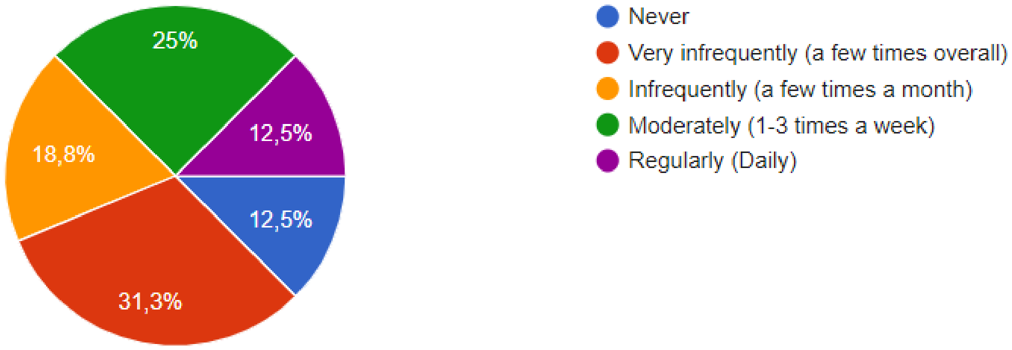
It's notable that while our users demonstrate proficiency in computer usage, only about half of them can accurately recognize a recommender system. This observation is intriguing given the widespread utilization of recommender systems like those found in Amazon or Netflix; a significant portion of users seem unaware of their presence.

3. Have you ever used a recommender system before?



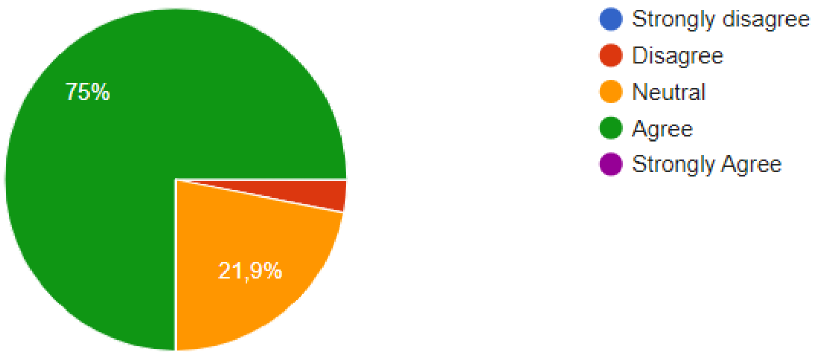
Further examination reveals a diverse user base, encompassing individuals who are already familiar with conversational agents as well as those who have never interacted with them.

4. How frequently have you used conversational agents and digital assistants?



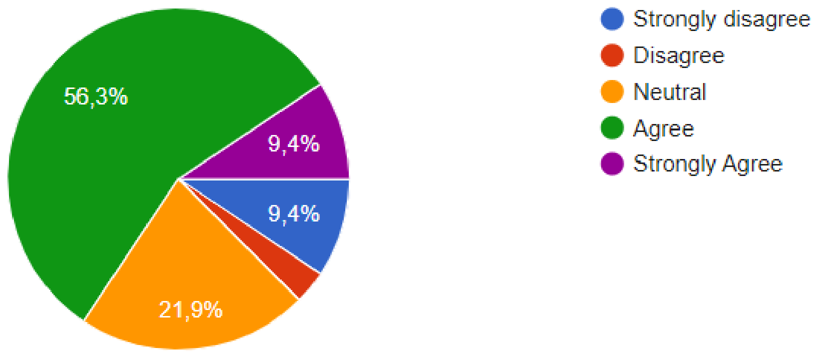
Additionally, we observe a strong alignment between the recommendations and the users' interests.

5. The items recommended to me match my interests:



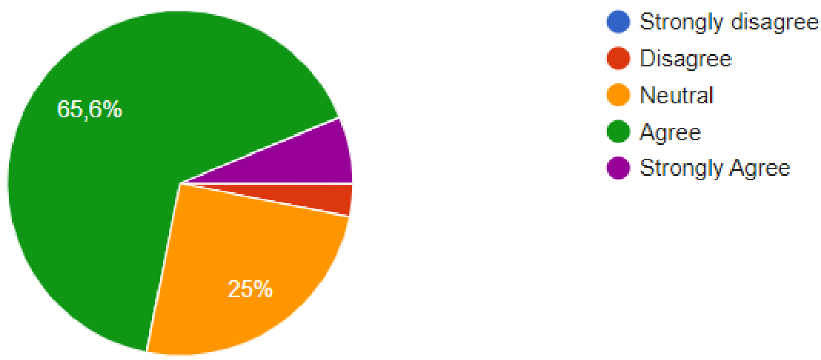
Moreover, the recommender system appears to facilitate users in discovering new products:

6. This recommender system helped me discover new products:



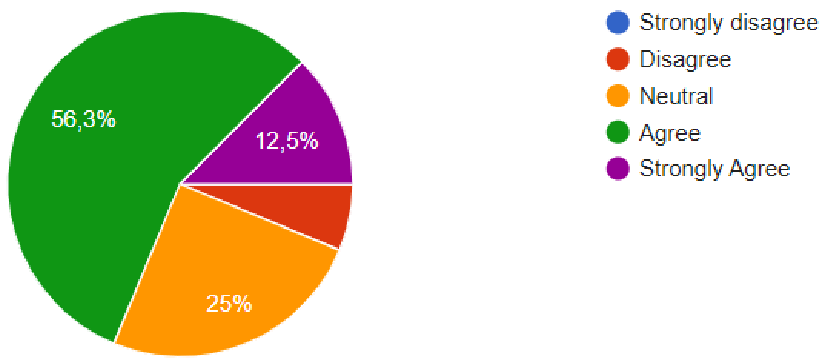
Similar percentages are evident when considering serendipitous discoveries.

7. This recommender system provided me with unexpected but useful suggestions:



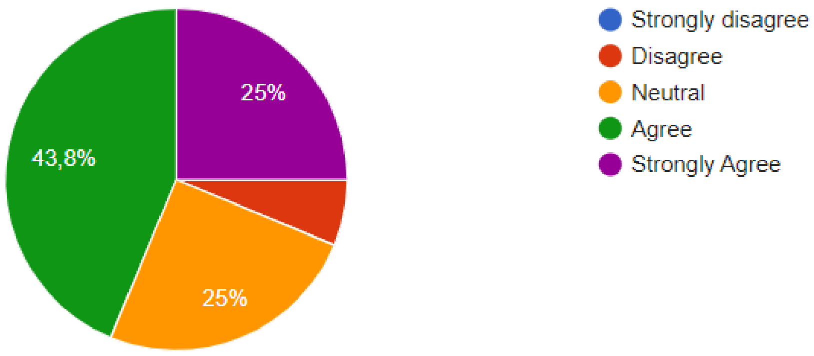
We now begin to discern the impact of explanations on our users:

8. The information provided for the recommended items is sufficient for me to make a decision:



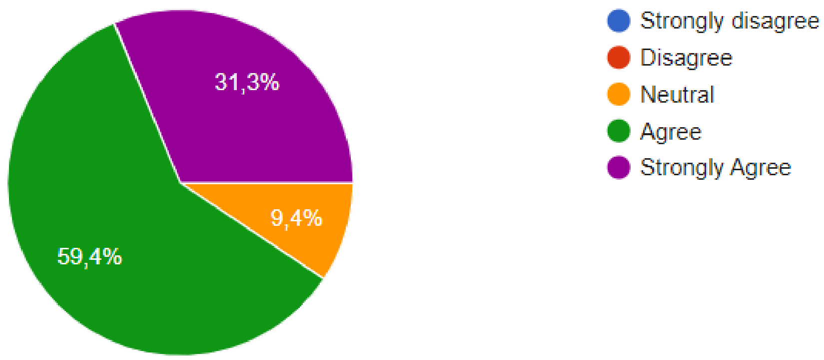
We have also gained a good level of trust:

10. The recommender can be trusted



In facts we see that the recommender pleased the vast majority of them with over 90% of the users that are satisfied with the recommender.

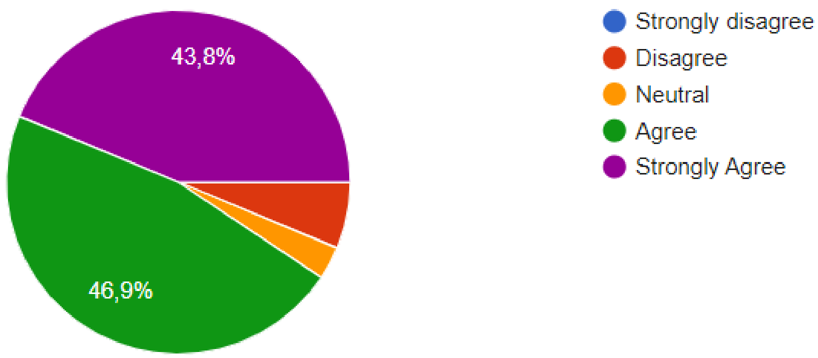
11. Overall, I am satisfied with the recommender:



Let's now focus on the most interesting questions for this experiment.

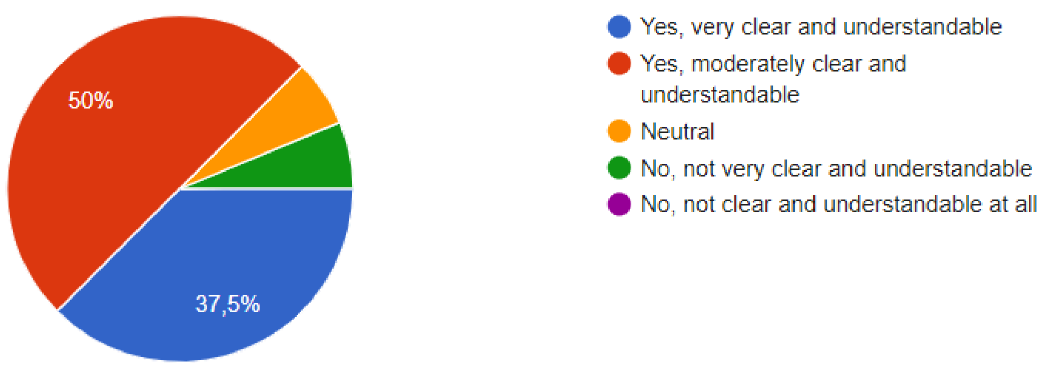
In particular, we can observe how the explanations about the healthiness and sustainability provided some new information to the users, making them more informed. Only a very small portion of the users (less than 10%) disagree with the statement, but we have to keep in mind that there are people that are also well informed about such aspects of recipes.

12. After reading the explanations about the healthiness and sustainability of the recipe, I am more informed about these aspects:



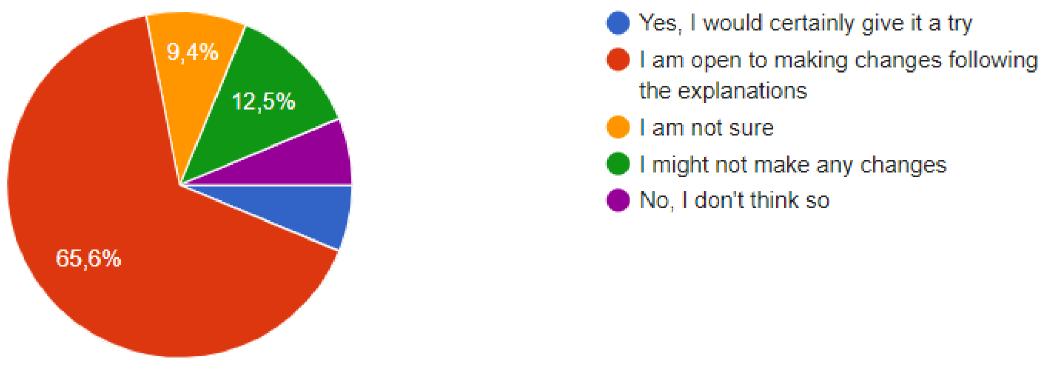
The bot also appears to deliver clear and understandable explanations, as evidenced here, almost 90% of the user agreed:

13. Do you think the explanations provided by the system were clear and understandable regarding the healthiness and sustainability of the recipe?

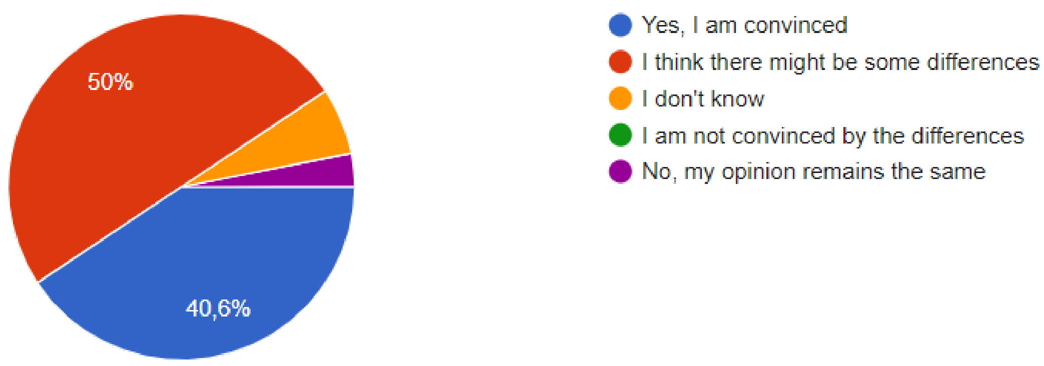


Most importantly, however, we can also see how the bot's explanations can not only change the perception of the users regarding a given recipe but also influence their behaviours:

14. After receiving the explanations, would you change your eating or consumption habits to promote healthiness and sustainability?



15. After receiving the explanations, do you believe that the recipes recommended by the system have a different healthiness and sustainability value than you initially thought?



The significance of the final question cannot be overstated, as it serves as a pivotal point where users may question their existing knowledge and be prompted to construct a new understanding. Remarkably, over 90% of users responded affirmatively, indicating their acknowledgment of the disparities between their initial perceptions of healthiness and sustainability versus the insights gained through the bot's explanations.

5.2. Experimental Data

In our comprehensive dataset sourced from giallozafferano.it, we've meticulously collected details on 4615 recipes, covering a wide array of attributes for each entry, including URL, title, cost, category, imageURL, description, prepTime, cookTime, totalTime, yield, dietary specifications (such as vegetarian, lactose-free, gluten-free), nutritional information (such as calories, carbohydrates, sugars, proteins, fats, saturated fats, fibers, cholesterol, sodium), ingredient measurements, actual ingredients, cooking instructions, and even ratings.

To establish a reliable benchmark (the so-called Ground Truth) for comparing user responses, we've devised a system to evaluate both the healthiness and sustainability of each recommended recipe. Beginning with healthiness, we adhere to the Food Standards Agency (FSA)[31] Guidelines, specifically:

We start by taking the URL of a specific recipe as input, which serves as a reference point. Utilizing this URL, we filter our recipe DataFrame to extract essential nutrient values such as fat, saturated fat, sugars, and sodium from the corresponding row, storing them in a dictionary for streamlined computation. To conform to FSA standards, we normalize these nutrient values by dividing each by 1.2, adjusting from the typical 100 grams to an 80% portion size. The healthiness score, ranging from 4 to 12, is then calculated based on predefined thresholds provided by the FSA, with increments determined by how each nutrient value compares to these thresholds. This score enables us to categorize recipes into five distinct healthiness levels, ranging from "Very Healthy" to "Unhealthy", offering users clear insights into the nutritional quality of each dish.

Turning our attention to sustainability, our approach centers on evaluating recipes' environmental impact, specifically concerning their carbon and water footprints. The process commences by dissecting the ingredients of a given recipe and cross-referencing each ingredient with the data-processing process sourced from the library HeASe[32]. Leveraging the dataset from SU-EATABLE LIFE[33], we obtain information regarding the environmental implications of various ingredients. Through this dataset, we discern the carbon and water footprints associated with each ingredient. Subsequently, we compute the sustainability score utilizing HeASe's established methodology, incorporating a logarithmic normalization step to ensure consistency and reliability across diverse recipes. By computing scores for all recipes, spanning from 0 to 12, and distributing them across percentiles, we assign five labels denoting sustainability levels, ranging from "Very Sustainable" to "Unsustainable".

Collecting user profiles, responses, and opinions, along with generating our Ground Truth, provides us with valuable insights into the usage of the bot:

[31] https://www.food.gov.uk/
[32] https://github.com/GiovTemp/SustainaMeal_Case_Study/tree/main
[33] Petersson, Tashina; Secondi, Luca; Magnani, Andrea; Antonelli, Marta; Dembska, Katarzyna; Valentini, Riccardo; et al. (2021). SU-EATABLE LIFE: a comprehensive database of carbon and water footprints of food commodities. figshare. Dataset. https://doi.org/10.6084/m9.figshare.13271111.v2

As we can see from the tables below, we can observe that the explanation of 'health-benefits' enhances the perception of healthiness across all types of dishes, while the 'goals' explanation improves the perception of healthiness specifically for first courses. Regarding sustainability, the proposed explanations are particularly effective in enhancing the perception of second courses.

Average error PRE explanation = $|PRE\_opinion\_value - Ground\_Truth\_value| \ / \ count$

Average error POST explanation = $|POST\_opinion\_value - Ground\_Truth\_value| \ / \ count$

Overall per dish, regardless of type of explanation:

Topic	**Dish**	**Average error PRE explanation**	**Average error POST explanation**
Healthiness	First course	0.029	0.118
Healthiness	Second course	0.353	0.588
Healthiness	Dessert	1.000	1.382
Sustainability	First course	1.059	1.206
Sustainability	Second course	0.941	0.706
Sustainability	Dessert	1.971	2.059

Overall per explanation, regardless of dish:

Topic	**Type of explanation**	**Average error PRE explanation**	**Average error POST explanation**
Healthiness	Goal	0.222	0,444
Healthiness	Health-benefit	0,521	0,478
Healthiness	Health-risk	0,458	0,875
Healthiness	Macros	0,409	0,863
Sustainability	Ingredients	1,098	1,235
Sustainability	Sustainability	1,500	1,461

Now we combine Dishes and Explanations:

Topic	Dish - Explanation	Average error PRE explanation	Average error POST explanation
Healthiness	First course - Goal	0,083	0
Healthiness	First course - H-benefit	0,222	0,222
Healthiness	First course - H-risk	0	0,714
Healthiness	First course - Macros	0,285	0,428
Healthiness	Second course - Goal	0,076	0,307
Healthiness	Second course - H-benefit	0,666	0,500
Healthiness	Second course - H-risk	0,545	0,545
Healthiness	Second course - Macros	1,500	2,166
Healthiness	Dessert - Goal	0,800	1,200
Healthiness	Dessert - H-benefit	1,500	1,300
Healthiness	Dessert - H-risk	0,857	1,428
Healthiness	Dessert - Macros	0,555	1,333
Sustainability	First course - Ingredients	1,266	1,400
Sustainability	First course - Sustainability	1.000	1,176
Sustainability	Second course - Ingredients	0,875	0,750
Sustainability	Second course - Sustainability	1,041	0,750
Sustainability	Dessert - Ingredients	1,823	1,882
Sustainability	Dessert - Sustainability	2,117	2,235

Hence, we actually gained a decrement in the average errors of:

Topic	Dish	Decrement % on Average error PRE/POST explanation
Sustainability	Second Course	-25.00 %

Topic	Type of explanation	Decrement % on Average error PRE/POST explanation
Healthiness	Health-benefit	-8.24 %
Sustainability	Sustainability	-2.60 %

Topic	Dish - Explanation	Decrement % on Average error PRE/POST explanation
Healthiness	First course - Goal	-8.30 %
Healthiness	Second course - H-benefit	-24.92 %
Healthiness	Dessert - H-benefit	-13.33 %
Sustainability	Second course - Ingredients	-14.29 %
Sustainability	Second course - Sustainability	-28.00 %

6. Conclusions

The conclusions drawn from this study reveal promising implications for promoting healthier dietary choices among users. The research embarked on exploring the effectiveness of the chatbot in influencing user behaviour towards embracing healthier eating habits in an environment inundated with dietary choices. Specifically, it started from the research question: "Can we influence our users' perception of recipes to encourage healthier eating habits by providing them with more information about the recommended recipes?". The study demonstrates that it is possible to change user perception and also behaviours if they're prompted with the right information. First and foremost, the study underscores the positive reception of the recommender system within the chatbot interface. Despite a heterogeneous user group comprising individuals with varying levels of familiarity with conversational agents, the overwhelming majority expressed satisfaction with the recommendations provided. This high level of user satisfaction, exceeding 90%, suggests that the chatbot effectively caters to the diverse needs and preferences of its users, regardless of their prior experience with similar technologies. Such a favourable response underscores the potential of integrating technology-driven solutions, such as chatbots, in promoting healthy dietary choices.

Furthermore, the study sheds light on the transformative potential of the chatbot's explanations in enhancing user knowledge and awareness regarding the healthiness and sustainability of food choices. By providing clear and understandable explanations, the chatbot serves as an informative resource, enriching users' understanding of nutritional concepts and guiding them towards informed decision-making. The significant proportion of users who acknowledged the value of these explanations, coupled with the minimal dissent, underscores the efficacy of the chatbot in imparting new information and fostering a deeper appreciation for healthy eating principles.

Most notably, the study highlights the pivotal role of the chatbot's explanations in influencing user perceptions and behaviours towards food consumption. The bot's explanations not only challenge users' preconceived notions but also prompt them to reconsider their dietary choices and habits. The overwhelmingly affirmative response from users, also here exceeding 90%, indicates a paradigm shift in their understanding of healthiness and sustainability, guided by the insights gleaned from the chatbot's explanations. This suggests that the chatbot can serve as a catalyst for behaviour change, empowering users to make more conscious and informed decisions about their food intake.

Ultimately, the tables displaying the average errors with respect to the ground truth reveal notable enhancements across multiple instances, attributable to the explanatory capabilities of the bot.

In conclusion, the study underscores the potential of integrating a healthy food recommender system within a chatbot interface as a persuasive tool for promoting healthier dietary choices. The high level of user satisfaction, coupled with the transformative impact of the chatbot's explanations on knowledge, perceptions, and behaviours, signifies its effectiveness in fostering positive dietary changes among users.

Future Works

Future works in the context of this study could focus on:

- integrating wearable technology;
- leveraging behavioural nudges and gamification strategies ;
- enhancing multimodal communication within chatbot interfaces

Integration with wearable devices offers the potential for real-time monitoring of users' dietary behaviors and physical activity levels, enabling personalized feedback and support throughout the day. By analyzing data from activity trackers and biometric sensors, the bot could deliver timely reminders, goal-setting prompts, and rewards tailored to individual users' needs and preferences.

Furthermore, incorporating behavioural nudges and gamification elements, such as goal-setting, progress tracking, and rewards systems, can incentivize and motivate users to adhere to healthier eating habits. By transforming dietary goals into achievable milestones and turning healthy eating into an engaging and rewarding experience, the bot can foster sustained behaviour change over time.

Additionally, introducing multimodal communication modalities, such as voice interaction, visual cues, and interactive media, can improve user engagement and comprehension of dietary recommendations. Multimodal communication can enhance the effectiveness and accessibility of chatbot interventions for promoting healthier dietary choices among users.

Integrating these approaches could change how individuals make informed and sustainable dietary decision.

7. Bibliography

1. https://www.statista.com/chart/17518/data-created-in-an-internet-minute/
2. https://telegram.org/
3. https://www.apple.com/siri/
4. https://www.alexa.com/
5. https://assistant.google.com/
6. V. Kumar, A. Dixit, R. R. G. Javalgi e M. Dass, «Research framework, strategies, and applications of intelligent agent technologies (IATs) in marketing,» Journal of the Academy of Marketing Science, vol. 44, n. 1, pp. 24-45, 2016.
7. David L.Poole, Alan K. Mackworth «Artificial Intelligence: Foundations of Computational Agents».
8. https://www.deepmind.com/research/highlighted-research/alphago/
9. https://www.tensorflow.org/about
10. https://www.ibm.com/watson
11. Zhou, H., Zhou, X., Zeng, Z., Zhang, L., & Shen, Z. (2023, February 9). A Comprehensive Survey on Multimodal Recommender Systems: Taxonomy, Evaluation, and Future Directions.
12. Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. Communications of the ACM, 9(1): 36–45, 1966.
13. Jannach, Dietmar, et al. "A survey on conversational recommender systems." ACM Computing Surveys (CSUR) 54.5 (2021): 1-36.
14. Michael Jugovac and Dietmar Jannach. Interacting with recommenders – overview and research directions. TiiS, 7(3):10:1–10:46, 2017. doi: 10.1145/3001837. URL https://doi.org/10.1145/3001837.
15. Lops Pasquale, Musto Cataldo, Narducci Fedelucio, Semeraro Giovanni, "Semantics in Adaptive and Personalised Systems", Springer.
16. https://www.python.org/
17. https://cloud.google.com/dialogflow

18. https://www.yummly.com/
19. https://allrecipes.com
20. https://www.mealime.com/
21. Rostami, M., Farrahi, V., Ahmadian, S., Jalali, S. M. J., Oussalah, M. (2023). A novel healthy and time-aware food recommender system using attributed community detection. Expert Systems with Applications, 221, 119719. https://doi.org/10.1016/j.eswa.2023.119719
22. https://www.giallozafferano.it/
23. I. Paparella, «Progettazione e Implementazione di un Food Recommender System Basato su Holistic User Model» 2020.
24. Lopedota, F. (2022/2023) "FoodRecSysBot": Progettazione e sviluppo di un Agente conversazionale per il supporto personalizzato nella scelta di cibo e ricette.
25. Amir Shevat «Designing Chatbots: Creating Conversational Experiences».
26. https://pypi.org/project/python-telegram-bot/20.7/
27. https://pypi.org/project/inflect/
28. Roberto Polillo «Facile da usare, una moderna introduzione all'ingegneria dell'usabilità», 2010.
29. https://aws.amazon.com/
30. https://aws.amazon.com/ec2/
31. https://www.food.gov.uk/
32. https://github.com/GiovTemp/SustainaMeal_Case_Study/tree/main
33. Petersson, Tashina; Secondi, Luca; Magnani, Andrea; Antonelli, Marta; Dembska, Katarzyna; Valentini, Riccardo; et al. (2021). SU-EATABLE LIFE: a comprehensive database of carbon and water footprints of food commodities. figshare. Dataset. https://doi.org/10.6084/m9.figshare.13271111.v2

8. Acknowledgements

I would like to express my gratitude to all who have contributed in any way to the completion of this work, but in particular:

I am deeply grateful to my thesis supervisor, Professor Cataldo Musto, for his exceptional guidance, insightful advice, and unwavering support throughout every phase of my thesis. His expertise and encouragement have been instrumental in shaping the quality and direction of this work.

To my parents, I want to express my deep appreciation for making this second degree possible for me.

A special thanks goes to my partner, Sara, who has always been by my side with love and understanding. Her presence and support have made every challenge more manageable and every success more meaningful.

And finally I would like to extend my thanks to my colleagues, Francesco and Donato who have lightened the university environment and without whom I would still have many exams ahead in our degree course.